



Universidad de las Ciencias  
Informáticas

**Universidad de las Ciencias Informáticas**

**Facultad 2**

**Aplicación de sincronización de documentos  
del producto XABAL eXcriba basado en  
Alfresco Community 5.2**

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

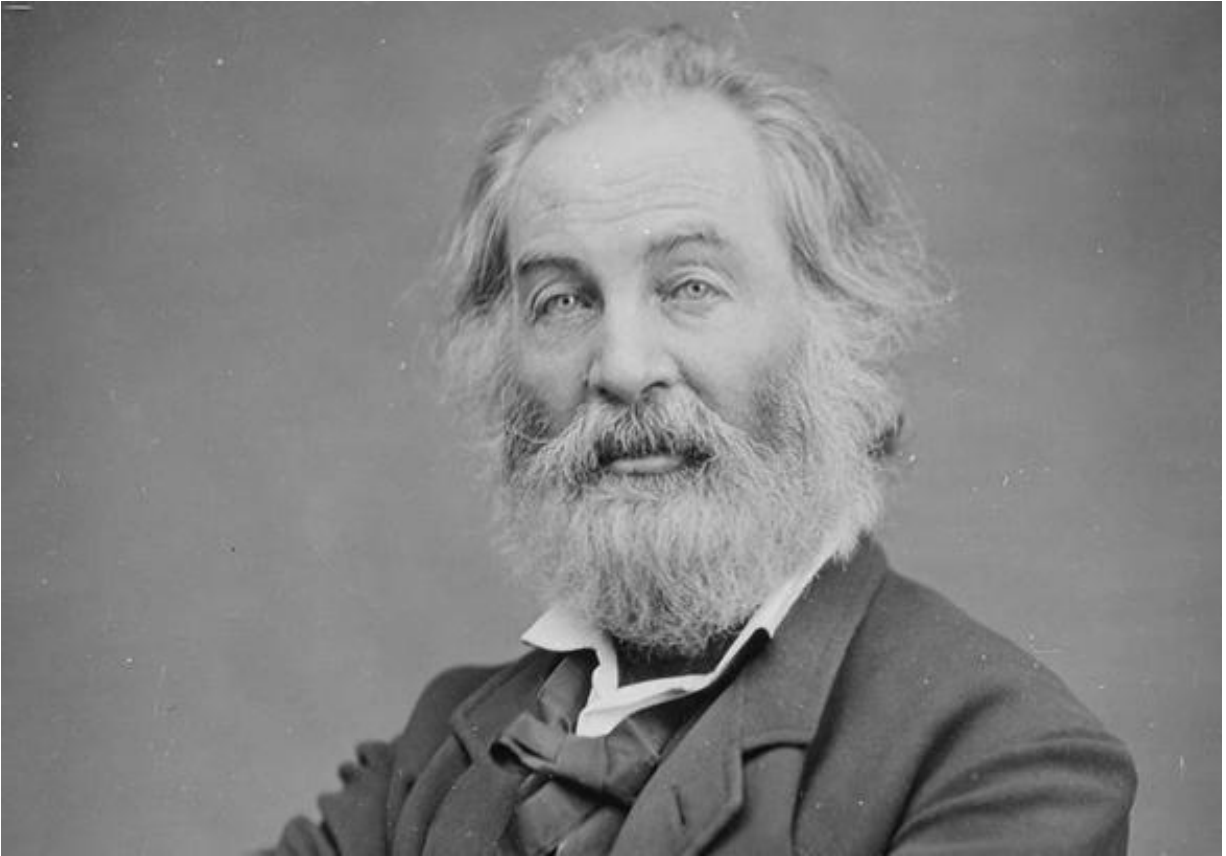
**Autor:** Alejandro González Rodríguez

**Tutores:** MSc. Aurelio Antelo Collado

Ing. Gilberto Enrique González Hidalgo

“Año 61 de la Revolución”

La Habana, Cuba, junio, 2019.



*“No dejes que termine el día sin haber crecido un poco, sin haber sido feliz,  
sin haber aumentado tus sueños. No te dejes vencer por el desaliento.  
No permitas que nadie te quite el derecho a expresarte, que es casi un deber.  
No abandones las ansias de hacer de tu vida algo extraordinario.  
No permitas que la vida te pase a ti sin que la vivas...”*

**Walt Whitman**

## Declaración de autoría

Declaro ser autor de la presente tesis que tiene por título: Aplicación de sincronización de documentos del producto XABAL eXcriba basado en Alfresco 5.2 y reconozco a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo. Para que así conste firmo la presente a los \_\_\_ días del mes de \_\_\_\_\_ del año \_\_\_\_\_.

\_\_\_\_\_  
Alejandro González Rodríguez

Firma autor

\_\_\_\_\_  
MSc. Aurelio Antelo Collado  
Firma tutor

\_\_\_\_\_  
Ing. Gilberto Enrique González Hidalgo  
Firma tutor

## **Datos de contacto**

### **MSc. Aurelio Antelo Collado**

Graduado en el año 1995 como Ingeniero Industrial. Master en Matemática aplicada en el año 200. Se desempeña como Director del Centro de Informatización de la Gestión Documental. Es profesor asistente y ha desempeñado el rol de líder de proyecto.

**Correo:** aantelo@uci.cu

Universidad de las Ciencias Informáticas, La Habana, Cuba

### **Ing. Gilberto Enrique González Hidalgo**

Ingeniero en Ciencias Informáticas 2015. Desarrollador del proyecto “Gestor de Documentos Administrativos Xabal eXcriba”. Especialista B en Ciencias Informáticas / Centro de Informatización de la Gestión Documental (CIGED) / Facultad 2.

**Correo:** gegonzalez@uci.cu.

Universidad de las Ciencias Informáticas, La Habana, Cuba

### **Alejandro González Rodríguez**

**Correo:** agrdguez@estudiantes.uci.cu

Universidad de las Ciencias Informáticas, La Habana, Cuba

## Agradecimientos

*Le agradezco a mis padres por la confianza que han depositado en mí, por apoyarme y comprenderme en todo momento.*

*Gracias por existir los quiero mucho.*

*A mis amigos Daniel, Raidel, Onibag, Reymundo, Iriarte, Pedro, Efrain, Frandier y*

*Hassan por estar ahí siempre que los necesité*

*Al 2502 especialmente a la gente de la canoa que gracias al equipo que formamos*

*llegamos a este punto*

*Gracias a todos*

## **Dedicatoria**

*Le dedico esta tesis a mis padres, mis amigos y a todas las personas que me quieren y apoyan.*

## Resumen

Los sistemas de gestión documental están orientados al despliegue y utilización de recursos para apoyar y lograr los objetivos de una institución o persona en el plano individual. Las plataformas de gestión documental que garantizan la sincronización de los contenidos con los usuarios aportan a estos últimos elementos actualizados a la hora de realizar modificaciones. El Centro de la Informatización de la Gestión Documental (CIGED) se especializa en el desarrollo de sistemas y servicios informáticos integrales de alta calidad y competitividad en la informatización o mejora de la gestión documental. En dicho centro se desarrolló el Gestor de Documentos Administrativos XABAL eXcriba 3.1 que no cuenta con una herramienta externa o funcionalidad añadida que garantice la sincronización de los contenidos descargados y posteriormente modificados con el repositorio documental. A partir de la problemática descrita se plantea como objetivo general desarrollar una herramienta que permita la sincronización de los contenidos generados en los procesos de gestión documental a través de Alfresco Community 5.2 garantizando el trabajo colaborativo. Para ello se utilizaron los lenguajes java 8.0 y UML 2.1 y como herramientas de desarrollo el Visual Paradigm 8.0, Axure RP 7.0, Eclipse 4.9 y JUnit 4.12. Luego del desarrollo de dicha herramienta se garantizó la sincronización con el repositorio documental solucionando así la problemática planteada.

**Palabras clave:** gestión documental, repositorio, sincronización

## **Abstract**

Records management systems are geared to the deployment and use of resources to support and achieve the objectives of an institution or individual at the individual level. The document management platforms that guarantee the synchronization of the contents with the users bring to these last elements updated at the time of making modifications. The Center for Computerization of Documentary Management (CIGED) specializes in the development of high quality and competitive integral computer systems and services for the computerization or improvement of document management. In this centre, the XABAL eXcriba 3.1 Administrative Document Manager was developed, which does not have an external tool or added functionality that guarantees the synchronization of the downloaded and subsequently modified contents with the document repository. Based on the problems described above, the general objective is to develop a tool that allows the synchronization of the contents generated in the document management processes through Alfresco Community 5.2, guaranteeing collaborative work. For this purpose, the languages java 8.0 and UML 2.1 were used and Visual Paradigm 8.0, Axure RP 7.0, Eclipse 4.9 and JUnit 4.12 were used as development tools. After the development of this tool, the synchronization with the document repository was guaranteed, thus solving the problem raised.

**Keywords:** document management, repository, synchronization



# Índice

<b>Capítulo 1. Fundamentación teórica</b> .....	5
<b>Introducción</b> .....	5
<b>1.1 Principales conceptos asociados al dominio del problema</b> .....	5
<b>1.2 Estudio de antecedentes</b> .....	10
<b>1.3 Ambiente de desarrollo</b> .....	11
<b>1.3.1 Metodología de desarrollo</b> .....	12
<b>1.3.2 Lenguaje de modelado</b> .....	14
<b>1.3.3 Lenguaje de programación</b> .....	14
<b>1.3.4 Herramientas y tecnologías</b> .....	15
<b>Conclusiones parciales</b> .....	16
<b>Introducción</b> .....	17
<b>2.1 Propuesta de solución</b> .....	17
<b>2.2 Modelo de dominio</b> .....	17
<b>2.3 Requisitos</b> .....	18
<b>2.3.1 Requisitos funcionales</b> .....	18
<b>2.3.2 Requisitos no Funcionales</b> .....	20
<b>2.4 Diagrama de Casos de Uso del Sistema</b> .....	21
<b>2.4.1 Descripción del Actor</b> .....	21
<b>2.4.2 Especificación de casos de uso</b> .....	22
<b>2.5 Matriz de trazabilidad</b> .....	24
<b>2.6 Arquitectura de software</b> .....	26
<b>2.6 Patrones de diseño</b> .....	27
<b>Conclusiones parciales</b> .....	30
<b>Capítulo 3. Implementación y validación de la propuesta de solución</b> .....	31
<b>Introducción</b> .....	31
<b>3.1 Implementación</b> .....	31
<b>3.1.1 Diagrama de despliegue</b> .....	31
<b>3.1.2 Estándar de codificación</b> .....	32
<b>3.2 Pruebas de software</b> .....	33
<b>3.2.1 Estrategia de prueba</b> .....	33
<b>Conclusiones parciales</b> .....	43

<b>Conclusiones Generales</b> .....	44
<b>Recomendaciones</b> .....	45
<b>Referencias Bibliográficas</b> .....	46
<b>Bibliografía</b> .....	48
<b>Anexos</b> .....	51
<b>Anexo 1 Entrevista</b> .....	51
<b>Anexo 2 Descripción de los casos de uso</b> .....	51
<b>Anexo 3 Diseño caso de prueba</b> .....	59
<b>Anexo 4 Acta de aceptación</b> .....	65

## Índice de Tablas

Tabla. 1 Resultado del estudio de antecedentes.....	11
Tabla. 2 Especificación de los requisitos del software.....	19
Tabla. 3 Descripción del actor .....	21
Tabla. 4 CU9 Sincronizar Documentos .....	22
Tabla. 5 CU3 Configurar Sincronización.....	23
Tabla. 6 Caso de prueba del camino básico 1 .....	37
Tabla. 7 Caso de prueba del camino básico 2.....	37
Tabla. 8 Resultados de las pruebas realizadas con JUnit .....	38
Tabla. 9 Diseño caso de prueba del caso de uso autenticar usuario.....	40
Tabla. 10 Descripción de las variables de entrada autenticar usuario.....	41
Tabla. 11 Diseño caso de prueba del caso de uso configurar sincronización .....	41
Tabla. 12 Descripción de las variables de entrada configurar sincronización .....	42
Tabla. 13 CU1. Autenticar Usuario .....	51
Tabla. 14 CU2 Configurar servidor .....	52
Tabla. 15 CU4. Gestionar Documentos .....	53
Tabla. 16 CU5 Descargar Documentos .....	55
Tabla. 17 CU6. Subir Documentos .....	56
Tabla. 18 CU7 Mostrar estadísticas.....	57
Tabla. 19 CU8 Mostrar datos del usuario .....	58
Tabla. 20 Diseño caso de prueba del caso de uso configurar servidor .....	59
Tabla. 21 Descripción de las variables de entrada configurar servidor .....	60
Tabla. 22 Diseño caso de prueba del caso de uso gestionar documento SC1 mostrar documentos.....	60
Tabla. 23 Diseño caso de prueba del caso de uso gestionar documento SC2 Eliminar documentos.....	61
Tabla. 24 Diseño caso de prueba del caso de uso descargar documentos .....	62
Tabla. 25 Diseño caso de prueba del caso de uso subir documentos.....	63
Tabla. 26 Diseño caso de prueba del caso de uso sincronizar documentos .....	63
Tabla. 27 Diseño caso de prueba del caso de uso mostrar datos del usuario .....	64
Tabla. 28 Diseño caso de prueba del caso de uso mostrar estadísticas .....	64

## Índice de Figuras

Figura. 1 Diagrama del Modelo de domino.....	18
Figura. 2 Diagrama de Caso de Uso del Sistema .....	22
Figura. 3 Matriz de trazabilidad Requisito-Requisito.....	25
Figura. 4 Matriz de trazabilidad Caso de Uso-Requisito .....	25
Figura. 5 Arquitectura de la propuesta de solución (Elaboración propia).....	27
Figura. 6 Patrón experto .....	28
Figura. 7 Patrón creador.....	28
Figura. 8 Patrón bajo acoplamiento (Elaboración propia) .....	29
Figura. 9 Alta cohesión .....	30
Figura. 10 Diagrama de despliegue.....	32
Figura. 11 Método getSites() de la clase Mainclass .....	36
Figura. 12 Método de camino básico (Elaboración propia).....	36
Figura. 13 Gráfico de no conformidades.....	43

## Introducción

El mundo empresarial se ha caracterizado por la necesidad de tomar control total sobre los procesos que se llevan a cabo dentro de una institución, lo cual deriva en la gestión eficaz de la información que da vida a dichos procesos. Hasta hace unas décadas, la documentación producida en cada organización como resultado de transacciones, procesos estratégicos, administrativos y los propios del negocio se almacenaba en formato duro. Así, el constante crecimiento del volumen de documentos afectaba directamente la disponibilidad y recuperación de la información, incluyendo los gastos por concepto de remodelación o adquisición de nuevos locales para el almacenamiento.

En la actualidad con el desarrollo alcanzado por las Tecnologías de la Información y las Comunicaciones (TIC), han surgido nuevas alternativas con el fin de preservar cualquier tipo de información y a su vez añadir valor agregado al conocimiento como recurso clave dentro del desarrollo social y económico de la humanidad. Las TIC facilitan en gran medida la gestión de la información tanto en el plano personal como institucional fundamentalmente, apoyándose en las facilidades que estas brindan en el área de la informática. Las organizaciones tienen a su merced nuevas herramientas y sistemas informáticos que les permiten manejar, conservar y preservar de manera eficaz y eficiente toda la información referente a los procesos generados en su seno, tales sistemas son conocidos como Sistemas de Gestión Documental.

Los sistemas de gestión documental están orientados al despliegue y utilización de recursos para apoyar y lograr los objetivos de una institución o persona en el plano individual. Por tal motivo es una parte imprescindible dentro del sistema de información de las empresas y por consiguiente estas últimas se ven comprometidas con dos importantes aspectos: la gestión de la información originada en las mismas mediante el uso intensivo de las TIC y como segundo aspecto el conocimiento, el cual se deriva de la información generada por la entidad.

Cuba no se encuentra aislada de este desarrollo, la necesidad de establecer programas de administración de documentos en los organismos y empresas estatales, se convierte en un elemento indispensable para el aumento de la eficiencia, la productividad y el cumplimiento de los objetivos que hoy se propone el sistema socialista, en el cual la planificación tiene un papel fundamental. Enmarcados en un esfuerzo coherente demandado por los profundos cambios económicos a los que se enfrenta la humanidad y por supuesto el país, es indispensable el uso efectivo de todos los recursos disponibles, incluyendo, por supuesto, la información y el manejo de la misma de manera eficiente.

La Universidad de las Ciencias Informáticas (UCI) pretende ser el eslabón fundamental del desarrollo de software en Cuba y llevar la informatización a todos los sectores de la sociedad. Condicionado por lo

anterior, en el seno de dicha universidad han surgido diferentes centros productivos, entre los cuales se destaca el Centro de Informatización de la Gestión Documental (CIGED).

Actualmente en dicho centro se lleva a cabo la evolución e integración de los softwares desarrollados en una suit cubana de gestión documental y archivística. Por este motivo se hace de vital importancia la migración hacia una misma tecnología, escogiéndose para ello Alfresco Community en su versión 5.2. la misma es una plataforma de colaboración y gestión de documentos de código abierto. Combina la colaboración avanzada de documentos con la ejecución de procesos para que pueda acelerar la transformación digital de una organización (Shariff, 2014).

Entre los softwares a integrar en dicha suit se encuentra el Gestor de Documentos Administrativos en su versión 3.1 (GDA XABAL eXcriba 3.1) diseñado para tramitar los documentos administrativos que se generan o reciben dentro de las organizaciones. Este producto se encuentra desplegado en varios ministerios y entidades del país y hasta el momento el mayor reclamo de todos los usuarios que explotan sus funcionalidades son los siguientes: para actualizar un documento que ya existe en XABAL-eXcriba y fue descargado y modificado en la PC del usuario, el mismo, para subirlo como nueva versión debe estar conectado con el producto XABAL-eXcriba, esto también ocurre, cuando crea un documento nuevo y necesita subirlo o cuando necesita conocer si existe un documento nuevo en el Gestor de Documentos Administrativos; además, cuando se modifican localmente un grupo de documentos descargados y el usuario debe actualizar la nueva versión uno a uno en el Gestor de Documentos Administrativos.

De acuerdo a lo anteriormente expuesto se plantea el siguiente **problema de investigación**: ¿Cómo mejorar la usabilidad en la gestión documental del sistema GDA XABAL eXcriba 3.1?

El **objeto de estudio** del presente trabajo es el proceso de gestión documental en el sistema GDA XABAL eXcriba 3.1.

De acuerdo al problema a resolver planteado se define como **objetivo general**: Desarrollar una aplicación de escritorio que permita la sincronización de los contenidos generados en los procesos de gestión documental a través de Alfresco Community 5.2 garantizando el trabajo colaborativo.

**Campo de acción**: Sincronización de contenidos en Alfresco 5.2 durante el trabajo colaborativo en el sistema GDA XABAL eXcriba 3.1.

#### **Preguntas científicas:**

1. ¿Cuáles son los referentes teóricos y metodológicos a tener en cuenta para mejorar la usabilidad en la gestión documental del sistema GDA XABAL eXcriba 3.1?
2. ¿Cuál es el estado de la usabilidad en la gestión documental del sistema GDA XABAL eXcriba 3.1?

3. ¿Cómo desarrollar una aplicación de escritorio que permita la sincronización de los contenidos generados en los procesos de gestión documental a través de Alfresco Community 5.2?
4. ¿Qué resultados tendrá la implementación de la aplicación de escritorio que permite la sincronización de los contenidos generados en los procesos de gestión documental a través de Alfresco Community 5.2?

#### **Tareas de la investigación:**

1. Sistematización de los referentes teóricos y metodológicos a tener en cuenta para mejorar la usabilidad en la gestión documental del sistema GDA XABAL eXcriba 3.1.
2. Diagnóstico de la usabilidad en la gestión documental del sistema GDA XABAL eXcriba 3.1.
3. Diseño de una aplicación de escritorio que permita la sincronización de los contenidos generados en los procesos de gestión documental a través de Alfresco Community 5.2.
4. Valoración de la implementación de la solución elaborada.

