Universidad de las Ciencias Informáticas

Facultad 1

Edición y lectura de documentos de formato abierto desde el cliente web del Gestor de Documentos Administrativos eXcriba

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

Autor: Miguel Enrique Vives Enríquez.

Tutores:

Ing. Misael Fonseca Mata.

Ing. Julio Antonio Zamora Rosabal.

La Habana, 2013

Declaración de autoría

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

Para que así conste firmo la presente a los ____días del mes de____ del año____.

_____

Firma del autor

Miguel Enrique Vives Enríquez.

_____

Firma del tutor

Ing. Misael Fonseca Mata.

_____

Firma del tutor

Ing. Julio Antonio Zamora Rosabal.

Agradecimientos

Agradezco a mis padres por brindarme en todo momento su apoyo e inmenso amor incondicional, por guiarme siempre por el buen camino, por ser ejemplo de sacrificio y empeño que se tiene que tener para alcanzar las cosas que se quieren en la vida, y por darme de vez en cuando un buen regaño.

Agradezco a mi familia por siempre preocuparse por mí, y estar presente en todo momento.

Agradezco a mis amigos por ayudarme y por compartir todos estos momentos inolvidables que hemos pasado juntos.

Agradezco a mis tutores, por siempre preocuparse, darme apoyo y ayuda en la realización de este trabajo.

Resumen

En la Universidad de las Ciencias Informáticas (UCI), específicamente en el departamento de Gestión Documental se desarrolla la segunda versión del Gestor de Documentos Administrativos (GDA) eXcriba, sistema que tiene como objetivo automatizar los procesos documentales que se ejecutan dentro de cualquier entidad. Este sistema brinda la posibilidad de trabajar con documentos de formato abierto almacenados en un único repositorio. Sin embargo, la lectura y edición de los documentos, se debe realizar a través de un complejo, dependiente y lento proceso, por lo que es necesario, facilitar y agilizar el mismo para eliminar así los diferentes factores que la retrasan y complejizan. Para dar solución a este problema se va a desarrollar un módulo que permita al GDA eXcriba editar y leer documentos desde el cliente web.

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

Con el desarrollo del módulo los usuarios pueden leer y editar documentos de formato abierto desde el cliente web del eXcriba, agilizando dicho proceso y mejorando la experiencia del usuario.

Palabras clave: documento de formato abierto, edición y lectura, gestión documental.

Abstract

At the University of Information Sciences, in the Department of Management and Archives Documentary develops the second version of Manager Administrative Document's eXcriba, a system that aims to automate document processes running within any entity. This system provides the opportunity to work with open format documents stored in a single repository. However, given the need for reading and editing such documents must be made through a cumbersome and slow process dependent, so it is necessary to facilitate and streamline the process for different factors, thereby eliminating delay and complicate this process. To solve this problem is to develop a module to allow the eXcriba edit and read documents from the web client. For its development, a study of theoretical aspects of editing and reading documents open format on the web, as well as document management systems. Subsequently performs the design of the solution as a starting point to implement its functionalities. Finally, check if the module complies with the required quality and customer needs by applying black-box testing and white box. With the development of module users can read and edit documents within the open format eXcriba, streamlining the process and improving the user experience.

Keywords: document management, document open format, document management system.

Tabla de contenidos

Índice de figuras

Índice de tablas

Introducción

El ser humano es un ser sociable por naturaleza, por lo que está obligado a establecer lazos con distintas personas por diversas razones. La información no puede quedar fuera de estos vínculos entre seres humanos, de ahí que haya evolucionado y formado la llamada Sociedad de la Información (Valdés, 2008); que no es más que el enlace entre individuos, organismos e instituciones con una perspectiva o área en común, los cuales se han relacionado con el fin de intercambiar información.

Es en esta sociedad de la información que pasa a jugar un papel determinante el documento electrónico como soporte de información, que no es más que un documento que está en forma electrónica porque se ha creado mediante un programa informático de aplicación o bien porque se ha digitalizado, (MoReq, 2001) a través de algún tipo de dispositivo electrónico o magnético, y su contenido puede ser también leído, interpretado, o reproducido por un programa informático.