En el desarrollo de la investigación se utilizan los siguientes métodos de investigación científica.

#### **Métodos teóricos:**

**Analítico-Sintético:** Se estudian las diferentes documentaciones y bibliografías especializadas referentes a los temas de la gestión de documentos, trabajo colaborativo y sincronización de contenidos para arribar a conclusiones que ayuden a comprender mejor el negocio de la herramienta a realizar.

**Modelación:** Se describe el sistema a realizar a través de un modelo conceptual y se modela el mismo con casos de uso de sistema.

**Histórico-Lógico:** Su utilización está dada pues se realizará un estudio acerca de la evolución histórica y las tendencias actuales de la gestión documental.

**Sistémico-Estructural-Funcional:** Se utilizó para hacer un análisis de como se relacionan la sincronización de documentos y la gestión documental a través de Alfresco Community 5.2.

#### **Métodos empíricos:**

**Observación:** Utilizado para tener una visión general de los requerimientos del sistema, las posibles restricciones y propiedades, así como para determinar las posibles soluciones durante todo el proceso de desarrollo del software.

**Entrevista:** este método de investigación fue utilizado para obtener información de forma directa con el cliente para determinar las herramientas que se utilizarían y levantamiento de requisitos funcionales y no funcionales que presenta el sistema. Ver [Anexo 1](#).

El presente documento está compuesto por introducción, desarrollo, conclusiones y bibliografía. El contenido del desarrollo está estructurado en tres capítulos organizados de la siguiente manera:

**Capítulo 1. Fundamentación teórica:** se fundamentan los elementos teóricos de la investigación. Se documenta el estudio e investigación de soluciones informáticas existentes y se describe la metodología de desarrollo, herramientas y tecnologías utilizadas para desarrollar la solución propuesta.

**Capítulo 2. Propuesta de solución:** se exponen los elementos asociados al análisis y el diseño de la solución informática. Se describe la propuesta de solución a través de los requisitos funcionales y no funcionales, muestra la arquitectura y los patrones de diseño, así como otros artefactos en correspondencia con la metodología de desarrollo de software utilizada.

**Capítulo 3. Implementación y validación de la propuesta de solución:** incluye la especificación de los principales aspectos que se corresponden con la implementación de la propuesta de solución. Se define la estrategia de pruebas para la validación de la solución informática y se documentan los resultados de las pruebas desarrolladas.



# **Capítulo 1. Fundamentación teórica**

## **Introducción**

En el presente capítulo se detallan los elementos teóricos que apoyan la investigación y el desarrollo del tema propuesto, a través del análisis y el estudio realizado en busca de soluciones existentes que contribuyan a dar solución al problema de la investigación a través de mecanismos que estas utilicen en soluciones similares. Se abordan diferentes conceptos relacionados con el dominio del problema, además, se seleccionan las herramientas, tecnologías y metodología que se utilizan durante el análisis, diseño e implementación de la solución.

### **1.1 Principales conceptos asociados al dominio del problema**

Para una mejor comprensión del presente trabajo se hace necesario identificar los conceptos fundamentales que se manejan en el ámbito de la gestión documental y la sincronización de documentos entre los que se destacan los siguientes.

#### **Documento**

Información creada o recibida, conservada como información y prueba, por una organización o un individuo en el desarrollo de sus actividades o en virtud de sus obligaciones legales (Norma ISO UNE 15489, 2006).

Un documento es todo registro de información independiente de su soporte físico. Abarca todo lo que puede transmitir el conocimiento humano: libros, láminas, medallas, y de manera general todo lo que tenga carácter representativo en las tres dimensiones (Heredia, 1991).

En otras palabras, un documento es un testimonio material registrado en una unidad de información en cualquier tipo de soporte (papel, cintas, fotografías) en lengua natural o convencional. Es el testimonio de una actividad humana fijada en un soporte.

#### **Gestión documental**

Una de las tareas más antiguas es la gestión documental, la cual surgió de la necesidad de tener constancia de los sucesos ocurridos y de documentar acciones, transacciones comerciales y legales. En sus inicios esta actividad se realizaba mediante la utilización de medios rústicos, pero con el transcurso del tiempo, la complejidad de la misma se iba incrementando debido al gran volumen de datos adquiridos por los documentos generados. Con el surgimiento de Internet el avance y uso de las tecnologías de la información y las comunicaciones, la creación y manipulación de documentos crece de manera desmedida y las distintas instituciones demandan la implantación de nuevos sistemas, métodos y herramientas que resuelvan el problema de la gestión de la información y conocimiento. La mayor parte de las empresas tienen la

necesidad de consultar y acceder frecuentemente a su información archivada, lo que estimula a la investigación de soluciones que aceleren la búsqueda y almacenamiento de la información, así como los procesos que intervienen en la gestión documental.

La gestión documental es el conjunto de operaciones y técnicas relativas a la concepción, al desarrollo, a la implantación y a la evaluación de los sistemas administrativos necesarios, desde la creación de los documentos hasta su destrucción o su transferencia a los archivos (Roberge, 2006). Una característica de la gestión documental es su transversalidad en toda una organización, poniendo énfasis en la vinculación con todas las actividades y otorgando una alta prioridad al valor primario del documento, es decir, a la utilidad de la documentación para la acción del quehacer organizacional. La integración de soluciones tecnológicas puede ser un elemento imprescindible, porque ya no se puede depender exclusivamente de las transferencias físicas en la mayoría de los casos resulta necesaria la implantación de un sistema que ofrezca los datos necesarios sobre los documentos, desde que nacen, hasta que se guardan permanentemente o destruyen (Risso, 2012).

La decisión de utilizar una solución de gestión documental se toma a menudo tras una crisis o una situación tensa causada por la gestión de la información, situación que requiere el establecimiento de una organización más estructurada, de una trazabilidad o una mejor usabilidad. La crisis se manifestaba de diferentes formas: la imposibilidad de encontrar la última versión de un documento en el que se han invertido días de trabajo y las dificultades causadas por el tiempo perdido. El tiempo perdido en búsquedas de documentos y la reutilización de documentos existentes también es problemática cuando se intercambian muchos documentos en la empresa y la eficacia del trabajo puede verse mermada por la falta de organización, lo cual evidencia pérdida de productividad (Soluciones Open Source, 2008).

El término gestión documental es utilizado para enmarcar el control de los documentos dentro de una organización durante todo su ciclo de vida, desde su creación inicial hasta su archivado final. Las empresas tienen un gran vínculo con los documentos, debido a que toda su información está gestionada en estos, es por ello que la gestión correcta de los documentos, dará lugar al incremento de la competitividad de las instituciones.

### **Sistemas de gestión documental**

Debido al enorme volumen de documentación que se maneja en las organizaciones, se ha convertido en todo un reto la administración y control de los documentos, lo cual ha fomentado la adopción de diversas herramientas que favorecen su gestión. Estas aportan mejoras destacadas en el control de los documentos y ofrecen reducciones de tiempos de búsqueda, utilización y ahorro de gastos, muy valioso en los procesos administrativos. La integración de los procesos y controles documentales en los procesos de trabajo debe

ser el objetivo principal de cualquier modelo de gestión de documentos, porque permite: la disminución del tiempo de localización y recuperación de los documentos, la disminución, de espacio físico de almacenamiento y acrecienta la rapidez en la atención al cliente (Risso, 2012).

Un Sistema de Gestión Documental (SGD), es una parte del sistema de información de la empresa desarrollado con el propósito de almacenar y recuperar documentos, que debe estar diseñado para coordinar y controlar todas aquellas funciones y actividades específicas que afectan a la creación, recepción, almacenamiento, acceso y preservación de los documentos, salvaguardando sus características estructurales, contextuales y garantizando su autenticidad y veracidad (Bustelo, 2014).

Los SGD deben tener como objetivo (Mugica, 2015):

- Hacer más fácil a las personas trabajar con los documentos. Cada persona sabe qué documentos tiene que guardar, cuándo, cómo y dónde. Además, debe saber realizar búsqueda de información.
- Facilitar que la información se comparta y se aproveche como un recurso colectivo.
- Conservar la memoria de la organización más allá de los individuos que trabajan en ella y poder aprovechar el valor de los contenidos en los que queda plasmada la experiencia, evitando empezar de cero sobre aspectos en los que ya hay experiencia acumulada.

Ventajas de los sistemas de gestión de documentos

Teresa Allepuz Ros y Carmen Gutiérrez consideran que las ventajas de implantar un SGD puede dividirse en tres grandes bloques (Allepuz, y otros, 2005):

1. Estratégicas, que incluyen consideraciones sobre la marcha general de la empresa y afectan a la mayoría de sus áreas, ejemplo de estas ventajas se encuentran:
  - Hacer más rápidos los procesos de trabajo en los que interviene el manejo de documentación.
  - Asegurarse de que la documentación no es al factor que retrasa los procesos de trabajo.
  - Incrementar la posibilidad de crecimiento, que está limitada por la capacidad de procesar más documentación en papel.
  - Mejorar el servicio al usuario.
  - Mejorar la satisfacción en el trabajo del personal.

- Mejorar la seguridad de los documentos contra incendios o inundaciones.
2. Financieras, que inciden en la reducción de costes y aumento de los ingresos:
- Incrementar la productividad, reduciendo el tiempo necesario para realizar determinados procesos: reduciendo los recursos necesarios, produciendo más con los mismos o aumentando la calidad con los mismos recursos.
  - Ahorrar espacio físico, eliminando todos los espacios dedicados al archivo de documentos en papel.
  - Incrementar el volumen de negocio al poder poner los productos antes en el mercado, o al mejorar la calidad o al aumentar la capacidad de creación de nuevos productos.
3. Técnicos, que presentan la instalación de un SGD como la solución a una serie de problemas existentes dentro de la organización:
- Deficiente gestión de la documentación en papel, que supone dificultades y demoras en la recuperación de la información, documentos de gran valor que no son explotados en su totalidad, pérdida de documentos, entre otros.
  - Lenta comunicación de la información.
  - Duplicidad en el tratamiento de la información que es procesada en sistemas distintos.
  - Retrasos en el procesamiento de la información.

Por todo lo antes expuesto, cabe destacar la importancia que tienen las nuevas herramientas informáticas en los sistemas de gestión documental de última generación para la creación, control y gestión de documentos incluidos en los procesos de negocio.

### **Metadatos**

Como resultado del aumento de grandes cantidades de datos digitales disponibles en la red aparece la sobrecarga de información, situación por la cual se comienzan a usar los metadatos para facilitar la recuperación y catalogación de la misma. Existen diferentes definiciones para acuñar el término metadato, comúnmente son llamados datos sobre datos o información referente a información, otros conceptos se muestran a continuación:

Es la suma total de lo que se puede decir acerca de cualquier objeto de información. En este contexto, un objeto de información es algo que puede ser dirigido y manipulado como una entidad discreta por un ser humano o un sistema de información (Gilliland, y otros, 2016).

Es la información que permite clasificar los documentos en una estructura predefinida, limitar o expandir el acceso y uso, identificar las acciones de disposición o preservación previstas para uno determinado o recoger los eventos que han sucedido a un documento a lo largo del tiempo (Bustelo Ruesta, 2011).

Son datos significativos que representan otros objetos de datos discretos, descripciones estructuradas de un objeto informático y estructuras de organización de la información – legibles por una máquina – cuya finalidad es hacer útiles los datos de distintas formas, según las necesidades concretas de cada servicio de información digital (los archivos digitales también) y según la aplicación que se les otorgue (Méndez, 2017).

Los metadatos pueden ser definidos también como información estructurada que describe, explica, localiza o de lo contrario hace que sea más fácil recuperar, utilizar o administrar determinado recurso de información. Este término puede ser usado para referirse a información comprensible de una máquina, mientras que otros los utilizan solo para describir los registros electrónicos, aunque en ambientes bibliotecarios pueden ser comúnmente usados para la descripción de recursos que sean aplicables a cualquier tipo de objeto digital o no digital siendo la catalogación una forma de uso de los metadatos (NISO, 2004).

En el ámbito de la gestión de documentos, los metadatos se definen como datos que describen el contexto, contenido y estructura de los documentos, así como su gestión a lo largo del tiempo (Norma ISO UNE 15489, 2006).

Según lo observado hasta ahora se puede hacer referencia a un metadato ya sea cuando se hable de un catálogo de biblioteca o cuando se quiera describir un sistema electrónico, por lo que se utiliza la definición de conjunto estructurado de datos cuyo objetivo es proporcionar información, para ello describen características, estructuras y contenido de un determinado objeto de información, incrementando conocimiento de forma tal que se pueda administrar dicho objeto.

### **Sincronización de contenidos**

En informática, sincronizar hace referencia a la coordinación de procesos que se ejecutan simultáneamente para completar una tarea, con el fin de obtener un orden de ejecución correcto y evitar así estados inesperados (Balanov, y otros, 2009).

En el campo de los dispositivos informáticos, sincronizar significa que uno o más archivos se mantengan en la misma versión, es decir que los cambios que se hagan en cualquiera de los documentos, se reflejen en

los todos los dispositivos, que pueden ser computadoras, teléfonos y otros aparatos de computación (Pikovsky, y otros, 2001).

La sincronización de contenidos consiste en mantener la coherencia de los datos contenidos en varias aplicaciones, bases de datos y sistemas. El mismo hace posible que en los sistemas de información los contenidos que se administran de forma independiente por múltiples aplicaciones o bases de datos obtengan la información actualizada generándose un flujo de contenidos que puede ser unidireccional o bidireccional (Vigo Valdés, y otros, 2009).

## **1.2 Estudio de antecedentes**

A continuación, se describen algunas de las soluciones informáticas de interés para la investigación, ya sea por su notoriedad en el mercado o por sus características que las relacionan con el contexto del problema enunciado.

### **Dropbox**

Dropbox es un servicio de alojamiento de archivos multiplataforma en la nube, operado por la compañía Dropbox. El servicio permite a los usuarios almacenar y sincronizar archivos en línea y entre ordenadores y compartir archivos y carpetas con otros usuarios y con tabletas y móviles. Existen versiones gratuitas y de pago, cada una de las cuales tiene opciones variadas. Está disponible para Android, IOS, Windows, Linux y MacOS. (Dropbox.Inc, 2018)

Principales características:

- Permite almacenar cualquier tipo de archivo.
- Crear copias de seguridad en servidores remotos.
- Compartir archivos con contactos o comunidades de usuarios.
- Acceder a los archivos desde cualquier punto de la red y con casi todos los dispositivos disponibles.
- Sincronizar automáticamente los archivos para incorporar las modificaciones.

### **Google Drive**

Google Drive es el servicio de almacenamiento basado en la nube de Google. Proporciona un espacio único para almacenar, crear, modificar, compartir y acceder a documentos, archivos y carpetas de todo tipo, los cuales mantiene sincronizados automáticamente. Los archivos serán accesibles desde el navegador web, el cliente de Google Drive disponible para Windows, Linux, MacOS, IOS y Android. (Google.Inc, 2018)

Principales características:

- Editar sus archivos en el ordenador y tenerlos disponibles en la nube.
- Respaldo automático.
- Control de versiones, pudiendo acceder a versiones anteriores de un archivo después de ser modificado.

- Realizar subidas o bajadas masivas de archivos, respetando la estructura de carpetas.
- Invitar a otros usuarios para que vean, descarguen o modifiquen documentos sin necesidad de enviar archivos adjuntos por correo electrónico.

## Git

Es un sistema de control de versiones libre y de código abierto, diseñado por Linus Torvalds para manejar todo, desde pequeños a grandes proyectos con rapidez y eficiencia. Trabaja pensando en la confiabilidad del mantenimiento de versiones de aplicaciones cuando estas tienen un gran número de archivos de código fuente. Controla todas las modificaciones dentro del código y sabe exactamente los cambios y modificaciones que se hicieron y alerta a los demás desarrolladores para no tener que reescribir el código o romper modificaciones de otros desarrolladores (Git, 2017).

## Resultados del estudio

Después de realizar un análisis de los sistemas de sincronización descritos, para encontrar características y funcionalidades que pudieran servir de base al desarrollo de la propuesta de solución. Se pudo apreciar que la mayoría de estos poseen diversas funcionalidades de sincronización, pero ninguno posee integración con alfresco como es el caso de Dropbox y Google Drive que se especializan en la nube o el Git que está orientado a garantizar la organización de proyectos de desarrollo.

A continuación, se muestra una tabla comparativa de las herramientas estudiadas que realizan la sincronización, los rasgos de comparación se definieron de acuerdo a características que presentan dichas aplicaciones y de acuerdo a las necesidades de la investigación.

Tabla. 1 Resultado del estudio de antecedentes

Indicadores	Aplicaciones		
	Dropbox	Google Drive	Git
Utiliza sincronización de contenidos	X	X	X
Multiplataforma	X	X	X
Código abierto	-	-	X
Posee integración con Alfresco	-	-	-

## 1.3 Ambiente de desarrollo

En aras de preparar el ambiente de desarrollo del software, se realiza la identificación y el análisis de la metodología de desarrollo, lenguaje de programación, entorno de desarrollo, lenguaje y herramienta de

modelado. En esta sección se analizan las principales características y ventajas de cada uno de los elementos antes mencionados que justifican su elección como vía para dar solución a la problemática planteada.

### **1.3.1 Metodología de desarrollo**

Una metodología es una colección de procedimientos, técnicas, herramientas y documentos auxiliares que ayudan a los desarrolladores de software en sus esfuerzos por implementar nuevos sistemas de información. Una metodología está formada por fases, cada una de las cuales se puede dividir en sub-fases, que guiarán a los desarrolladores de sistemas a elegir las técnicas más apropiadas en cada momento del proyecto y también a planificarlo, gestionarlo, controlarlo y evaluarlo (Fitzgerald, 2006).

Los métodos ágiles son métodos de desarrollo incremental donde los incrementos son mínimos y, por lo general, se crean las nuevas liberaciones del sistema, y cada dos o tres semanas se ponen a disposición de los clientes. Involucran a los clientes en el proceso de desarrollo para conseguir una rápida retroalimentación sobre los requerimientos cambiantes. Minimizan la cantidad de documentación con el uso de comunicaciones informales, en vez de reuniones formales con documentos escritos (Sommerville, 2011). Existen numerosas propuestas metodológicas que inciden en distintas dimensiones del proceso de desarrollo de software. Una metodología de desarrollo de software tiene entre sus objetivos aumentar la calidad del software que se produce, de ahí la importancia de aplicar buenas prácticas.

La UCI cuenta con 14 centros productivos en su gran mayoría pertenecientes a las facultades que conforman a la Universidad. Cada uno de estos centros se dedica al desarrollo de software y/o servicios asociados a un dominio de aplicación bien definido. Todos los desarrollos se encuentran organizados en proyectos. Esta diversidad de centros y proyectos hace que la actividad productiva sea cada vez más amplia, y trae consigo la heterogeneidad en el proceso de desarrollo de software. Dicho desarrollo se caracteriza por el uso de diferentes metodologías de desarrollo entre robustas y ágiles, específicamente las nueve metodologías que se listan a continuación (Sánchez, 2015):

- XP
- OPEN UP
- RUP
- BPM
- DAC



- KIMBAL
- SXP
- SCRUM
- NOVA OPEN UP

El Proceso Unificado Ágil o Agile Unified Process (AUP) es una versión simplificada del Proceso Unificado de Racional (RUP). Este describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de software de negocio usando técnicas ágiles conceptos que aún se mantienen válidos en RUP. AUP se preocupa especialmente de la gestión de riesgos. Propone que aquellos elementos con alto riesgo obtengan prioridad en el proceso de desarrollo y sean abordados en etapas tempranas del mismo (Sánchez, 2015).

A pesar de la variedad de metodologías usadas, se ha comprobado que muy pocos proyectos la aplican en su totalidad. Al no existir una metodología de software universal, ya que toda metodología debe ser adaptada a las características de cada proyecto exigiéndose así que el proceso sea configurable. Se decide hacer una variación de la metodología AUP, de forma tal que se adapte al ciclo de vida definido para la actividad productiva de la UCI (Sánchez, 2015).

### **Variación de AUP para la UCI**

AUP-UCI divide el ciclo de vida de la producción del software en tres fases (Sánchez, 2015):

- **Inicio:** Fase donde se realiza un estudio inicial de la organización cliente que permite obtener una información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo y decidir si se ejecuta o no el proyecto.
- **Ejecución:** Fase donde se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto.
- **Cierre:** Fase donde se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto.

Dentro de la fase de Ejecución, dicha metodología propone 7 disciplinas para el ciclo de vida de los proyectos de la UCI, donde sus flujos de trabajos son: modelado de negocio, requisitos, análisis y diseño, implementación, pruebas internas, pruebas de liberación y pruebas de aceptación.

Partiendo que la disciplina Modelado de negocio propone tres variantes a utilizar en los proyectos (casos de uso del negocio, descripción de proceso de negocio y modelo conceptual) y existen tres formas de encapsular los requisitos (casos de uso del sistema, historias de usuario y descripción de requisitos por proceso), surgen cuatro escenarios para modelar el sistema en los proyectos estos son (Sánchez, 2015):

- **Escenario no.1:** Proyectos que modelen el negocio con Caso de Uso del Negocio (CUN) solo pueden modelar el sistema con Caso de Uso del Sistema (CUS).
- **Escenario no.2:** Proyectos que modelen el negocio con Modelo Conceptual solo pueden modelar el sistema con CUS.
- **Escenario no.3:** Proyectos que modelen el negocio con Descripción de Proceso de Negocio (DPN) solo pueden modelar el sistema con Descripción de Requisitos por Proceso (DRP).
- **Escenario no.4:** Proyectos que no modelen negocio solo pueden modelar el sistema con Historias de Usuario (HU).

Se selecciona para esta investigación el escenario número dos pues este aplica a los proyectos que hayan evaluado el negocio a informatizar y como resultado obtengan que no es necesario incluir las responsabilidades de las personas que ejecutan las actividades, de esta forma modelarían exclusivamente los conceptos fundamentales del negocio. Se recomienda este escenario para proyectos donde el objetivo primario es la gestión y presentación de información.

### ***1.3.2 Lenguaje de modelado***

El Lenguaje Unificado de Modelado en su versión 2.1 (Unified Modeling Language, UML por sus siglas en inglés) es el lenguaje estándar especificado por el Object Management Group (OMG) para visualizar, especificar, construir y documentar los artefactos de un sistema, incluyendo su estructura y diseño. Utiliza un conjunto de símbolos y notaciones para representar gráficamente los diversos componentes que forman parte de la arquitectura de software. Permite el modelado de procesos de negocio y el modelado de requisitos apoyándose en el análisis orientado a objetos (Group, 2016).

### ***1.3.3 Lenguaje de programación***

Java es un lenguaje de programación de propósito general orientado a objetos desarrollado por Sun Microsystems. También se puede decir que Java es una tecnología que no sólo se reduce al lenguaje, sino

que además provee de una máquina virtual Java que permite ejecutar código compilado Java, sea cual sea la plataforma que exista por debajo; plataforma tanto hardware, como software (el sistema operativo que soporte ese hardware). El apoyo a esta tecnología viene dado por la gran cantidad de fabricantes que apoyan esta especificación de máquina virtual (UPV, 2018). El tiempo estimado para desarrollar la solución es corto, por lo que no es aconsejable el aprendizaje de otro lenguaje, ya que el equipo tiene experiencia utilizando Java. Para el desarrollo de la solución se utilizará Java en su versión 8.0.

### **1.3.4 Herramientas y tecnologías**

#### **Herramienta de modelado**

Visual Paradigm utiliza UML en su versión 8.0 como lenguaje de modelado. Es una herramienta para el desarrollo de aplicaciones utilizando modelado UML diseñada para ingenieros de software, analistas de sistemas y arquitectos de sistemas. Estos deben estar interesados en la construcción de sistemas a gran escala y necesitan confiabilidad y estabilidad en el desarrollo orientado a objetos. Captura requisitos mediante el modelado de los casos de uso. Permite además exportar los diagramas a imágenes y páginas HTML (Paradigm, 2018).

#### **Entorno de desarrollo integrado**

Un Entorno de Desarrollo Integrado (IDE por sus siglas en inglés) es un entorno de programación que ha sido empaquetado como un programa de aplicación. Consiste en un editor de código de programación, un compilador, un depurador y un constructor de interfaz gráfica (PROGRAMACIÓN DESARROLLO, 2011).

El entorno seleccionado es el **Eclipse 4.9** que es una plataforma de desarrollo con un IDE diseñado para ser extendido a través de plugins. Este IDE no está pensado para un lenguaje en específico, sino que es genérico. Brinda la funcionalidad de programar en el lenguaje Java usando el plugin Herramientas de desarrollo en Java (JDT por sus siglas en inglés) que viene incluido en la distribución estándar del IDE. Dispone además de un editor de texto con un analizador sintáctico y realiza la compilación en tiempo real.

#### **SQLite**

SQLite es una biblioteca en lenguaje C que implementa un motor de base de datos SQL pequeño, rápido, autónomo, de alta fiabilidad y completo. Es el motor de base de datos más utilizado en el mundo. El formato de archivo SQLite es estable, multiplataforma y compatible con versiones anteriores. Los archivos de base de datos se usan comúnmente como contenedores para transferir contenido entre sistemas. El código fuente de SQLite está en el dominio público y es de uso gratuito para todos, para cualquier propósito (SQLite, 2018).

#### **JUnit 4.12**

Es un framework para automatizar las pruebas unitarias de aplicaciones Java en la fase de desarrollo. Este permite controlar la ejecución de las clases y de esta forma evaluar si el funcionamiento de cada uno de los métodos se comporta como se espera (JUnit, 2018). Para realizar esta evaluación se compara el resultado

obtenido durante la ejecución del método con el resultado esperado, partiendo de los datos de entrada necesarios para realizar la prueba. Si ambos resultados son iguales JUnit retorna que el método pasó la prueba correctamente, de lo contrario devuelve un fallo en la misma.

### **Axure RP v 7.0.**

Axure RP es una herramienta rápida de creación de diagramas, prototipos y especificaciones para sitios web (Scientec, 2016). Axure RP se distingue principalmente por su facilidad de uso y riqueza de su diseño, tendrá acceso a:

- Especificaciones automáticas pre-configurables.
- Simulación de ideas de forma rápida.
- Comunicación de ideas más claras.
- Diseño eficiente.
- Ahorro de tiempo con la documentación.

Respecto a la interactividad con otras herramientas, Axure permite la exportación de sus propuestas de prototipado a los siguientes formatos: Html, Word, Cvs, algunos formatos de imagen (Bitmap, PNG, JPG o GIF) y RP (formato propio de Axure). En cambio, sólo permite la importación directa de ficheros en formato RP de Axure.

### **Conclusiones parciales**

El estudio de los principales conceptos referentes a la gestión documental y sincronización permitieron una mejor comprensión del problema de la presente investigación. El análisis realizado a las herramientas de sincronización arrojó que ninguna de estas son soluciones factibles al problema planteado en la investigación. Se seleccionó AUP-UCI como metodología de desarrollo de software. El estudio de los lenguajes, tecnologías y herramientas permitió la selección de UML como lenguaje de modelado y Java versión 8 como lenguaje de programación, Visual Paradigm en su versión 8.0 como herramienta de modelado y el eclipse como IDE de desarrollo.

## **Capítulo 2. Propuesta de solución**

### **Introducción**

Una vez analizadas las soluciones similares y elegidas las herramientas y la metodología a utilizar, se procede a plantear una solución al problema. Para el desarrollo de la solución se siguieron los pasos definidos por la metodología seleccionada que guían el proceso de desarrollo de software. En el presente capítulo se identifican los requisitos funcionales y no funcionales del software, se describen las fases por las que transcurre el desarrollo de la herramienta: la de implementación y la de pruebas, haciendo referencia a todo lo concerniente a ellas, así como una descripción de cada uno de los artefactos generados.

### **2.1 Propuesta de solución**

Se propone la implementación de una aplicación de escritorio desarrollada en java que se comunice con el ECM Alfresco Community 5.2. Este permitirá a un usuario acceder a las áreas de las que es miembro, descargar los documentos de dichas áreas y sincronizarlos una vez que se hayan modificado tanto localmente como en el repositorio documental.

### **2.2 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 (JACOBSON, 2000).

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.

A continuación, se representan los principales conceptos, los objetos más importantes en el contexto del sistema, así como las relaciones entre los mismos:

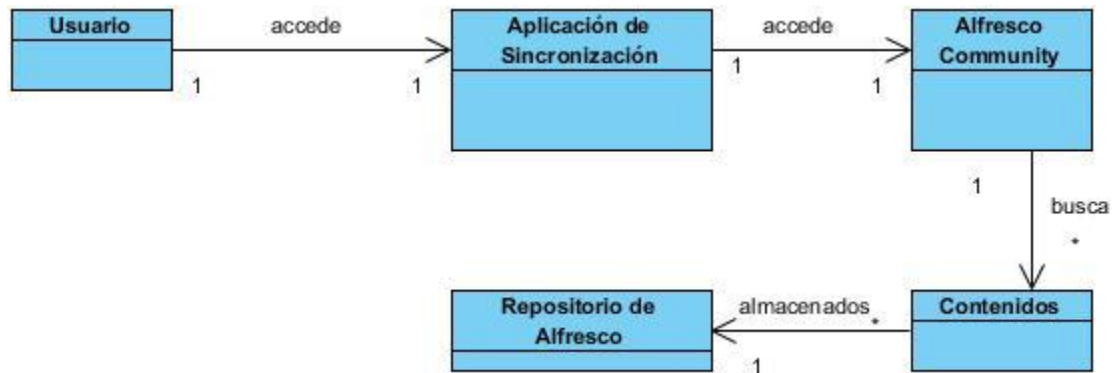


Figura. 1 Diagrama del Modelo de dominio

Descripción de las entidades involucradas en el modelo de dominio:

**Alfresco Community:** Es una plataforma abierta y segura que activa procesos y contenido de manera inteligente para acelerar el flujo del negocio. Ofrece el camino más rápido para que las personas interactúen con la información y para que las empresas respondan a las cambiantes necesidades empresariales (Alfresco Software, 2019).

**Aplicación de Sincronización:** Es la aplicación de escritorio a través de la cual el usuario se comunica con Alfresco Community.

**Contenidos:** Son los documentos electrónicos que se encuentran archivados en el repositorio que contiene Alfresco Community.

**Repositorio de alfresco:** Es el repositorio de contenido de Alfresco, donde se almacena toda la información.

**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 Alfresco Community.

## 2.3 Requisitos

Los requisitos para un sistema son descripciones de lo que el sistema debe hacer: el servicio que ofrece y las restricciones en su operación. Tales requerimientos reflejan las necesidades de los clientes por un sistema que atienda cierto propósito, como sería controlar un dispositivo, colocar un pedido o buscar información. (Sommerville, 2011) En esta sección se describen los requisitos funcionales y no funcionales de la propuesta de solución.

### 2.3.1 Requisitos funcionales

Son enunciados acerca de servicios que el sistema debe proveer, de cómo debería reaccionar el sistema a entradas particulares y de cómo debería comportarse el sistema en situaciones específicas. En algunos

casos, los requerimientos funcionales también explican lo que no debe hacer el sistema. (Sommerville, 2011) A continuación, se enuncian los requisitos funcionales que responden a la propuesta de solución:

RF1: Autenticar usuario

RF2: Configurar sincronización

RF3: Configurar servidor

RF4: Subir documentos

RF5: Descargar documentos

RF6: Mostrar documentos

RF7: Eliminar documentos

RF8: Abrir documentos

RF9: Mostrar estadísticas

RF10: Mostrar datos del usuario

RF11: Sincronizar documentos

*Tabla. 2 Especificación de los requisitos del software*

No	Nombre	Descripción
RF1	Autenticar usuario	Permite al usuario registrado autenticarse.
RF2	Configurar sincronización	Permite seleccionar el área con la que desea sincronizarse y el destino donde se guardarán los documentos sincronizados.
RF3	Configurar servidor	Permite configurar el protocolo, la dirección y el puerto donde se encuentra alojado el servidor.
RF4	Subir documentos	Permite subir uno o varios documentos al repositorio.
RF5	Descargar documentos	Permite descargar a la carpeta destino los documentos o carpetas seleccionados.
RF6	Mostrar documentos	Muestra los árboles documentales, tanto del área seleccionada como los descargados.

RF7	Eliminar documentos	Permite eliminar los documentos descargados y los que se encuentran en el repositorio.
RF8	Abrir documentos	Permite abrir los documentos descargados.
RF9	Mostrar estadísticas	Muestra la cantidad de documentos descargados, subidos además de la fecha en la que se hizo la última sincronización y la última descarga.
RF10	Mostrar datos del usuario	Muestra nombre y apellidos del usuario y su correo electrónico.
RF11	Sincronizar documentos	Verifica que documentos descargados han sido modificados en el repositorio para luego descargarlos si el usuario lo desea.

### **2.3.2 Requisitos no Funcionales**

Son limitaciones sobre servicios o funciones que ofrece el sistema. Incluyen restricciones tanto de temporización y del proceso de desarrollo, como impuestas por los estándares. Los requerimientos no funcionales se suelen aplicar al sistema como un todo, más que a características o a servicios individuales del sistema (Sommerville, 2011). A continuación, se muestran los requisitos no funcionales:

#### **1. RNF-Usabilidad**

- El sistema deberá ser intuitivo, de modo que el esfuerzo para usarlo sea mínimo, además debe ser atractivo para los usuarios.
- Deberá presentar botones organizados por la funcionalidad, de tal manera que permita al usuario una interacción consistente con el mismo.
- Se utilizará el idioma español para los mensajes y textos de la interfaz.

#### **2. RNF- Seguridad:**

- Los usuarios se autenticarán contra Alfresco para poder acceder a sus archivos.

#### **3. RNF-Software:**

- Se requiere la instalación de la máquina virtual de java 1.7 o superior.

#### **4. RNF-Hardware:**

- Para la instalación de la aplicación se debe disponer de una computadora de 512 MB de RAM o superior, un microprocesador de 1 GHz o superior y una interfaz de red.



## 2.4 Diagrama de Casos de Uso del Sistema

Los casos de uso son artefactos narrativos que describen, bajo la forma de acciones y reacciones, el comportamiento del sistema desde el punto de vista del usuario. Por lo tanto, establece un acuerdo entre clientes y desarrolladores sobre las condiciones y posibilidades (requisitos) que debe cumplir el sistema.

### 2.4.1 Descripción del Actor

El actor es quien inicializa los casos de uso. A continuación, el actor implicado en la propuesta de solución del presente trabajo.

Tabla. 3 Descripción del actor

Actor	Descripción
Usuario	Persona que tiene los privilegios de realizar acciones sobre la aplicación.

A partir de la definición de los requisitos se obtuvieron los siguientes casos de uso:

CU1 Autenticar usuario

CU2 Configurar servidor

CU3 Configurar sincronización

CU4 Gestionar documentos

CU5 Descargar documentos

CU6 Subir documentos

CU7 Mostrar estadísticas

CU8 Mostrar datos del usuario

CU9 Sincronizar documentos

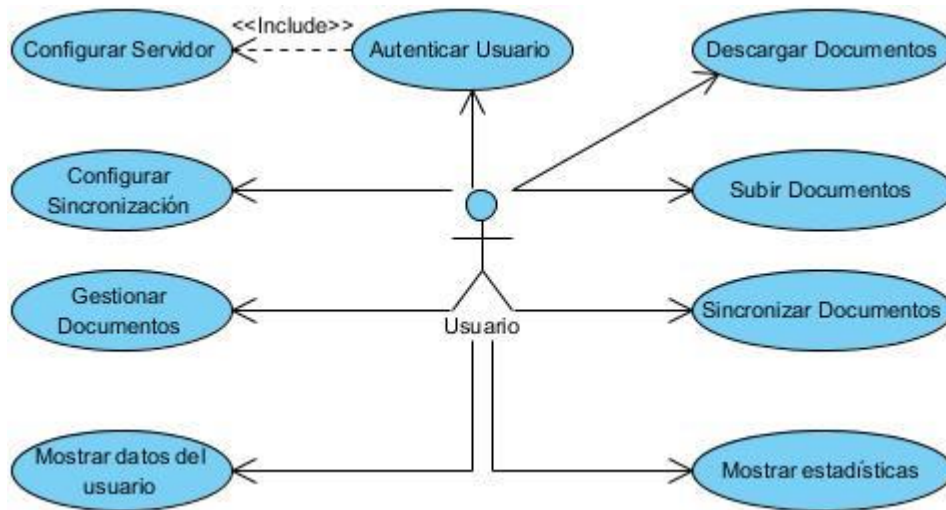


Figura. 2 Diagrama de Caso de Uso del Sistema

### 2.4.2 Especificación de casos de uso

A continuación se muestra la descripción del caso de uso Sincronizar documentos y configurar sincronización, el resto de las descripciones se encuentran en el [Anexo 2](#).