Producto al flujo continuo de información, generado por los documentos electrónicos, surge la necesidad de gestionar dicho flujo de una forma adecuada, es aquí donde pasa a jugar un rol determinante la gestión documental.

En la ISO 15489[1] se define la gestión documental como el área de gestión responsable de un control eficaz y sistemático de la creación, la recepción, el mantenimiento, el uso y la disposición de documentos de archivo, incluidos los procesos para incorporar y mantener en forma de documentos la información y prueba de las actividades y operaciones de la organización.

Para la presente investigación la gestión documental es el modo en que se procesa la información generada en forma de documentos por las diferentes organizaciones, a través de un conjunto de métodos, procesos, normas, técnicas y prácticas, a lo largo del ciclo de vida de dichos documentos, que comprende: creación/captura, gestión/revisión, publicación y almacenamiento. Es una vía para que los usuarios accedan de forma oportuna a la información, organicen grandes volúmenes, mantengan sus flujos adecuados en la organización y soporten su integridad y seguridad.

[1] Norma ISO 15489. Secretaría del CTN50, calle Santa Engracia, 17, 3.º, 28010 Madrid.

El crecimiento exponencial de los volúmenes de información ha colocado la gestión documental en un lugar privilegiado para garantizar el uso adecuado y oportuno de la información. La gestión documental abarca desde la identificación del documento hasta su archivo, búsqueda y recuperación.

Los documentos que generan estos volúmenes de información están relacionados con varios tipos de formatos, estos se pueden agrupar en formato abierto y propietario. Dentro del formato propietario se encuentran agrupados diferentes tipos de documentos como: DOC y PPT, generados por la herramienta ofimática Microsoft Office. Y como parte del formato abierto se encuentran los siguientes tipos de documentos: PDF (Portable Document Format) y ODF (Open Document Format), estos son llamados documentos de formato abierto, en estos se centrará la presente investigación.

Las herramientas que trabajan con estos formatos entran dentro de la clasificación de herramientas ofimáticas que no son más que aplicaciones o programas que sirven para crear, modificar, visualizar, organizar, imprimir archivos y documentos, como resultado de los procesos que se llevan a cabo en oficinas, instituciones y universidades. Estas herramientas, pueden adquirirse por separado o en un paquete, llamado paquete ofimático.

Debido al auge de Internet y de las tecnologías web, han surgido herramientas ofimáticas en línea (*online*), que permiten hacer todas las tareas comunes, pero de manera *online*. Estas nuevas herramientas poseen ciertas ventajas sobre los paquetes ofimáticos convencionales, como son: la movilidad total en el acceso a la información de trabajo, ahorro económico, pues no es necesario adquirir ningún paquete o herramienta ofimática; además, no es necesaria la instalación de *software* adicional en la computadora.

En la Universidad de las Ciencias Informáticas se desarrolla el Gestor de Documentos Administrativos eXcriba, el mismo está implementado con una arquitectura cliente-servidor[2] y usa como núcleo el Gestor de Contenidos Empresariales Alfresco (ECM por sus siglas en inglés), y tiene como objetivo automatizar los procesos documentales que se ejecutan dentro de cualquier entidad, desde la elaboración de un documento en su fase de inicio hasta su conservación o expurgo en el Archivo de Gestión.

[2] Es un modelo de aplicación distribuida en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores, y los demandantes, llamados clientes. Un cliente realiza peticiones a otro programa, el servidor, quien le da respuesta.

Una vez incorporado los documentos al repositorio documental se aplica un flujo de trabajo donde los documentos deben sufrir cambios en su contenido o simplemente ser visualizado por el usuario. Esto requiere que el usuario tenga acceso a los documentos que se encuentran almacenados en el repositorio documental a través del Gestor de Documentos Administrativos eXcriba. Una vez garantizado el acceso, si el usuario desea realizar cambios en los documentos terminados tiene que descargarlos hacia su computadora y una vez terminada la edición de los documentos enviarlos desde su computadora al eXcriba, para que a través del mismo se actualicen dichos documentos en el repositorio documental. Si el usuario desea leer el contenido almacenado en los documentos tiene que descargarlos desde el cliente web del eXcriba hacia su computadora. Tanto si se desea leer o editar los documentos, una vez que son descargados los mismos, se requiere tener instalado en la computadora del usuario determinadas herramientas ofimáticas como son el caso de OpenOffice o Adobe Reader para interpretar y/o modificar el documento, que puede encontrarse en formatos tales como: ODF o PDF.