Tabla. 4 CU9 Sincronizar Documentos

<b>Objetivo</b>	Permite al usuario sincronizar los documentos descargados en caso de que hayan sido modificados en Alfresco.	
<b>Actores</b>	Usuario	
<b>Resumen</b>	El caso de uso inicia cuando el actor decide sincronizar los documentos.	
<b>Complejidad</b>	Media	
<b>Prioridad</b>	Media	
<b>Precondiciones</b>	El usuario debe estar autenticado	
<b>Postcondiciones</b>	Se sincronizaron correctamente el /los documentos	
<b>Flujo de eventos</b>		
<b>Flujo básico Gestionar Configuración</b>		
	<b>Actor</b>	<b>Sistema</b>
1.		Muestra la ventana principal de la aplicación.
2.	Presiona el botón "Sincronizar documentos"	
3.		Muestra una ventana con los documentos que hayan sido modificados en el repositorio
4.	Selecciona los documentos que desea sincronizar y presiona el botón "Sincronizar"	
		Se actualizan los documentos descargados.
		Termina Caso de Uso.
<b>Flujos alternos</b>		
4a.Cancelar		

	Actor	Sistema
1.	Presiona el botón "Cancelar"	
		Vuelve a la pantalla principal de la aplicación.
Relaciones	CU incluidos	N/A
	CU extendidos	N/A
<b>Prototipo elemental de interfaz gráfica de usuario</b>		

Tabla. 5 CU3 Configurar Sincronización

<b>Objetivo</b>	Permite al usuario autenticado seleccionar un área y añadir un destino donde se guardarán los archivos sincronizados	
<b>Actores</b>	Usuario	
<b>Resumen</b>	El caso de uso inicia cuando el actor decide configurar la sincronización.	
<b>Complejidad</b>	Baja	
<b>Prioridad</b>	Alta	
<b>Precondiciones</b>	El usuario debe estar autenticado	
<b>Postcondiciones</b>	Se configuró correctamente la sincronización.	
<b>Flujo de eventos</b>		
<b>Flujo básico Gestionar Configuración</b>		
	<b>Actor</b>	<b>Sistema</b>
1.	Selecciona la opción Configurar sincronización.	
2.		Muestra una lista desplegable con las áreas a las que el usuario puede acceder y un campo para seleccionar el destino. Permite guardar y cancelar

3.	Selecciona el área y destino	
4.	Presiona el botón guardar.	
5.		Guarda los datos ingresados y regresa a la ventana principal
6.		Termina caso de uso

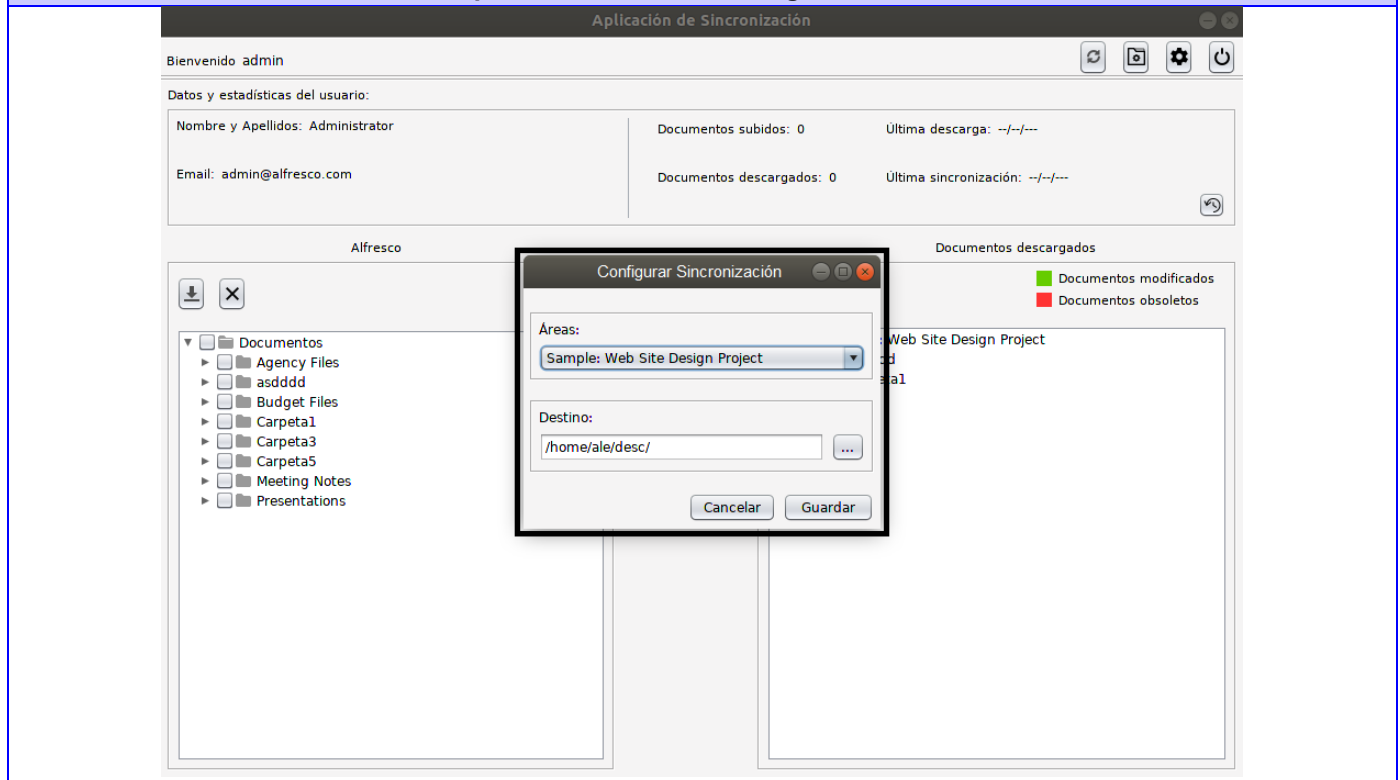
**Flujos alternos**

4a. Cancelar operación

	Actor	Sistema
1.	Presiona el botón cancelar	
2.		Vuelve a la pantalla principal de la aplicación.

<b>Relaciones</b>	<b>CU incluidos</b>	N/A
	<b>CU extendidos</b>	N/A

**Prototipo elemental de interfaz gráfica de usuario**



**2.5 Matriz de trazabilidad**

La matriz de trazabilidad de requisitos es un cuadro que vincula los requisitos del producto desde su origen hasta los entregables que los satisfacen. Proporciona un medio para realizar el seguimiento de los requisitos a lo largo del ciclo de vida del proyecto, lo cual contribuye a asegurar que al final del proyecto se entreguen efectivamente los requisitos aprobados en la documentación de requisitos (Guerrero, y otros, 2016). A continuación, se muestran las matrices de trazabilidad de requisitos funcionales-requisitos funcionales y requisitos funcionales-casos de uso:

(11) Requirement												
By: <input type="text" value="Transitor"/>		Autenticar usuario	Configurar servidor	Configurar sincronización	Descargar documentos	Eliminar documentos	Abrir documentos	Mostrar datos del usuario	Mostrar documentos	Mostrar estadísticas	Sincronizar documentos	Subir documentos
	(11) Requirement	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
	Autenticar usuario		<input checked="" type="checkbox"/>					<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		<input checked="" type="checkbox"/>	
	Configurar servidor	<input checked="" type="checkbox"/>										
	Configurar sincronización				<input checked="" type="checkbox"/>				<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>		
	Descargar documentos			<input checked="" type="checkbox"/>			<input checked="" type="checkbox"/>		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
	Eliminar documentos							<input checked="" type="checkbox"/>				
	Abrir documentos				<input checked="" type="checkbox"/>							
	Mostrar datos del usuario	<input checked="" type="checkbox"/>										
	Mostrar documentos	<input checked="" type="checkbox"/>		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>					<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
	Mostrar estadísticas			<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>						<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Sincronizar documentos	<input checked="" type="checkbox"/>			<input checked="" type="checkbox"/>				<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>			
Subir documentos								<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>			

Figura. 3 Matriz de trazabilidad Requisito-Requisito

(11) Requirement												
By: <input type="text" value="Transitor"/>		Autenticar usuario	Configurar servidor	Configurar sincronización	Descargar documentos	Abrir documentos	Eliminar documentos	Mostrar datos del usuario	Mostrar documentos	Mostrar estadísticas	Sincronizar documentos	Subir documentos
	(9) Use Case	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
	Configurar Servidor		<input checked="" type="checkbox"/>									
	Autenticar Usuario	<input checked="" type="checkbox"/>										
	Configurar Sincronización			<input checked="" type="checkbox"/>								
	Descargar Documentos				<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>						
	Gestionar Documentos						<input checked="" type="checkbox"/>		<input checked="" type="checkbox"/>			
	Mostrar datos del usuario							<input checked="" type="checkbox"/>				
	Mostrar estadísticas									<input checked="" type="checkbox"/>		
	Sincronizar Documentos										<input checked="" type="checkbox"/>	
	Subir Documentos											<input checked="" type="checkbox"/>

Figura. 4 Matriz de trazabilidad Caso de Uso-Requisito

## **2.6 Arquitectura de software**

La Arquitectura de Software es a grandes rasgos, una vista del sistema que incluye los componentes principales del mismo, la conducta de esos componentes según se la percibe desde el resto del sistema y las formas en que los componentes interactúan y se coordinan para alcanzar la misión del sistema. Los elementos son entidades que existen en tiempo de ejecución (objetos, hilos), entidades lógicas que existen en tiempo de desarrollo (clases, componentes) y entidades físicas (nodos, directorios). Las relaciones entre elementos dependen de sus propiedades visibles, quedando ocultos los detalles de implementación (L. Bass, 2003). Por lo cual implementar una arquitectura de software permite representar el diseño a alto nivel del sistema a desarrollar y sirve como guía para el proceso de desarrollo de software; permitiendo dividir el sistema en estructuras arquitectónicas relacionadas, posibilitando la reutilización y el acoplamiento entre las mismas.

### **Arquitectura cliente-servidor**

En una arquitectura cliente-servidor, la funcionalidad del sistema se organiza en servicios, y cada servicio lo entrega un servidor independiente. Los clientes son usuarios de dichos servicios y para utilizarlos ingresan a los servidores. Se usa cuando, desde varias ubicaciones, se tiene que ingresar a los datos en una base de datos compartida. La principal ventaja de este modelo es que los servidores se pueden distribuir a través de una red. La funcionalidad general (por ejemplo, un servicio de impresión) estaría disponible a todos los clientes, así que no necesita implementarse en todos los servicios (Sommerville, 2011).

Los principales componentes de este modelo son:

1. Un conjunto de servidores que ofrecen servicios a otros componentes. Ejemplos de éstos incluyen servidores de impresión; servidores de archivo que brindan servicios de administración de archivos, y un servidor compilador, que proporciona servicios de compilación de lenguaje de programación.
2. Un conjunto de clientes que solicitan los servicios que ofrecen los servidores. Habrá usualmente varias instancias de un programa cliente que se ejecuten de manera concurrente en diferentes computadoras.
3. Una red que permite a los clientes acceder a dichos servicios. La mayoría de los sistemas cliente-servidor se implementan como sistemas distribuidos, conectados mediante protocolos de Internet.

A continuación, se muestra la arquitectura de la propuesta de solución.

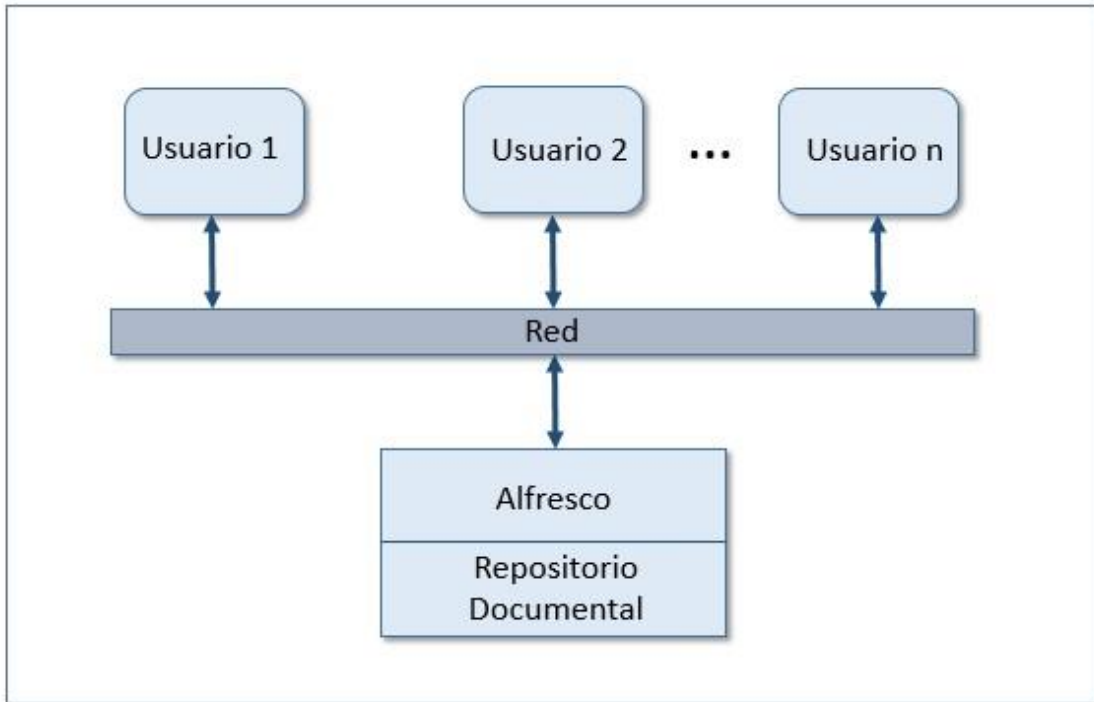


Figura. 5 Arquitectura de la propuesta de solución (Elaboración propia)

## 2.6 Patrones de diseño

Los patrones de diseño son solución general reusable que puede ser aplicada a problemas que ocurren comúnmente en el desarrollo de software, es la descripción o plantilla de cómo resolver un problema que puede ser usada en diferentes situaciones. Constituyen soluciones probadas, expresivas y fáciles de mantener. Muchos desarrolladores están familiarizados con los patrones de diseño, por lo que se puede decir que es un tipo de estándar de desarrollo (Larman, 2003). Por lo que los patrones de diseño brindan una solución ya probada y documentada a problemas de desarrollo de software que están sujetos a contextos similares.

### Patrones de asignación de responsabilidades

Los Patrones Generales de Software para Asignación de Responsabilidades (General Responsibility Assignment Software Patterns, GRASP por sus siglas en inglés) indican que la responsabilidad de la creación de un objeto o la implementación de un método debe recaer sobre la clase que conoce toda la información necesaria para crearlo (Larman, 2003). Son guías o principios que sirven para asignar responsabilidades a las clases.

### Experto

El experto en información establece el principio básico de asignación de responsabilidades. Indica que la

responsabilidad de la creación de un objeto o la implementación de un método, debe recaer sobre la clase que conoce toda la información necesaria para crearlo (Larman, 2003). Por tanto, la utilización de este patrón conserva el encapsulamiento, ya que los objetos se valen de su propia información para hacer lo que se les pide. El comportamiento se distribuye entre las clases que cuentan con la información requerida. Este patrón se evidencia en la clase MainClass específicamente en el método Datos() que es el encargado de crear los nodos del árbol documental, como se muestra a continuación en la Figura 6 Patrón experto.

```

Mainclass.java
555 public Nodo Datos(String parte) throws JSONException {
556     String      createdate;
557     boolean     isFolder;
558     String      modifiedAt;
559     String      createdby;
560     String      name;
561     String      id;
562     String      parentId;
563     Nodo        nodo;
564
565     JSONArray jsonarray = new JSONArray("[+"parte+"]");
566     JSONObject objeto = jsonarray.getJSONObject(0);
567     createdate=objeto.getString("createdAt");
568     modifiedAt=objeto.getString("modifiedAt");
569     createdby=objeto.getString("createdByUser");
570     name=objeto.getString("name");
571     id=objeto.getString("id");
572     parentId=objeto.getString("parentId");
573
574     if(objeto.getString("isFolder").equals("true")) {
575         isFolder=true;
576     }else {
577         isFolder=false;
578     }
579
580
581     nodo=new Nodo(createdate,isFolder,createdby,modifiedAt,name,id,parentId,"",this.getSite().getTitle());
582     return nodo;
583 }

```

Figura. 6 Patrón experto

## Creador

El patrón creador guía la asignación de responsabilidades relacionadas con la creación de objetos, tarea muy frecuente en los sistemas orientados a objetos. El propósito fundamental de este patrón es encontrar un creador que debemos conectar con el objeto producido en cualquier evento. Al escogerlo como creador, se da soporte al bajo acoplamiento (Larman, 2003). Este patrón se evidencia en la clase Window la cual contiene las acciones y es la encargada de ejecutarlas. Dentro de esta se crean varios objetos o instancias de las clases que representan cada una de las funcionalidades. Un ejemplo de esto es el método que se ejecuta al presionar el botón “Configurar Sincronización” como se muestra a continuación en la Figura 7 Patrón creador:

```

Mainclass.java Window.java
884
885 private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {GEN-FIRST:event_jButton4ActionPerformed
886     // TODO add your handling code here:
887
888     ConfigSinc confSinc=new ConfigSinc(princip,this);
889     confSinc.setVisible(true);
890
891 }GEN-LAST:event_jButton4ActionPerformed

```

Figura. 7 Patrón creador



## Bajo acoplamiento

El bajo acoplamiento sostiene la idea de mantener las entidades y clases lo menos ligadas entre sí posible, de tal forma que en caso de producirse modificaciones en alguna de ellas, la repercusión sea la mínima posible en el resto de las entidades o clases (Larman, 2003). Por tanto, este patrón potencia la reutilización, y disminuye la dependencia entre clases. Soporta un diseño de clases más independientes, que reducen el impacto de los cambios, y también más reutilizables, que acrecientan la oportunidad de una mayor productividad. Durante la implementación se definieron un conjunto de clases entre las cuales se establecieron las relaciones necesarias de manera que fueran más independientes y reutilizables. A continuación, la Figura 8 Bajo acoplamiento muestra un ejemplo:

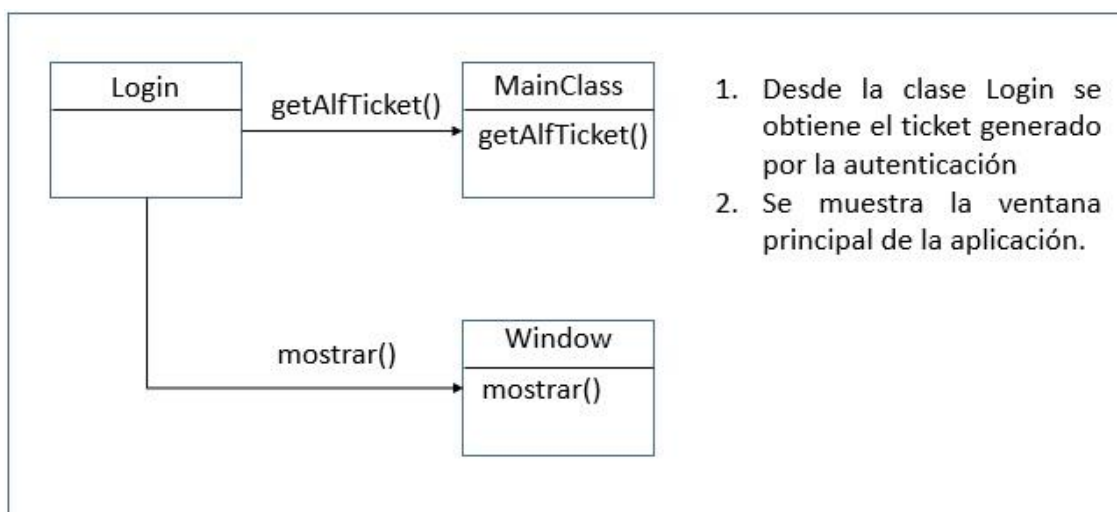


Figura. 8 Patrón bajo acoplamiento (Elaboración propia)

## Alta cohesión

Asignar una responsabilidad de manera que la cohesión permanezca alta. Un elemento con responsabilidades altamente relacionadas, y que no hace una gran cantidad de trabajo, tiene alta cohesión. Estos elementos pueden ser clases, subsistemas y otros. Este patrón incrementa la claridad y facilita la comprensión del diseño. Se simplifican el mantenimiento y las mejoras (Larman, 2003). Fue utilizado en el diseño de los componentes de manera general; donde se agruparon las clases en dependencia de los requerimientos a los que se les debía dar respuesta, según la premisa de que cada clase debe implementar las operaciones que estén sobre la misma área funcional.

```
UploadThread.java
20 public class UploadThread implements Runnable {
21     private JDialogSinc frame;
22     private Window win;
23     private Mainclass main;
24     private ArrayList<Nodo> selectedNodes;
25     private SQLiteDatabase sqLiteDB;
26     private ArrayList<Nodo> fatherList;
27
28     public UploadThread(ArrayList<Nodo> selectedNodes, Mainclass main, Window win, JDialogSinc frame) {
29         // TODO Auto-generated constructor stub
30         this.selectedNodes=selectedNodes;
31         this.win=win;
32         this.main=main;
33         this.frame=frame;
34     }
35
36     @Override
37     public void run() {
38         // TODO Auto-generated method stub
39         fatherList=new ArrayList<>();
40         for(int i=0;i<selectedNodes.size();i++) {
41             System.out.println(selectedNodes.get(i).getId());
42             if(selectedNodes.get(i).getId().equals("")) {
43                 System.out.println("entreeeeeeee"+selectedNodes.get(i).getName());
44                 File file=new File(selectedNodes.get(i).getPath());
45                 String path=file.getParentFile().getPath();
46
47                 if(path.equals(main.getDestino()+main.getSite())) {
48                     try {
49                         main.upload(file, main.gettree().getroot().getId());
50                     }
51                     fatherList.add(main.gettree().getroot());
52                 } catch (HttpException e) {
53                     // TODO Auto-generated catch block
```

Figura. 9 Alta cohesión

## Conclusiones parciales

Se realizó una descripción de la herramienta propuesta, aportando una solución a la problemática planteada. La definición del modelo conceptual facilitó el entendimiento del negocio a informatizar. Al identificar las características que debe tener la propuesta de solución se definieron los requisitos funcionales y requisitos no funcionales. Se modeló el diagrama de casos de uso del sistema para la posterior descripción de los casos de uso que lo componen proporcionando una visión inicial de como pudiera quedar la aplicación. Para el desarrollo de la aplicación se utilizó una arquitectura cliente-servidor, aplicando los patrones de diseño: experto, creador, bajo acoplamiento y alta cohesión.

## Capítulo 3. Implementación y validación de la propuesta de solución

### Introducción

Dentro del proceso de desarrollo de software la etapa de implementación, es el proceso de convertir una especificación del sistema en un sistema ejecutable. La misma implica los procesos de transformar las clases y objetos en ficheros fuente, binarios y ejecutables. En la etapa de pruebas el resultado final, o sea el ejecutable, es evaluado en cuanto a su calidad y desempeño como producto de software.

### 3.1 Implementación

La implementación parte con el resultado del diseño y se implementa el sistema en términos de componentes, es decir ficheros de código, ejecutables. Durante esta fase todos los componentes, características y requisitos deben ser implementados, integrados y probados en su totalidad, obteniendo una versión aceptable del producto (Santiago Mellano, 2014).

#### 3.1.1 Diagrama de despliegue

Un diagrama de despliegue consta de la interconexión de nodos a partir de relaciones de asociación. En este contexto, las relaciones representan enlaces físicos (normalmente bidireccionales). Un nodo es un recurso de ejecución tal como un computador, un dispositivo o memoria. Los estereotipos permiten precisar la naturaleza del equipo: dispositivos, procesadores y memoria (Sibat, 2013).

La aplicación desarrollada posee una arquitectura cliente-servidor. La misma se instala en una PC cliente y guarda todos los datos de sincronización generados en una base de datos SQLite. Dicha aplicación se comunica con una PC servidor que aloja el Alfresco Community 5.2.

El diagrama de despliegue de la propuesta de solución está compuesto por:

**PC Cliente:** Es donde estará alojada la aplicación de sincronización con una base de datos SQLite.

**PC\_Servidor:** Representa el lugar donde estará alojado el Alfresco Community 5.2.

A continuación, se muestra el diagrama de despliegue.

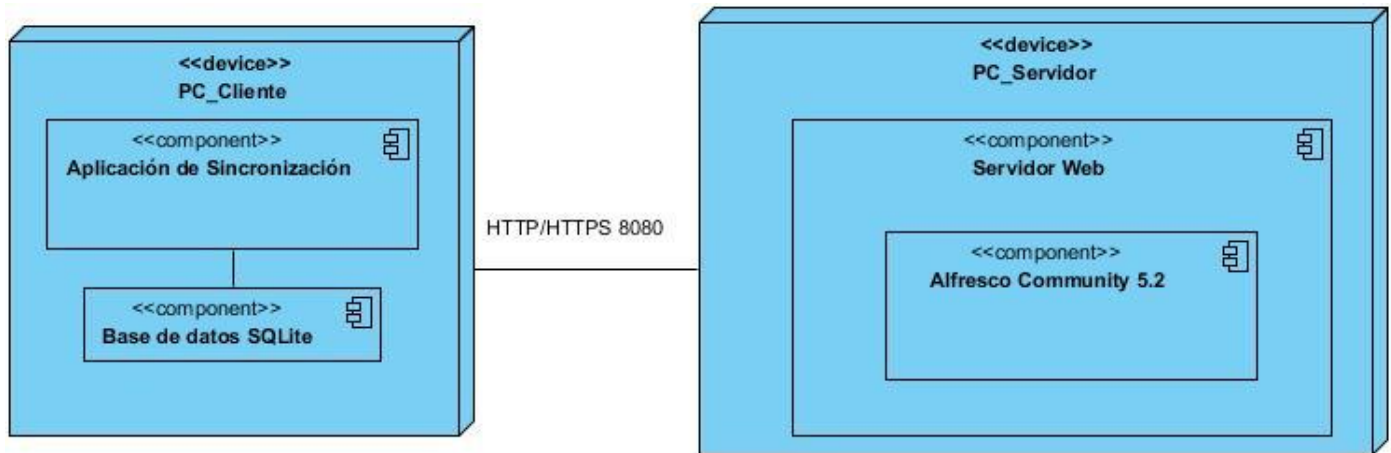


Figura. 10 Diagrama de despliegue

### 3.1.2 Estándar de codificación

Un estándar de codificación es una tecnología, formato o método desarrollado, adoptado a través de proceso abierto de consenso, con la ventaja de facilitar que el código fuente de cualquier aplicación sea legible, se le pueda dar mantenimiento, sea interoperable y pueda ser distribuido (Lugo Robaina, 2017). En la propuesta de solución se utiliza el estándar de codificación de Java, del cual se tomaron los siguientes aspectos:

1. En todos los casos se utilizarán nombres descriptivos que ayuden a una mejor comprensión del código.
2. No se utilizará una misma línea para definir más de una variable y siempre que sea posible estas se inicializarán en su misma línea de declaración.
3. Los nombres de las variables deben ser cortos y significativos.
4. El nombre de la clase será simple y descriptivo.
5. Las clases comenzarán con mayúscula y en caso de estar conformada por palabras compuestas, están unidas y cada palabra debe iniciar con mayúscula siguiendo el estilo determinado.
6. Los métodos deben comenzar con letra minúscula, escribiéndose de manera seguida. En caso de estar conformados por palabras compuestas, a partir de la segunda palabra debe comenzar con letra mayúscula.
7. Las variables comienzan con minúsculas, aquellas que sean compuestas están unidas, comenzando minúscula la segunda palabra.
8. Los nombres de las variables deben ser cortos y significativos.

## 3.2 Pruebas de software

Las pruebas del software son un elemento fundamental para la garantía de calidad del sistema. Estas pruebas representan una revisión final de las especificaciones del diseño y de la codificación, es decir, las pruebas verifican que el software funcione como se diseñó y que los requerimientos son satisfechos, además de brindar soporte para encontrar y documentar defectos del sistema (Pressman, 2010). A continuación, se muestran los casos de pruebas realizados.

La metodología que guía el proceso de desarrollo de la presente investigación propone realizar pruebas internas, de liberación y de aceptación. Sin embargo, para la solución propuesta solo se formalizarán las pruebas internas, y las de aceptación, puesto que se encargan de verificar el resultado de la implementación al probar cada construcción (Sánchez, 2015). La otra prueba que propone la metodología requiere de una entidad certificadora y una aprobación por parte del cliente, elementos que no se contemplan dentro de los requerimientos del presente trabajo.

### 3.2.1 Estrategia de prueba

La estrategia de prueba es lo primero que se debe realizar antes de las pruebas de un software, ya que mediante esta queda plasmado los niveles de prueba a tratar, así como los tipos de pruebas que se deben llevar a cabo por cada nivel, los métodos de pruebas aplicar y las técnicas a utilizar por el método.

Una estrategia de prueba de software proporciona una guía que describe los pasos que deben realizarse como parte de la prueba, cuando se planean y se llevan a cabo dichos pasos, y cuanto esfuerzo, tiempo y recursos se requieran. Por tanto, cualquier estrategia de prueba debe incorporar la planificación de la prueba, el diseño de casos de prueba, la ejecución de la prueba y la recolección y evaluación de los resultados. Una estrategia de prueba de software debe ser suficientemente flexible para promover un uso personalizado de la prueba. Al mismo tiempo, debe ser suficientemente rígida para alentar la planificación razonable y el seguimiento de la gestión conforme avanza el proyecto. (Pressman, 2010)

La metodología AUP-UCI define 3 pruebas, las cuales son (Sánchez, 2015):

**Pruebas internas:** mediante estas pruebas se verifica el resultado de la implementación probando cada construcción, incluyendo tanto las construcciones internas como intermedias, así como las versiones a ser liberadas. Se deben desarrollar artefactos de pruebas como: diseños de casos de prueba, listas de chequeo y de ser posible componentes de pruebas ejecutables para automatizar las pruebas.

**Pruebas de liberación:** pruebas diseñadas y ejecutadas por una entidad certificadora de la calidad externa, a todos los entregables de los proyectos antes de ser entregados al cliente para su aceptación.

**Pruebas de aceptación:** es la prueba final antes del despliegue del sistema. Su objetivo es verificar que el software está listo y que puede ser usado por usuarios finales para ejecutar aquellas funciones y tareas para las cuales el software fue construido.

De las pruebas anteriormente mencionadas se realizaron las pruebas internas y las pruebas de aceptación. Las pruebas internas se realizan mediante la prueba de caja blanca y pruebas de caja negra. Para las pruebas de caja blanca se aplica la prueba de camino básico y las pruebas unitarias. Para las pruebas de caja negra se utiliza la técnica partición de equivalencia. A partir de la realización de las pruebas de aceptación por parte del cliente, el mismo manifestó que el software está listo y que los requisitos definidos funcionan correctamente, como constancia de la prueba el cliente hizo entrega de un acta de aceptación, ver [Anexo 4](#).

### **Prueba de caja blanca**

Las pruebas de caja blanca o pruebas estructurales son un enfoque sistemático a las pruebas donde se usa el conocimiento del código fuente del programa para diseñar pruebas de defecto. La meta es diseñar pruebas que proporcionen algún nivel de cobertura de programa. Esto es, el conjunto de pruebas debe garantizar que se ejecute toda ruta lógica a través del programa, con la consecuencia de que cada enunciado de programa se efectúe al menos una vez. Pueden usarse analizadores de ejecución de programa para demostrar que se logra este nivel de cobertura de pruebas (Sommerville, 2011).

### **Prueba del camino básico**

La prueba de caja blanca realizada a la propuesta de solución fue la prueba del camino básico, a partir del cálculo de la complejidad ciclomática del algoritmo a ser analizado.

Para realizarla se deben enumerar las sentencias de código y a partir de ahí elaborar el grafo de flujo de esta funcionalidad. Se definieron una serie de pasos a seguir (Pressman, 2010):

1. **Notación del grafo de flujo:** usando el código como base se realiza la representación del grafo de flujo, mediante una sencilla notación. Cada construcción estructurada tiene su correspondiente símbolo.
  - **Nodo:** a cada círculo denominado nodo, representa una o más sentencias procedimentales.
  - **Arista:** las flechas del grafo de flujo, denominadas aristas, representan el flujo de control y son análogas a las flechas del diagrama de flujo.
  - **Región:** las áreas delimitadas por aristas y nodos se denominan regiones.

2. **Complejidad ciclomática:** es una métrica que proporciona una medición cuantitativa de la complejidad lógica de un programa. El valor calculado define el número de caminos independientes del conjunto básico de un programa. Esto indica el límite superior para el número de pruebas que se deben realizar, para asegurar que se ejecuta cada sentencia al menos una vez. La complejidad ciclomática tiene fundamentos en la teoría de gráficos y proporciona una medición de software extremadamente útil. La complejidad se calcula de la siguiente forma:

- El número de regiones del gráfico (P) de flujo corresponde a la complejidad ciclomática.
- La complejidad ciclomática  $V(G)$  para un gráfico de flujo G se define como  $V(G) = A - N + 2$  donde A es el número de aristas del gráfico de flujo y N el número de nodos del gráfico de flujo.
- La complejidad ciclomática  $V(G)$  para un gráfico de flujo G también se define como  $V(G) = P + 1$  donde P es el número de nodos predicado (nodos con dos o más salidas) contenidos en el gráfico de flujo.

3. Determinar un conjunto básico de caminos linealmente independientes: el valor de  $V(G)$  es el número de caminos linealmente independientes de la estructura de control del programa.

4. Obtención de casos de prueba: se realizan los casos de pruebas que forzarán la ejecución de cada camino del conjunto básico.

A continuación, se muestra el método al que se le aplica la prueba de camino básico:

```

public ArrayList<NodeSite> getSites() throws IOException, JSONException {
    ArrayList<NodeSite> sites = new ArrayList<>();

    URL url1;
    1  url1 = new URL(protocol:"://" + address:"" + port:"/alfresco/service/api/people/" + this.getUser() + "/sites?alf_ticket=" + ticket);
    URLConnection con = url1.openConnection();
    InputStream in = con.getInputStream();
    String encoding = con.getContentEncoding();
    encoding = encoding == null ? "UTF-8" : encoding;

    String json = IOUtils.toString(in, encoding);
    JSONArray jsonarray = new JSONArray(json);
    2  for(int i=0; i<jsonarray.length(); i++) {
        String id = "";
        String url = "";
        String sitePreset = "";
        String sName = "";
        String title = "";
        String description = "";
        JSONObject jsonObject = jsonarray.getJSONObject(i);
        String idtemp = jsonObject.getString("node");
        3  String [] p;
        p = idtemp.split("\\/");
        id = p[p.length-1];
        url = jsonObject.getString("url");
        sitePreset = jsonObject.getString("sitePreset");
        sName = jsonObject.getString("shortName");
        title = jsonObject.getString("title");
        description = jsonObject.getString("description");
        NodeSite nodeSite = new NodeSite(id, url, sitePreset, sName, title, description);
        sites.add(nodeSite);
    }
    4  return sites;
}

```

Figura. 11 Método getSites() de la clase Mainclass

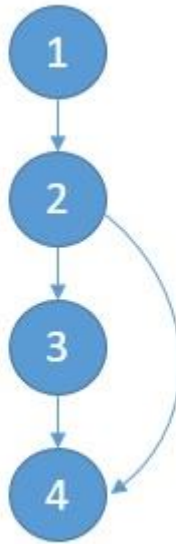


Figura. 12 Método de camino básico (Elaboración propia)

$$V(G)=4-4+2$$

$$=2$$

$$V(G)=1+1$$

$$=2$$

$$V(G)=\text{Regiones}$$

$$=2$$

### Resultados de la prueba del camino básico

El cálculo efectuado mediante las fórmulas ha dado el mismo valor, dando como resultado 2, lo que indica que existen 2 posibles caminos por donde el flujo puede circular, y determina el número de pruebas que se deben realizar para asegurar que se ejecute cada sentencia al menos una vez. El cálculo arrojó que  $V(G)=2$ , por lo que los posibles caminos básicos son:

Camino 1:1,2,3,4

Camino 2:1,2,4



Tabla. 6 Caso de prueba del camino básico 1

Caso de prueba para el camino básico 1	
Descripción: Obtener todas las áreas a las que pertenece el usuario autenticado.	
Condición de ejecución: El usuario debe estar autenticado y pertenecer al menos a un área.	
Datos de entrada	Ticket generado durante la autenticación
Tipos de datos esperados	Lista de las áreas a las que pertenece el usuario
Evaluación del caso de prueba: Satisfactoria	

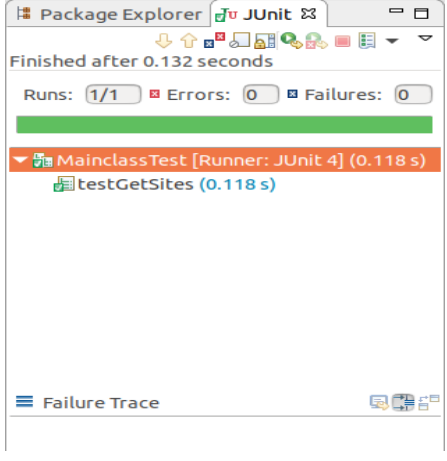
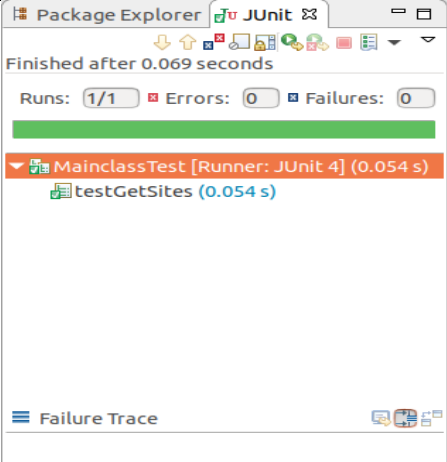
Tabla. 7 Caso de prueba del camino básico 2

Caso de prueba para el camino básico 2	
Descripción: Obtener todas las áreas a las que pertenece el usuario autenticado.	
Condición de ejecución: El usuario debe estar autenticado pero no tiene áreas disponibles	
Datos de entrada	Protocolo, dirección, puerto, usuario y contraseña
Tipos de datos esperados	-
Evaluación del caso de prueba: Satisfactoria	

### Resultados de las pruebas unitarias

Una vez detectados los dos caminos básicos aplicando método de camino básico, estos sirvieron como entrada para la confección de las pruebas en el framework JUnit 4.12 y comprobar si la respuesta del sistema es la esperada, además de medir el tiempo de respuesta como se muestra en la tabla 15. y como referencia de los resultados obtenidos en las tablas 13 y 14.

Tabla. 8 Resultados de las pruebas realizadas con JUnit

Camino	Tiempo en ms	Resultado esperado	Resultado
1	118ms	Si	 <p>The screenshot shows the JUnit test runner interface. It indicates that the test 'testGetSites' passed successfully in 0.118 seconds. The overall test run completed in 0.132 seconds with 1/1 runs, 0 errors, and 0 failures.</p>
2	54ms	Si	 <p>The screenshot shows the JUnit test runner interface. It indicates that the test 'testGetSites' passed successfully in 0.054 seconds. The overall test run completed in 0.069 seconds with 1/1 runs, 0 errors, and 0 failures.</p>

### Pruebas de caja negra

Las pruebas de caja negra también denominadas pruebas de comportamiento, se centran en los requisitos funcionales del software. O sea, permite obtener conjuntos de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa (Pressman, 2010).

La prueba de caja negra intenta encontrar las siguientes categorías de errores (Pressman, 2010):

- Funciones incorrectas o ausentes.
- Errores de interfaz

- Errores en estructuras de datos o en accesos a bases de datos externas.
- Errores de rendimiento.
- Errores de inicialización y de terminación.

### **Técnica de partición equivalente**

Para desarrollar las pruebas de caja negra es posible aplicar varias técnicas, entre ellas (JACOBSON, 2000):

- Técnica de la partición de equivalencia: esta técnica divide el campo de entrada en clases de datos que tienden a ejercitar determinadas funciones del software.
- Técnica del análisis de valores límites: esta técnica prueba la habilidad del programa para manejar datos que se encuentran en los límites aceptables.
- Técnica de grafos de causa-efecto: es una técnica que permite al encargado de la prueba validar complejos conjuntos de acciones y condiciones.

La técnica de partición de equivalencia es considerada una de las técnicas más efectivas, pues permite examinar los valores válidos e inválidos de las entradas existentes en el software. Esta técnica se dirige a la definición de casos de prueba que descubran clases de errores, reduciendo así el número total de casos de prueba que hay que desarrollar. Una clase de equivalencia representa un conjunto de estados válidos o no válidos para condiciones de entrada. Por lo general, una condición de entrada es un valor numérico específico, un rango de valores, un conjunto de valores relacionados o una condición lógica (JACOBSON, 2000).

### **Casos de prueba**

Para la realización de las pruebas al módulo de gestión de nomencladores se generó el artefacto “Diseño de casos de pruebas basado en casos de uso”. Un caso de prueba es un conjunto de entradas de pruebas, condiciones de ejecución, resultados esperados desarrollados para cumplir un objetivo en particular o una función esperada. La entidad más simple que siempre es ejecutada como una unidad, desde el comienzo hasta el final (Ferrer Rodríguez, 2013).

Los casos de pruebas deben verificar (Ferrer Rodríguez, 2013):

- Si el producto satisface los requisitos del usuario, tal y como se describe en las especificaciones.

- Si el producto se comporta como se desea, tal y como se describe en las especificaciones funcionales del diseño.

Un caso de prueba (CP) se diseña según las funcionalidades descritas en los escenarios. Este es elaborado previo a la realización de las pruebas funcionales de la aplicación. Cada planilla de caso de prueba recoge la especificación de un escenario, dividido en secciones, para hacer más fructífera la ejecución de las pruebas. En esta investigación fueron diseñados 10 casos de pruebas. A continuación, se muestra la SC1 Crear nomenclador del CP Gestionar nomenclador y en el Anexo 5 Casos de prueba, se describen el resto de los casos de pruebas por escenarios.

A continuación se muestra el diseño del caso de prueba autenticar usuario y configurar sincronización el resto de los casos de prueba se encuentran en el [Anexo 3](#).

Tabla. 9 Diseño caso de prueba del caso de uso autenticar usuario

Escenarios	Descripción	Usuario	Contraseña	Respuesta del sistema	Flujo central
EC 1.1 Autenticar usuario correctamente	Permite al usuario acceder a la aplicación proporcionando los datos correctamente	V	V	El usuario accede al sistema correctamente	1-Introducir los datos de acceso (usuario y contraseña). 2- Presionar enter o dar click en el botón acceder.
		admin	admin		
EC 1.2 Autenticar usuario con datos incorrectos	El usuario introduce datos incorrectos en el formulario de inicio de sesión.	V	I	Muestra un mensaje de error: "Existen datos incorrectos"	1-Introducir los datos incorrectos de acceso. 2- Presionar enter o dar click en el botón "Acceder".
		admin	asd		
		I	V		
		admin%*	admin		
		I	I		

		admin%*	tty		
EC 1.3 Autenticar usuario con campos vacíos	El usuario deja campos vacíos en el formulario de inicio de sesión	V	I	Muestra el mensaje: "Existen campos vacíos"	1-Dejar campos vacíos. 2-Presionar enter o dar click en el botón "Acceder".
		admin			
		I	V		
			admin		
		I	I		

Tabla. 10 Descripción de las variables de entrada autenticar usuario

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	Usuario	Campo de texto	No	Solo admite letras
2	Contraseña	Campo de texto	No	Admite cualquier caracter

Tabla. 11 Diseño caso de prueba del caso de uso configurar sincronización

Escenarios	Descripción	Área	Destino	Respuesta del sistema	Flujo central
EC 1.1 Configurar sincronización correctamente	Permite al usuario configurar la sincronización de la aplicación mediante la selección del área y el destino	V	V	Guarda los datos introducidos para la posterior sincronización	1-Presionar el botón "Configurar sincronización". 2-Selecciona los datos y presionar el botón "Aceptar".
		Selecciona el área deseada	/home/descarga/		

EC 1.2 Configurar sincronización con campos vacíos	El usuario deja campos vacíos en el formulario de configurar sincronización.	V	V	Muestra la ventana principal de la aplicación con los documentos del área seleccionada y los documentos en el destino si lo introdujo	1- Presionar el botón "Configurar sincronización".  2- Selecciona los datos y presionar el botón "Aceptar".
		Selecciona el área deseada			
		I	V	Muestra el mensaje "Para sincronizar debe seleccionar un área"	
			/home/descarga/		

Tabla. 12 Descripción de las variables de entrada configurar sincronización

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	Área	Campo de selección	No	Permite al usuario seleccionar un área
2	Destino	Campo de selección	Si	Permite al usuario seleccionar el destino

Para validar el correcto funcionamiento del sistema se realizaron las pruebas de caja negra a través de los casos de prueba basados en los casos de uso. Este proceso permitió verificar el cumplimiento de los requisitos funcionales del componente, donde los resultados de las pruebas que no fueron satisfactorios pasaron a ser no conformidades.

A continuación, se muestra un gráfico con los datos de las no conformidades encontradas en cada iteración. En la primera iteración se identificaron once no conformidades, de ellas, dos de validación, una de ortografía, cinco funcionales y tres de la interfaz. En la segunda iteración se identificaron dos no conformidades, ambas funcionales y fueron resueltas. En una tercera iteración no se detectó ninguna no conformidad por lo que la aplicación mostró un buen funcionamiento.

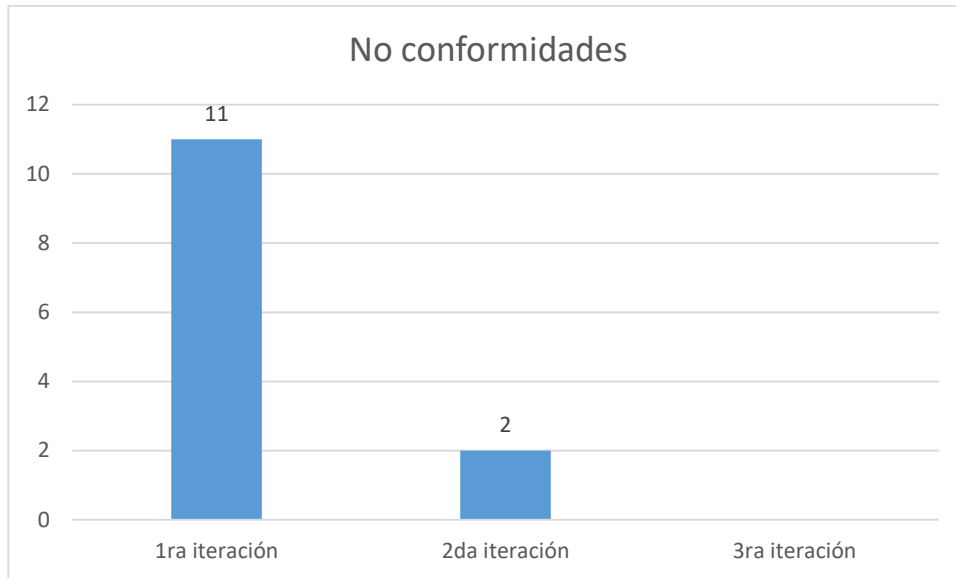


Figura. 13 Gráfico de no conformidades

### Conclusiones parciales

A partir de los resultados obtenidos en el presente capítulo se implementó la aplicación de sincronización de documentos del producto XABAL eXcriba 3.1 basado en alfresco 5.2. El diagrama de despliegue aportó una forma en la que se puede comunicar la aplicación con su entorno. Se describieron los aspectos utilizados para la codificación del sistema bajo el estándar de codificación de java. Se definió la estrategia de prueba a seguir que permitió la verificación de las funcionalidades del sistema.

## **Conclusiones Generales**

Al culminar la presente investigación y la implementación del componente propuesto, se concluye que:

1. Se realizó un estudio de los referentes teóricos y metodológicos a tener en cuenta para la solución del problema planteado relacionado con la usabilidad durante gestión documental en el sistema GDA XABAL eXcriba 3.1.
2. El estudio del estado actual de las herramientas informáticas de sincronización, evidenció la necesidad de desarrollar el componente propuesto, ya que ninguna de las aplicaciones analizadas constituye una solución al problema planteado en la presente investigación.
3. Se implementó la aplicación de sincronización contribuyendo a la mejora de la usabilidad durante la gestión documental en el sistema GDA XABAL eXcriba 3.1.
4. La definición de la estrategia de prueba y la posterior aplicación de las pruebas internas y de aceptación contribuyeron a la validación del componente garantizando su correcto funcionamiento.



## **Recomendaciones**

Teniendo en cuenta el estudio realizado durante todo el proceso de desarrollo de la herramienta y en aras de enriquecer la solución, el autor recomienda:

1. Definir un mecanismo de actualización para actualizar de forma automática el cliente de la solución implementada.
2. Incorporar a la solución un módulo de notificaciones.

## Referencias Bibliográficas

**Alfresco Software, Inc. 2019.** *Alfresco*. [En línea] 2019. [Citado el: 14 de mayo de 2019.]  
<https://www.alfresco.com/es/>.

**Allepuz, Teresa Ros y Gutiérrez, Carmen. 2005.** *Los sistemas de gestión de la documentación en las organizaciones*. s.l. : Gabinete de Asesores Documentalistas, 2005. ISBN 978-0-89236-896-9.

**Balanov, Alexander, y otros. 2009.** *Synchronization: From Simple to Complex*. Berlin : Springer-Verlag Berlin Heidelberg, 2009. ISBN 978-3-540-72127-7.

**Bustelo Ruesta, Carlota. 2011.** *Los grandes temas relacionados con la gestión de documentos: desafíos y oportunidades. El profesional de la información*. Bilbao : s.n., 2011.

**Bustelo, Carlota Ruesta. 2014.** *Gestión Documental en las Empresas: Una aproximación práctica*. Bilbao : s.n., 2014. ISBN: 978-0-470-37359-0.

**Dropbox.Inc. 2018.** <https://www.dropbox.com/es/>. [En línea] 2018. [Citado el: 14 de marzo de 2019.]

**Ferrer Rodríguez, Orquidia. 2013.** *Sistema de gestión de la información para las ópticas en cuba*. La Habana : s.n., 2013.

**Fitzgerald, D. E. Avison y G. 2006.** *Information system development*. s.l. : McGraw-Hill Education, 2006. ISBN.

**Gilliland, Anne J, y otros. 2016.** *Introduction to Metadata second edition*. Getty : s.n., 2016. ISBN 896-9-978-0-8923-6.

**Git. 2017.** <https://git-scm.com/>. [En línea] 2017. [Citado el: 14 de marzo de 2019.]

**Google.Inc. 2018.** <https://www.google.com/drive/>. [En línea] 2018. [Citado el: 17 de marzo de 2019.]

**Group, Object Managment. 2016.** Object Managment Group. [En línea] 2016. [Citado el: 26 de marzo de 2019.]  
<https://www.omg.org/spec/UML/About-UML/>.

**Guerrero, C. y Martín, D. 2016.** *Recopilar requisitos*. Madrid : s.n., 2016.

**Heredia, Antonia Heredia. 1991.** *Archivística General Teoría y Práctica*. Sevilla : s.n., 1991. I.S.B.N. 84 - 7798 - 056 - 5..

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

**JUnit. 2018.** JUnit. [En línea] 12 de abril de 2018. [Citado el: 21 de marzo de 2019.] <https://junit.org>.

**L. Bass, P. y Clements, R. K. 2003.** *Software Achitecture in Practice*. s.l. : Addison Wesley, 2003.

**Larman, Craig. 2003.** *UML y Patrones*. s.l. : 2da Edición, 2003.

**Lugo Robaina, B., Rodríguez Guerra, L. 2017.** *Herramienta informática para el diagnóstico inicial en el*. La Habana : s.n., 2017.

**Méndez, Eva Rodríguez. 2017.** *La descripción de documentos electrónicos a través de metadatos: una visión para la archivística desde la nueva e-administración*. s.l. : d'Arxius, 2017.

- Mugica, Mayra Mena. 2015.** *Gestión documental y organización de archivos*. La Habana : : Felix Varela, 2015.
- NISO. 2004.** *Understanding Metadata*. s.l. : NISO Press, 2004. ISBN 9-958-124-62-1.
- Norma ISO UNE 15489. 2006.** *Información y documentación: gestión de documentos*. Madrid : AENOR, 2006.
- Paradigm, Visual. 2018.** UML CASE tool for software development. UML CASE tool for software. [En línea] 2018. [Citado el: 26 de marzo de 2019.] <https://www.visual-paradigm.com/>.
- Pikovsky, Arkady, Rosenblun, Michael y Kurths, Jurgen. 2001.** *Synchronization a Universal Concept in Nonlinear Sciences*. Potsdam : Cambrige Nonlinear Science Series, 2001. ISBN-13 978-0511-07595-7.
- Pressman, Roger S. 2010.** *Ingeniería del software. Un enfoque práctico*. New York : s.n., 2010.
- PROGRAMACIÓN DESARROLLO. 2011.** Qué es un entorno de desarrollo, IDE. [En línea] 15 de febrero de 2011. [Citado el: 26 de marzo de 2019.] <http://programaciondesarrollo.es/que-es-un-entorno-de-desarrollointegrado-ide/>.
- Risso, Verónica Gauchi. 2012.** *Aproximación teórica a la relación entre los términos gestión documental, gestión de información y gestión del conocimiento*. Granada : s.n., 2012. 0210-0614.
- Roberge, Michel. 2006.** *Lo esencial de la gestión documental: sistema integrado de gestión de los documentos analógicos y de los documentos electrónicos*. s.l. : Solutions Documentaires Gestar, 2006.
- Rodríguez, T. Genebeta:dev Eclipse IDE. 2015.** [En línea] 10 de enero de 2015. [Citado el: 26 de marzo de 2018.] <http://www.genbetadev.com/herramientas/eclipse-ide>.
- Sánchez, Tamara Rodríguez. 2015.** *Metodología de desarrollo para la Actividad productiva UCI*. La Habana : s.n., 2015.
- Santiago Mellano, Cecilia Molina, and Germán Pratelli. 2014.** *Sistema para la Administración de Programas y Proyectos de Investigación coordinados por la Secretaría de Ciencia y Técnica*. Universidad Nacional de Río Cuarto : s.n., 2014.
- Scientec. 2016.** Scientec. [En línea] 17 de octubre de 2016. [Citado el: 26 de marzo de 2019.] <https://www.scientec.com.mx/axure-rp/>.
- Shariff, Munwar. 2014.** *Enterprise Content Management Implementation*. 2014. ISBN 1-904811-11-6.
- Sibat, Martinez Yanet. 2013.** *La ejecución de flujos de trabajo de aprobación*. 2013.
- Soluciones Open Source. 2008.** *Gestión Documental Open Source*. 2008. ISBN 1-880124-62-9..
- Sommerville, Ian. 2011.** *Ingeniería de Software*. México : PEARSON EDUCACIÓN, 2011.
- SQLite. 2018.** <https://www.sqlite.org/>. [En línea] 2018. [Citado el: 15 de abril de 2019.]
- UPV. 2018.** UPV. [En línea] 2018. [Citado el: 26 de marzo de 2019.] <http://personales.upv.es/rmartin/cursoJava/Java/Introduccion/PrincipalesCaracteristicas.htm>.
- Vigo Valdés, Yahima, Marrero Palomino, Antonio y Sánchez Tellez, Eddy. 2009.** *Sincronización para centros de gestión de emergencias*. Habana : s.n., 2009. Vol. 3.

## Bibliografía

**Alfresco Software, Inc. 2019.** *Alfresco*. [En línea] 2019. [Citado el: 14 de mayo de 2019.]  
<https://www.alfresco.com/es/>.

**Allepuz, Teresa Ros y Gutiérrez, Carmen. 2005.** *Los sistemas de gestión de la documentación en las organizaciones*. s.l. : Gabinete de Asesores Documentalistas, 2005. ISBN 978-0-89236-896-9.

**Balanov, Alexander, y otros. 2009.** *Synchronization: From Simple to Complex*. Berlin : Springer-Verlag Berlin Heidelberg, 2009. ISBN 978-3-540-72127-7.

**Bustelo Ruesta, Carlota. 2011.** *Los grandes temas relacionados con la gestión de documentos: desafíos y oportunidades. El profesional de la información*. Bilbao : s.n., 2011.

**Bustelo, Carlota Ruesta. 2014.** *Gestión Documental en las Empresas: Una aproximación práctica*. Bilbao : s.n., 2014. ISBN: 978-0-470-37359-0.

**Dropbox.Inc. 2018.** <https://www.dropbox.com/es/>. [En línea] 2018. [Citado el: 14 de marzo de 2019.]

**Ferrer Rodríguez, Orquidia. 2013.** *Sistema de gestión de la información para las ópticas en cuba*. La Habana : s.n., 2013.

**Fitzgerald, D. E. Avison y G. 2006.** *Information system development*. s.l. : McGraw-Hill Education, 2006. ISBN.

**Gilliland, Anne J, y otros. 2016.** *Introduction to Metadata second edition*. Getty : s.n., 2016. ISBN 896-9-978-0-8923-6.

**Git. 2017.** <https://git-scm.com/>. [En línea] 2017. [Citado el: 14 de marzo de 2019.]

**Google.Inc. 2018.** <https://www.google.com/drive/>. [En línea] 2018. [Citado el: 17 de marzo de 2019.]

**Group, Object Managment. 2016.** Object Managment Group. [En línea] 2016. [Citado el: 26 de marzo de 2019.]  
<https://www.omg.org/spec/UML/About-UML/>.

**Guerrero, C. y Martín, D. 2016.** *Recopilar requisitos*. Madrid : s.n., 2016.

**Heredia, Antonia Heredia. 1991.** *Archivística General Teoría y Práctica*. Sevilla : s.n., 1991. I.S.B.N. 84 - 7798 - 056 - 5..

**Hernández Sampieri, Roberto, y otros. 2010.** *Metodología de la investigación*. 2010. ISBN: 978-607-15-0291-9.

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

**JUnit. 2018.** JUnit. [En línea] 12 de abril de 2018. [Citado el: 21 de marzo de 2019.] <https://junit.org>.

**L. Bass, P. y Clements, R. K. 2003.** *Software Achitecture in Practice*. s.l. : Addison Wesley, 2003.

**Larman, Craig. 2003.** *UML y Patrones*. s.l. : 2da Edición, 2003.

**Lugo Robaina, B., Rodríguez Guerra, L. 2017.** *Herramienta informática para el diagnóstico inicial en el*. La Habana : s.n., 2017.

- Méndez, Eva Rodríguez. 2017.** *La descripción de documentos electrónicos a través de metadatos: una visión para la archivística desde la nueva e-administración.* s.l. : d'Arxius, 2017.
- Mugica, Mayra Mena. 2015.** *Gestión documental y organización de archivos.* La Habana: : Felix Varela, 2015.
- NISO. 2004.** *Understanding Metadata.* s.l. : NISO Press, 2004. ISBN 9-958-124-62-1.
- Norma ISO UNE 15489. 2006.** *Información y documentación: gestión de documentos.* Madrid : AENOR, 2006.
- Paradigm, Visual. 2018.** UML CASE tool for software development. UML CASE tool for software. [En línea] 2018. [Citado el: 26 de marzo de 2019.] <https://www.visual-paradigm.com/>.
- Pikovsky, Arkady, Rosenblun, Michael y Kurths, Jurgen. 2001.** *Synchronization a Universal Concept in Nonlinear Sciences.* Potsdam : Cambrige Nonlinear Science Series, 2001. ISBN-13 978-0511-07595-7.
- Pressman, Roger S. 2010.** *Ingeniería del software. Un enfoque práctico.* New York : s.n., 2010.
- PROGRAMACIÓN DESARROLLO. 2011.** Qué es un entorno de desarrollo, IDE. [En línea] 15 de febrero de 2011. [Citado el: 26 de marzo de 2019.] <http://programaciondesarrollo.es/que-es-un-entorno-de-desarrollointegrado-ide/>.
- Risso, Verónica Gauchi. 2012.** *Aproximación teórica a la relación entre los términos gestión documental, gestión de información y gestión del conocimiento.* Granada : s.n., 2012. 0210-0614.
- Roberge, Michel. 2006.** *Lo esencial de la gestión documental: sistema integrado de gestión de los documentos analógicos y de los documentos electrónicos.* s.l. : Solutions Documentaires Gestar, 2006.
- Rodríguez, T. Genebeta:dev Eclipse IDE. 2015.** [En línea] 10 de enero de 2015. [Citado el: 26 de marzo de 2018.] <http://www.genbetadev.com/herramientas/eclipse-ide>.
- Sánchez, Tamara Rodríguez. 2015.** *Metodología de desarrollo para la Actividad productiva UCI.* La Habana : s.n., 2015.
- Santiago Mellano, Cecilia Molina, and Germán Pratelli. 2014.** *Sistema para la Administración de Programas y Proyectos de Investigación coordinados por la Secretaría de Ciencia y Técnica.* Universidad Nacional de Río Cuarto : s.n., 2014.
- Scientec. 2016.** Scientec. [En línea] 17 de octubre de 2016. [Citado el: 26 de marzo de 2019.] <https://www.scientec.com.mx/axure-rp/>.
- Shariff, Munwar. 2014.** *Enterprise Content Management Implementation.* 2014. ISBN 1-904811-11-6.
- Sibat, Martinez Yanet. 2013.** *La ejecución de flujos de trabajo de aprobación.* 2013.
- Soluciones Open Source. 2008.** *Gestión Documental Open Source.* 2008. ISBN 1-880124-62-9..
- Sommerville, Ian. 2011.** *Ingeniería de Software.* México : PEARSON EDUCACIÓN, 2011.
- SQLite. 2018.** <https://www.sqlite.org/>. [En línea] 2018. [Citado el: 15 de abril de 2019.]
- UPV. 2018.** UPV. [En línea] 2018. [Citado el: 26 de marzo de 2019.] <http://personales.upv.es/rmartin/cursoJava/Java/Introduccion/PrincipalesCaracteristicas.htm>.

**Vigo Valdés, Yahima, Marrero Palomino, Antonio y Sánchez Tellez, Eddy. 2009. *Sincronización para centros de gestión de emergencias*. Habana : s.n., 2009. Vol. 3.**

## Anexos

### Anexo 1 Entrevista

La entrevista se le realizo al director del centro CIGED el MSc. Aurelio Antelo Collado.

1. ¿Porque surge la necesidad de desarrollar una aplicación de sincronización para el producto XABAL-eXcriba 3.1?
2. ¿Qué requisitos debe tener la aplicación de sincronización?
3. ¿Qué tipos de documentos debe manejar la aplicación?
4. ¿Qué tipos de usuarios pueden acceder a la aplicación?
5. ¿Cómo garantizar la seguridad de la aplicación?
6. ¿Cuáles serían las propiedades con las que debe contar la aplicación?

### Anexo 2 Descripción de los casos de uso

Tabla. 13 CU1. Autenticar Usuario

<b>Objetivo</b>	Permitir al usuario registrado acceder al sistema.	
<b>Actores</b>	Usuario	
<b>Resumen</b>	El caso de uso se inicia cuando un usuario desea acceder al sistema.	
<b>Complejidad</b>	Media	
<b>Prioridad</b>	Media	
<b>Precondiciones</b>	El sistema debe estar funcionando.	
<b>Postcondiciones</b>	El usuario quedó autenticado.	
<b>Flujo de eventos</b>		
<b>Flujo básico Autenticar Usuario</b>		
	<b>Actor</b>	<b>Sistema</b>
1.	Ejecuta la aplicación	Muestra la ventana principal de la aplicación.
2.	Selecciona el botón Configurar servidor	
3.		Ver Tabla 4 CU2 Configurar servidor
4.	Introduce el usuario y contraseña.	
5.	Presiona el botón acceder.	
6.		Valida los datos y accede.
7.		Termina caso de uso
<b>Flujos alternos</b>		
<b>5a. Cancelar operación</b>		
	<b>Actor</b>	<b>Sistema</b>
1.	Presiona el botón Cancelar	
		Accede al sistema solo puede configurar las opciones de conexión con el servidor.

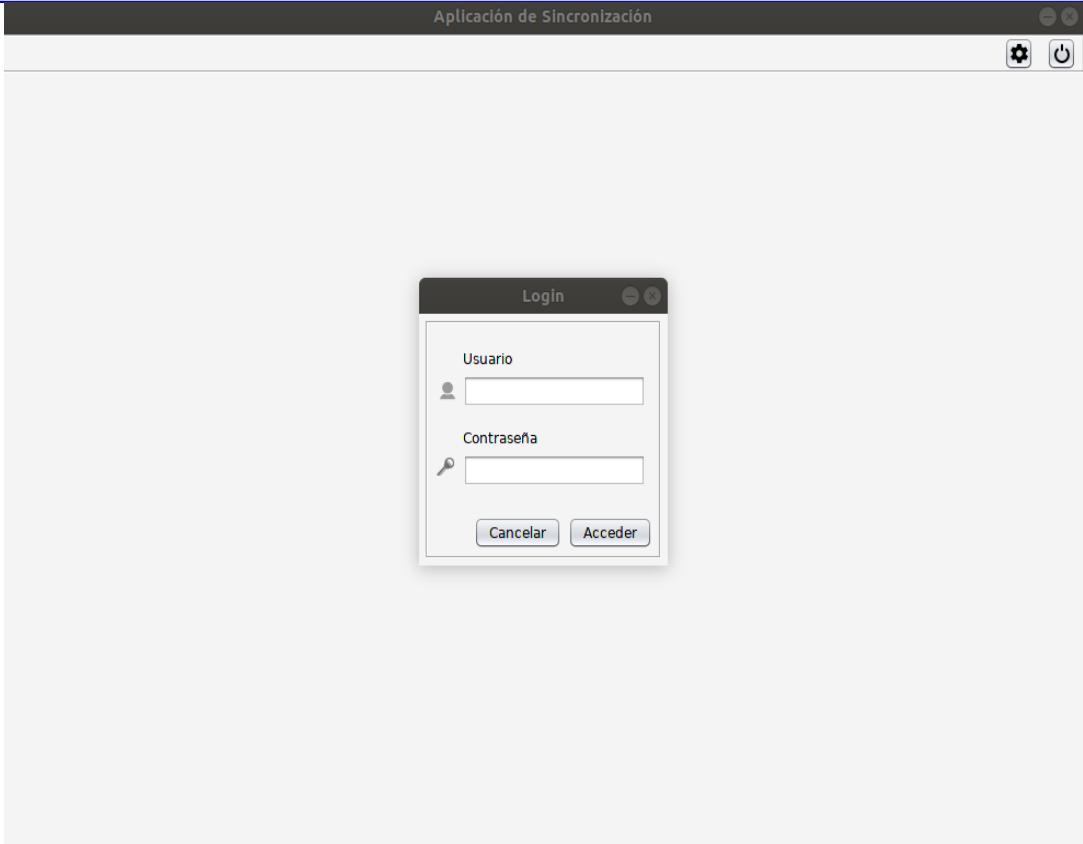
6a Existen campos incorrectos		
	Actor	Sistema
1.		Muestra el mensaje "Existen campos incorrectos".
6b Existen campos vacíos		
	Actor	Sistema
1.		Muestra el mensaje "Existen campos vacíos"
Relaciones	CU incluidos	CU3 Configurar Servidor
	CU extendidos	N/A
Prototipo elemental de interfaz gráfica de usuario		
		

Tabla. 14 CU2 Configurar servidor

<b>Objetivo</b>	Permite al Usuario configurar la conexión con el servidor.	
<b>Actores</b>	Usuario	
<b>Resumen</b>	El caso de uso inicia cuando el actor decide configurar la conexión con el servidor.	
<b>Complejidad</b>	Baja	
<b>Prioridad</b>	Alta	
<b>Precondiciones</b>	El sistema debe estar funcionando.	
<b>Postcondiciones</b>	Se configuró correctamente la conexión con el servidor.	
<b>Flujo de eventos</b>		
<b>Flujo básico Gestionar Configuración</b>		
	Actor	Sistema
1.	Selecciona la opción Configurar servidor.	



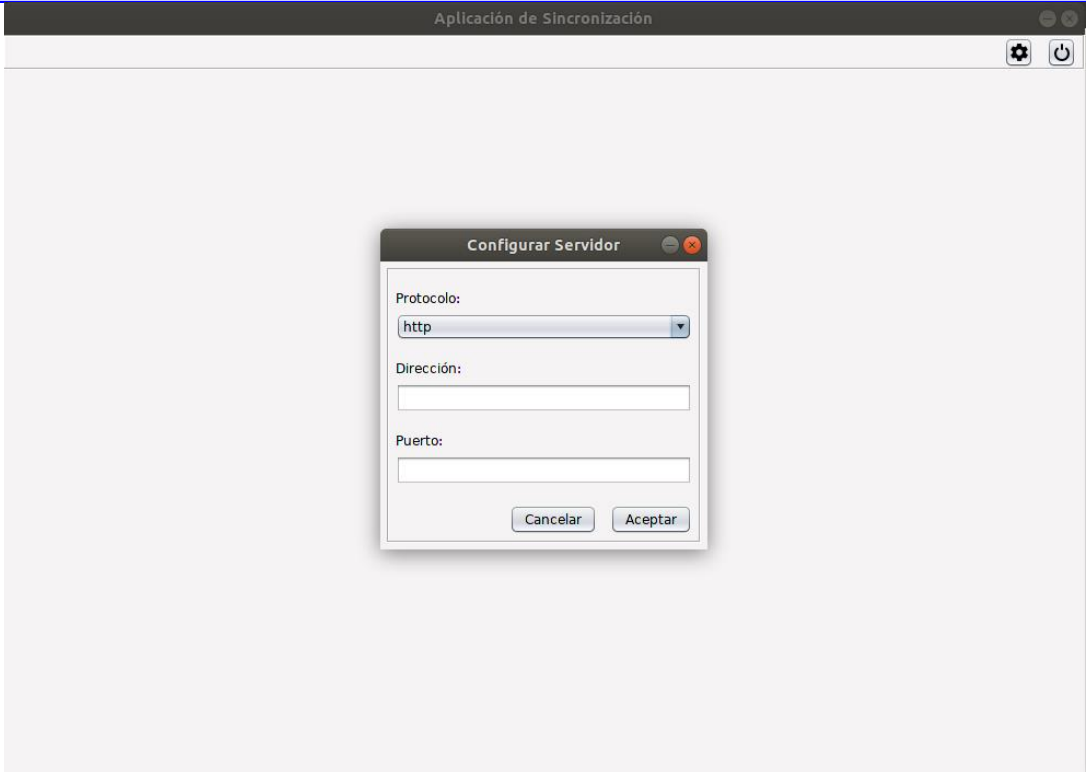
2.		Muestra una lista desplegable con los protocolos disponibles, un campo para introducir la dirección del servidor y otro para el puerto por donde se va a conectar. Permite guardar y cancelar.
3.	Selecciona el protocolo, introduce la dirección y el puerto.	
4.	Presiona el botón guardar	
5.		Valida y guarda los datos
6.		Termina caso de uso
5.		
<b>4a. Cancelar operación</b>		
	<b>Actor</b>	<b>Sistema</b>
1.	Presiona el botón cancelar	
		Elimina toda la configuración realizada y vuelve a la pantalla principal de la aplicación.
<b>5a. Existen campos vacíos</b>		
	<b>Actor</b>	<b>Sistema</b>
1.		Muestra el mensaje "Existen campos vacíos"
<b>Relaciones</b>	<b>CU incluidos</b>	N/A
	<b>CU extendidos</b>	N/A
<b>Prototipo elemental de interfaz gráfica de usuario</b>		
		

Tabla. 15 CU4. Gestionar Documentos

<b>Objetivo</b>	Permite al usuario mostrar, crear, eliminar y actualizar documentos.	
<b>Actores</b>	Usuario	
<b>Resumen</b>	El caso de uso inicia cuando el actor decide mostrar, crear, eliminar y actualizar documentos.	
<b>Complejidad</b>	Alta	
<b>Prioridad</b>	Alta	
<b>Precondiciones</b>	El usuario debe estar autenticado en el sistema.	
<b>Postcondiciones</b>	Se mostró, creó, eliminó o actualizó documentos correctamente.	
<b>Flujo de eventos</b>		
<b>Flujo básico Gestionar Documento</b>		
	<b>Actor</b>	<b>Sistema</b>
1.		Mostrar documentos Ver Sección 1 Mostrar documentos
	Selecciona opción eliminar documento	
2.		Ver Sección 2 Eliminar Documento.
3.		Termina caso de uso
<b>Sección 1: Mostrar Documentos</b>		
<b>Flujo Mostrar Documentos</b>		
	<b>Actor</b>	<b>Sistema</b>
1.		Muestra dos árboles documentales uno con los documentos que se encuentran en el eXcriba y otro con los que se han descargado.
<b>Sección 2: Eliminar Documento</b>		
<b>Flujo Eliminar Documento</b>		
	<b>Actor</b>	<b>Sistema</b>
1.	Selecciona el documento y presiona el botón eliminar.	
2.		Muestra una ventana con el mensaje " Esta seguro que desea eliminar el documento seleccionado".
3.	Presiona el botón " Sí ".	
4.		Muestra el mensaje "Documento eliminado satisfactoriamente "
<b>Flujos alternos</b>		
<b>4a Cancelar operación</b>		
	<b>Actor</b>	<b>Sistema</b>
1.	Presiona el botón "No"	
2.		Vuelve a la pantalla principal de la aplicación.
<b>Relaciones</b>	<b>CU incluidos</b>	N/A
	<b>CU extendidos</b>	N/A
<b>Prototipo elemental de interfaz gráfica de usuario</b>		

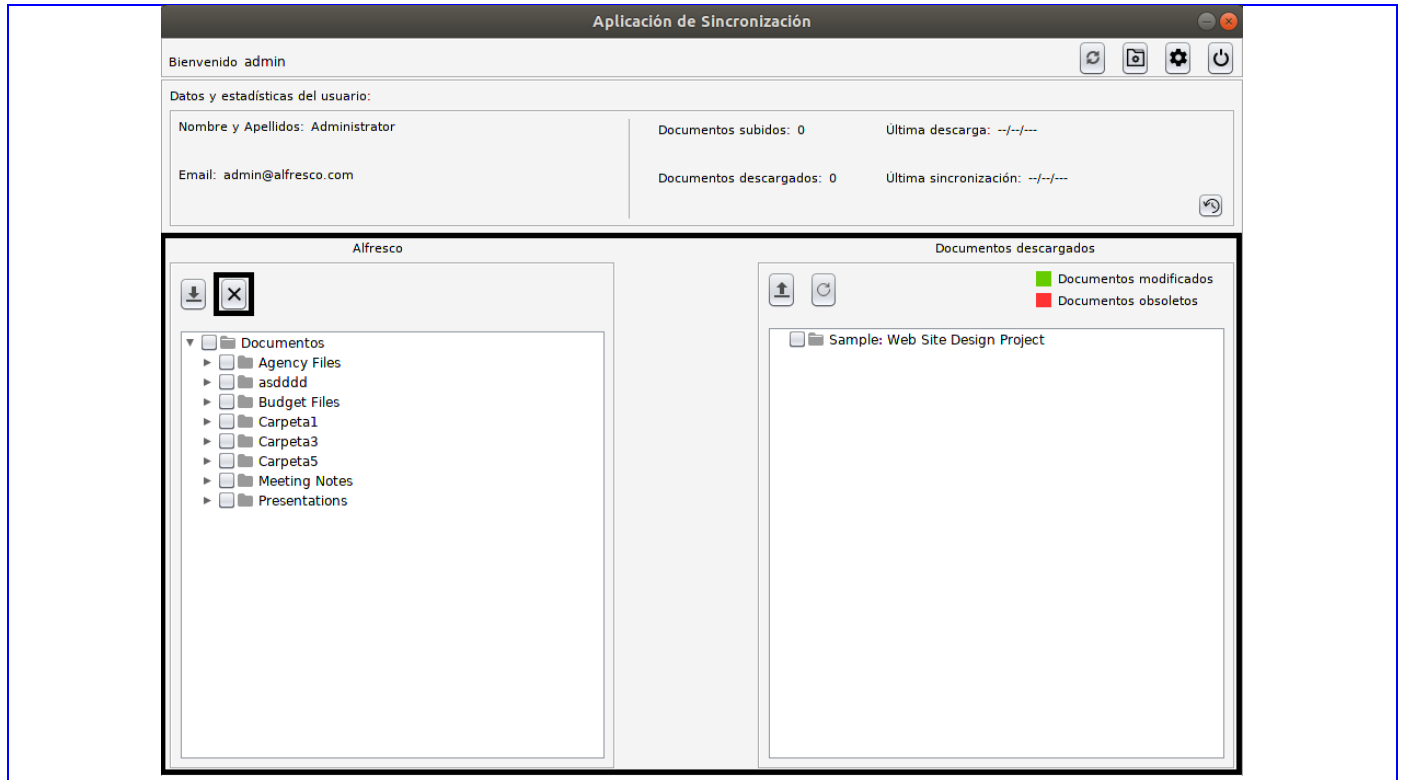


Tabla. 16 CU5 Descargar Documentos

<b>Objetivo</b>	Permitir al usuario autenticado descargar documentos	
<b>Actores</b>	Usuario	
<b>Resumen</b>	Descargar documentos o estructuras de carpetas y documentos completas.	
<b>Complejidad</b>	Alta	
<b>Prioridad</b>	Alta	
<b>Precondiciones</b>	El usuario debe estar autenticado	
<b>Postcondiciones</b>	Se descargaron correctamente el/los documentos.	
<b>Flujo de eventos</b>		
<b>Flujo básico Descargar Documentos</b>		
	<b>Actor</b>	<b>Sistema</b>
1.		Muestra la ventana principal de la aplicación
2.	Realiza las configuraciones de sincronización (Ver Tabla 5. CU3 Configurar Sincronización).	
3.	Selecciona las carpetas o documentos que desea descargar y presiona el botón descargar.	
4.		Muestra el mensaje "Documentos descargados satisfactoriamente". Además se brinda la posibilidad de abrir dichos documentos.
5.		Termina caso de uso
<b>Relaciones</b>	<b>CU incluidos</b>	N/A
	<b>CU extendidos</b>	N/A
<b>Prototipo elemental de interfaz gráfica de usuario</b>		

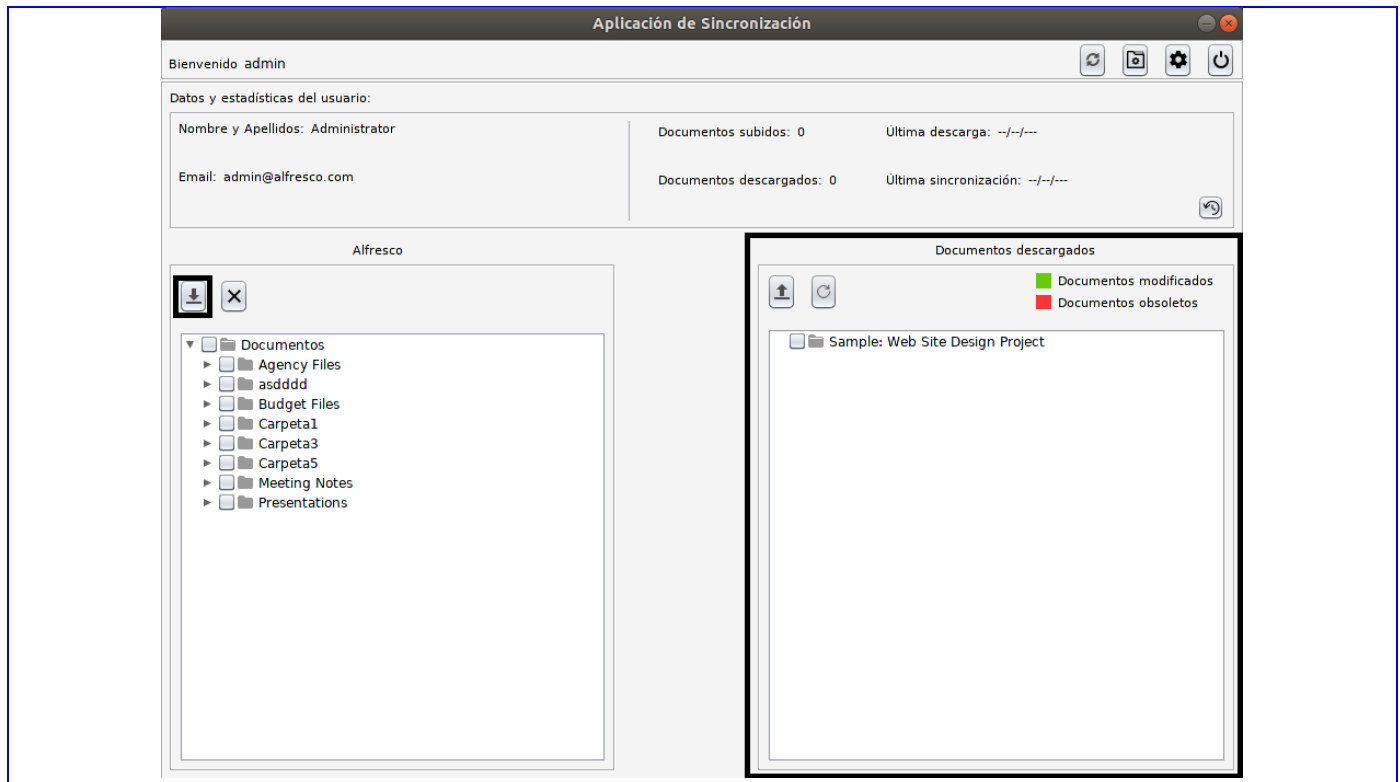


Tabla. 17 CU6. Subir Documentos

<b>Objetivo</b>	Permite al usuario subir documentos	
<b>Actores</b>	Usuario	
<b>Resumen</b>	El caso de uso inicia cuando el actor decide subir documentos.	
<b>Complejidad</b>	Media	
<b>Prioridad</b>	Media	
<b>Precondiciones</b>	El usuario debe estar autenticado	
<b>Postcondiciones</b>	Se subió correctamente el documento	
<b>Flujo de eventos</b>		
<b>Flujo básico Gestionar Configuración</b>		
	<b>Actor</b>	<b>Sistema</b>
1.		Muestra la ventana principal de la aplicación.
2.	Presiona el botón "Subir documentos"	
3.		Sube los documentos que hayan sido creados o modificados.
4.		Termina caso de uso
<b>Flujos alternos</b>		
2a. Seleccionar y subir		
	<b>Actor</b>	<b>Sistema</b>
1.	Selecciona el/los documentos que desea subir y presiona el botón "Subir documentos"	
		Sube los documentos seleccionados.

<b>Relaciones</b>	<b>CU incluidos</b>	N/A
	<b>CU extendidos</b>	N/A

**Prototipo elemental de interfaz gráfica de usuario**

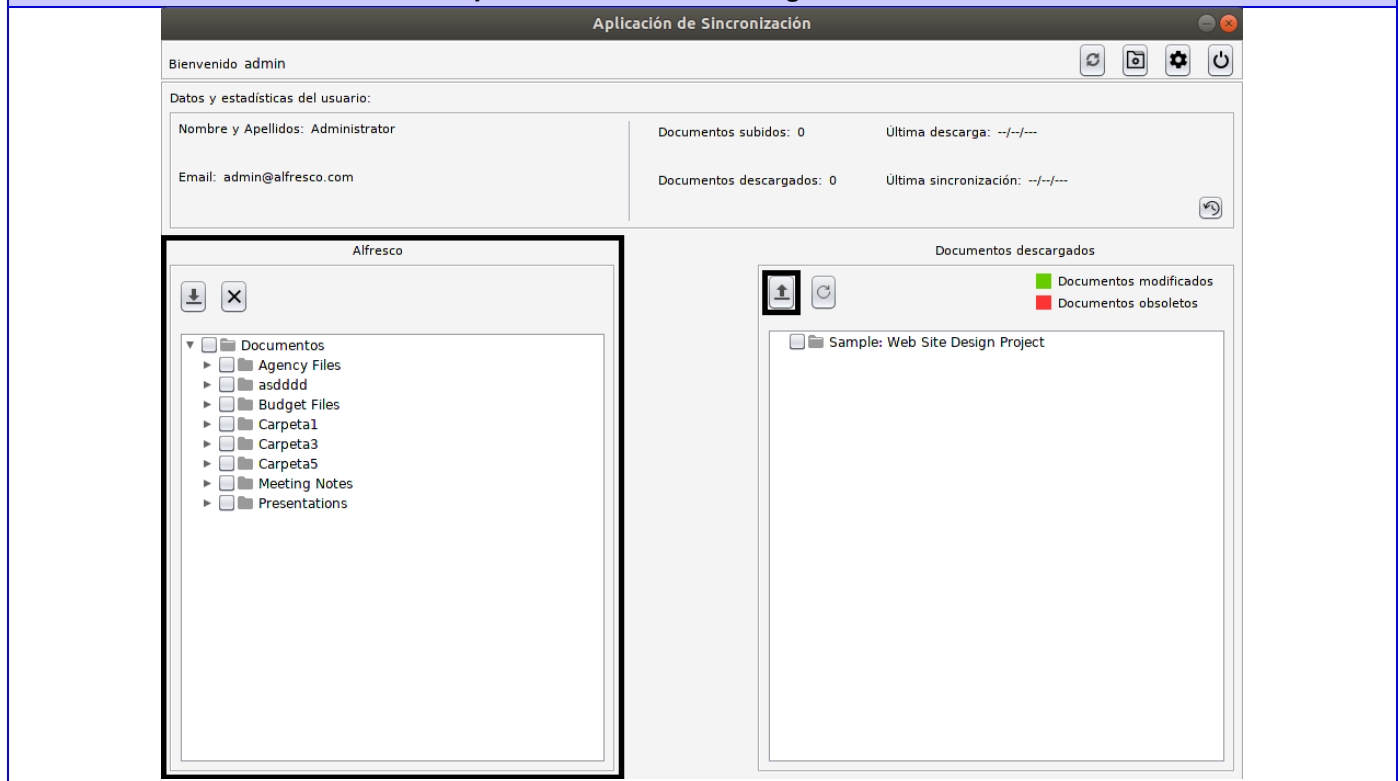


Tabla. 18 CU7 Mostrar estadísticas

<b>Objetivo</b>	Mostrar las estadísticas de las acciones realizadas por el usuario.	
<b>Actores</b>	Usuario	
<b>Resumen</b>	Permite mostrar la cantidad de documentos descargados, subidos y la fecha de la última sincronización y última descarga.	
<b>Complejidad</b>	Baja	
<b>Prioridad</b>	Baja	
<b>Precondiciones</b>	El usuario debe estar autenticado	
<b>Postcondiciones</b>	Se mostraron las estadísticas.	
<b>Flujo de eventos</b>		
<b>Flujo básico Mostrar Información</b>		
	<b>Actor</b>	<b>Sistema</b>
1.		Muestra la cantidad de documentos descargados, subidos y la fecha de la última descarga y de la última sincronización.
2.		Termina caso de uso
<b>Relaciones</b>	<b>CU incluidos</b>	N/A
	<b>CU extendidos</b>	N/A
<b>Prototipo elemental de interfaz gráfica de usuario</b>		

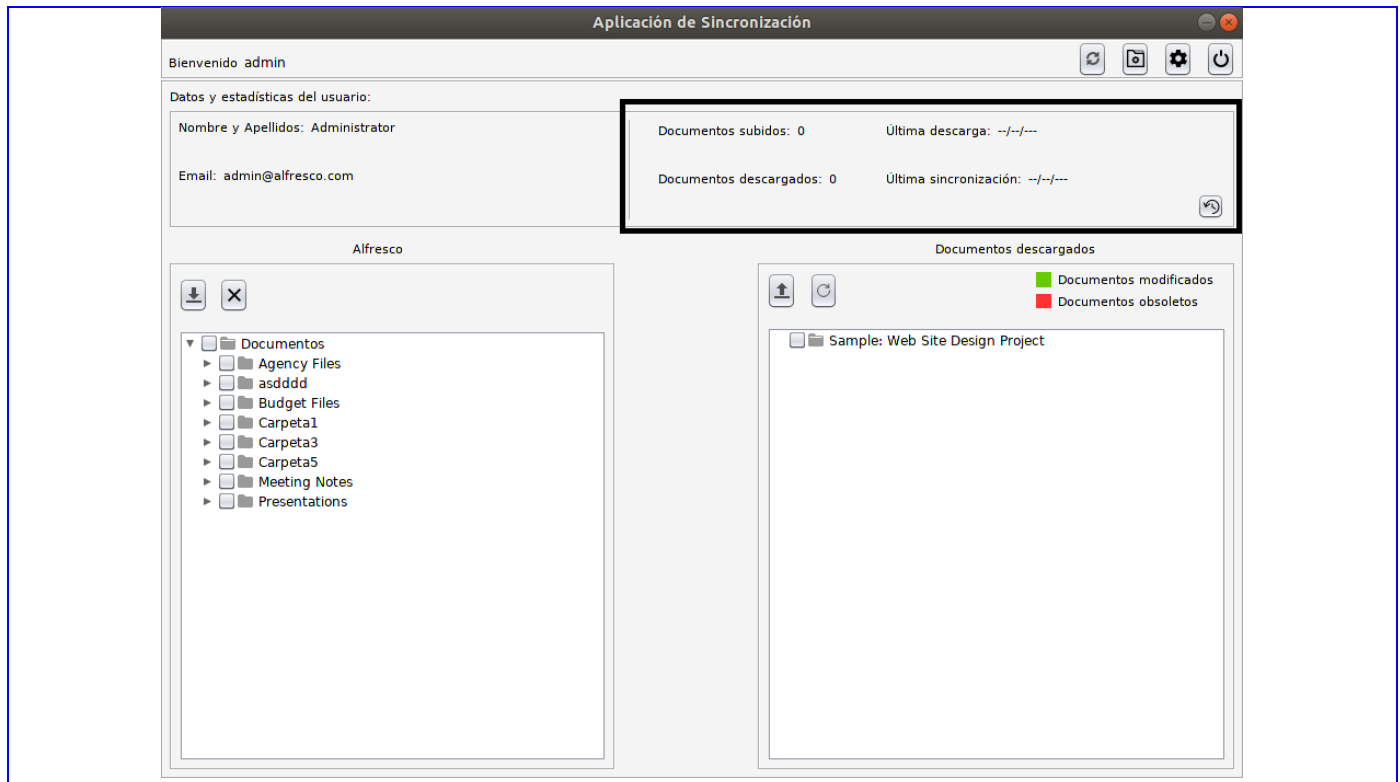
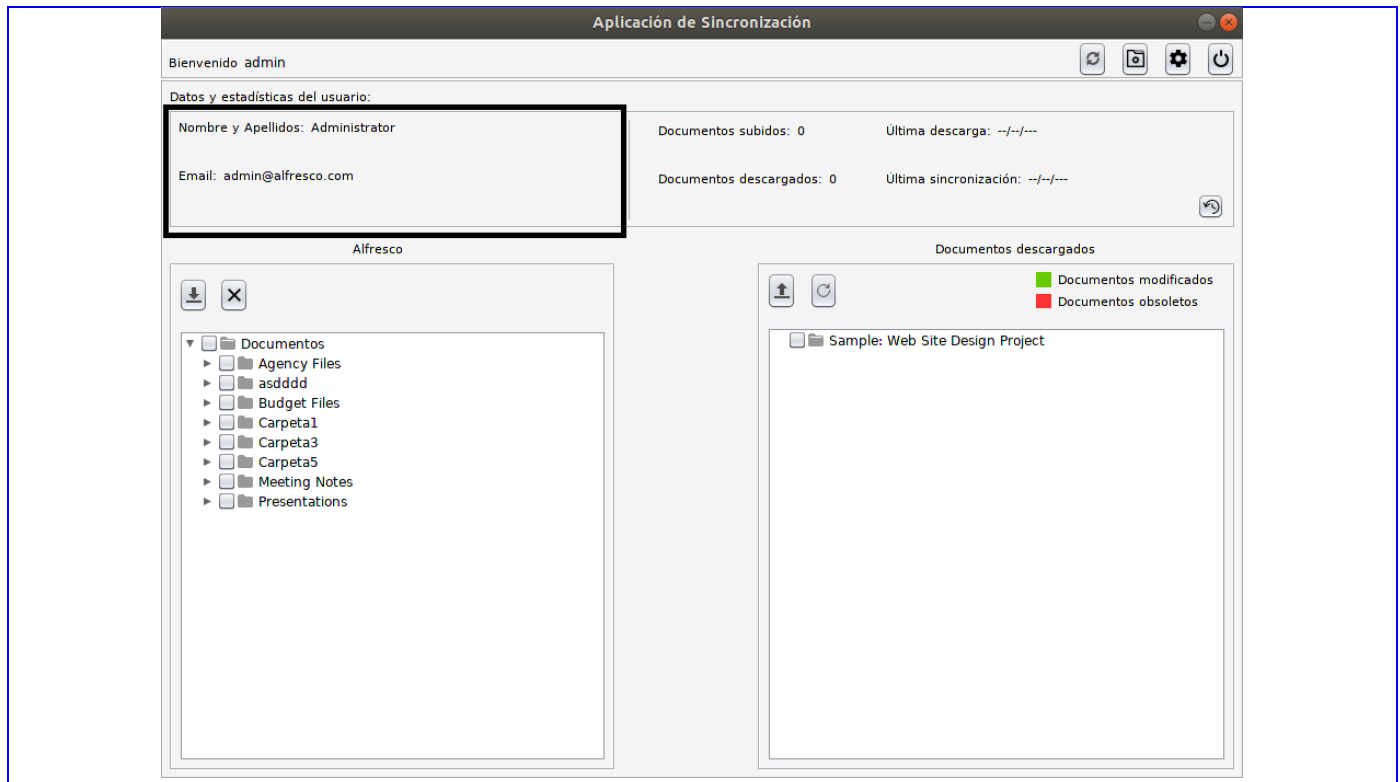


Tabla. 19 CU8 Mostrar datos del usuario

<b>Objetivo</b>	Mostrar los datos del usuario autenticado.		
<b>Actores</b>	Usuario		
<b>Resumen</b>	Muestra el nombre, apellidos y el correo del usuario		
<b>Complejidad</b>	Baja		
<b>Prioridad</b>	Baja		
<b>Precondiciones</b>	El usuario debe estar autenticado		
<b>Postcondiciones</b>	Se mostraron los datos del usuario.		
<b>Flujo de eventos</b>			
<b>Flujo básico Mostrar Información</b>			
	<b>Actor</b>		<b>Sistema</b>
1.			Muestra el usuario, nombre y apellidos del usuario y el correo
2.			Termina caso de uso
<b>Relaciones</b>	<b>CU incluidos</b>	N/A	
	<b>CU extendidos</b>	N/A	
<b>Prototipo elemental de interfaz gráfica de usuario</b>			



### Anexo 3 Diseño caso de prueba

Tabla. 20 Diseño caso de prueba del caso de uso configurar servidor

Escenarios	Descripción	Protocolo	Dirección	Puerto	Respuesta del sistema	Flujo central
EC 1.1 Configurar servidor correctamente	Permite al usuario configurar la conexión de la aplicación a través del protocolo, dirección y puerto	V	V	V	Guarda los datos introducidos para la posterior autenticación	1- Presionar el botón "Configurar servidor".
		http/https	127.0.0.1	8080		2- Introducir los datos y presionar el botón "Aceptar".
		V	I	I		

EC 1.2 Configurar servidor con datos incorrectos	El usuario introduce datos incorrectos en el formulario de configurar servidor.	http/https	excriba*!	afdf	Muestra un mensaje de error: "Existen datos incorrectos"	1-Presionar el botón "Configurar servidor". 2- Introducir los datos incorrectos y presionar el botón "Aceptar".
		V	V	I		
		http/https	127.0.0.1	65as		
		V	I	V		
		http/https	127!.5.5.1	8080		
EC 1.3 Configurar servidor con campos vacíos	El usuario deja campos vacíos en el formulario de configurar servidor.	V	I	I	Muestra el mensaje: "Existen campos vacíos"	1-1-Presionar el botón "Configurar servidor". 2-Dejar campos vacíos y presionar el botón "Aceptar"
		http/https				
		V	V	I		
		http/https	excriba			
		V	I	V		
		http/https		4045		

Tabla. 21 Descripción de las variables de entrada configurar servidor

No	Nombre de campo	Clasificación	Valor Nulo	Descripción
1	Protocolo	Campo de selección	No	Permite al usuario seleccionar el protocolo(http/https)
2	Dirección	Campo de texto	No	No admite caracteres especiales
3	Puerto	Campo de texto	No	Solo admite números

Tabla. 22 Diseño caso de prueba del caso de uso gestionar documento SC1 mostrar documentos



Escenarios	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Mostrar documentos a partir de configurar la sincronización correctamente	Permite al usuario visualizar los documentos en Alfresco y los documentos descargados	Muestra los arboles documentales correspondientes a alfresco y a la carpeta local	1-Configurar sincronización seleccionando área y destino y presiona el botón "aceptar"
EC 1.2 El usuario no tiene configurado el destino	No muestra árbol documental local	Muestra el árbol documental correspondiente a alfresco	1- Presionar el botón "Configurar sincronización". 2- Selecciona el área deseada

Tabla. 23 Diseño caso de prueba del caso de uso gestionar documento SC2 Eliminar documentos

Escenarios	Descripción	Respuesta del sistema	Flujo central
EC 2.1 Eliminar documento	El usuario elimina el documento correctamente	El sistema muestra un mensaje para confirmar la eliminación "¿Esta seguro que desea eliminar el documento seleccionado?"	1-Selecciona el documento que desea eliminar y presiona el botón "Eliminar"

EC 2.2 Cancelar operación	El usuario cancela la operación de eliminar documento	El sistema cancela la operación y regresa a la vista anterior	<p>1- Selecciona el documento que desea eliminar y presiona el botón "Eliminar"</p> <p>2- Presiona el botón "Cancelar"</p>
---------------------------	-------------------------------------------------------	---------------------------------------------------------------	----------------------------------------------------------------------------------------------------------------------------

Tabla. 24 Diseño caso de prueba del caso de uso descargar documentos

Escenarios	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Descargar documentos satisfactoriamente	El usuario descarga los documentos correctamente	El sistema muestra un mensaje para confirmar la descarga "¿Esta seguro que desea descargar el/los documentos seleccionado?"	<p>1-Selecciona el/los documentos que desea descargar y presiona el botón "Descargar documentos"</p> <p>2-Presiona el botón "Sí"</p>
EC 1.2 Cancelar operación	El usuario cancela la operación de descargar documentos	El sistema cancela la operación y regresa a la vista anterior	<p>1- Selecciona el/los documentos que desea descargar y presiona el botón "Descargar documentos"</p> <p>2- Presiona el botón "No"</p>

Tabla. 25 Diseño caso de prueba del caso de uso subir documentos

Escenarios	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Subir documentos satisfactoriamente	El usuario sube los documentos correctamente	El sistema muestra el mensaje "¿Esta seguro que desea subir el/los documentos seleccionado?"	1-Selecciona el/los documentos que desea subir y presiona el botón "Subir documentos" 2-Presiona el botón "Sí"
EC 1.2 Cancelar operación	El usuario cancela la operación de subir documentos	El sistema cancela la operación y regresa a la vista anterior	1- Selecciona el/los documentos que desea subir y presiona el botón "Subir documentos" 2- Presiona el botón "No"

Tabla. 26 Diseño caso de prueba del caso de uso sincronizar documentos

Escenarios	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Sincronizar documentos satisfactoriamente	El usuario sincroniza los documentos correctamente	El sistema muestra una ventana con los documentos que el usuario descargó y que	1-Presiona el botón "Sincronizar documentos" 2-Selecciona los documentos modificados que desea actualizar y presiona el botón "Sincronizar"

		han sido modificados en el repositorio	
EC 1.2 Cancelar operación	El usuario cancela la operación de sincronizar documentos	El sistema cancela la operación y regresa a la vista anterior	1- Presiona el botón "Sincronizar documentos" 2- Presiona el botón "Cancelar"

Tabla. 27 Diseño caso de prueba del caso de uso mostrar datos del usuario

Escenarios	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Mostrar datos del usuario	Permite al usuario visualizar sus datos (nombre, apellidos y correo)	El sistema muestra la ventana principal de la aplicación con los datos del usuario.	1-El usuario se autentica en la aplicación

Tabla. 28 Diseño caso de prueba del caso de uso mostrar estadísticas

Escenarios	Descripción	Respuesta del sistema	Flujo central
EC 1.1 Mostrar estadísticas	Permite al usuario visualizar las estadísticas generadas a partir de sus acciones (documentos descargados,	El sistema muestra la ventana principal de la aplicación con las estadísticas del usuario.	1-El usuario se autentica en la aplicación

	documentos subidos y la fecha de la última sincronización y de la última descarga)		
--	------------------------------------------------------------------------------------	--	--

#### Anexo 4 Acta de aceptación

**ACTA DE ACEPTACIÓN**

En cumplimiento del **Convenio de colaboración con el centro CIGED** y en función de la ejecución del proyecto de tesis: **Aplicación de sincronización de documentos del producto XABAL-eXcriba basado en Alfresco 5.2**, se hace entrega de los productos que se relacionan a continuación:

- *Especificación de Requisitos de Software*

La Parte Cliente, luego de haber revisado los productos de trabajo determina su aceptación.

<p><b>Entrega Tesista</b></p> <hr/> <p><b>Nombre y apellidos:</b> Alejandro González Rodríguez</p> <hr/> <p><b>Cargo:</b> Desarrollador</p>	<p><b>Recibe Centro CIGED</b></p> <hr/> <p><b>Nombre y apellidos:</b> Aurelio Antelo Collado</p> <hr/> <p><b>Cargo:</b> Director de CIGED</p>
---------------------------------------------------------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------







**Fecha:** 5/06/2019