Todo este proceso de edición y lectura de los documentos almacenados en el repositorio documental, es bastante complejo, dependiente y lento, es precisamente el tiempo de realización de este proceso, un factor importante a la hora de analizar su complejidad y a su vez aceptación por parte del usuario. Pero, el factor que más influye sobre este proceso de edición y lectura de los documentos es el siguiente:

- En ocasiones, las dependencias de *software* necesarias para la edición y lectura de los documentos de formato abierto no están instaladas en las computadoras de los usuarios.

Estos elementos mencionados demoran el proceso de edición o lectura de los documentos desde el cliente web del Gestor de Documentos Administrativos eXcriba. Por esta razón surge la necesidad de agilizar en gran medida dicho proceso, mediante la edición o lectura de los documentos *online*, es decir, directamente desde el cliente web del Gestor de Documentos Administrativos eXcriba, eliminando así los diferentes factores que retrasan y complejizan dicho proceso.

A raíz de esta problemática se formula el siguiente **problema de investigación**: ¿cómo facilitar la edición y lectura de documentos de formato abierto desde el cliente web del Gestor de Documentos Administrativos eXcriba? Lo que conlleva a plantear el siguiente **objeto de estudio**: edición y lectura de documentos de formato abierto en la web, y a definir como **campo de acción**: edición y lectura de documentos de formato abierto en los Gestores de Contenidos Empresariales.

Como resultado de lo anteriormente planteado se propone como **objetivo general**: desarrollar un módulo para el cliente web del Gestor de Documentos Administrativos eXcriba que permita la edición y lectura de documentos de formato abierto desde el cliente web mediante el uso de tecnologías libres para la web. Para lograr el cumplimiento de dicho objetivo se definen los siguientes **objetivos específicos**:

- Fundamentar los antecedentes y aspectos teóricos relacionados con la edición y lectura de documentos de formato abierto en los Gestores de Contenidos Empresariales.
- Analizar las herramientas, tecnologías, lenguajes de programación y metodologías para el desarrollo de la solución.
- Diseñar el módulo para la edición y lectura de documentos de formato abierto desde el cliente web del eXcriba.
- Implementar el módulo para el Gestor de Documentos Administrativos eXcriba que permita la edición y lectura de documentos de formato abierto desde el cliente web.
- Validar el correcto funcionamiento del módulo mediante pruebas de caja blanca y caja negra.

Tareas de investigación

- Análisis bibliográfico de los antecedentes y aspectos teóricos relacionados con la edición y lectura de documentos de formato abierto en el cliente web del Gestor de Documentos Administrativos eXcriba.
- Selección de las herramientas, tecnologías, lenguajes de programación y metodologías para el desarrollo de la solución.
- Identificación de los requisitos funcionales y no funcionales de la solución.
- Elaboración del diseño del módulo para la edición y lectura de documentos de formato abierto desde el cliente web del eXcriba.
- Implementación de las funcionalidades identificadas.
- Selección de las técnicas de caja negra y caja blanca para realizar la validación.

- Validación del correcto funcionamiento de la solución.

Posibles resultados

Se espera que el módulo implementado permita al Gestor de Documentos Administrativos eXcriba editar y leer documentos de formato abierto desde el cliente web sin la necesidad de depender de que determinados *software* estén instalados en la computadora del usuario, como son OpenOffice, Microsoft Office y Adobe Reader, lo que contribuiría a facilitar y agilizar el proceso de edición o lectura de documentos de formato abierto desde el cliente web del eXcriba, a través de los siguientes factores:

Usabilidad: el módulo facilitará el proceso de edición y/o lectura de los documentos, debido a que ya no habrá necesidad de descargar o subir el documento hacia el servidor en todo este proceso de edición y lectura de documentos desde el cliente web del eXcriba.

Tiempo: el módulo permitirá un ahorro considerable de tiempo dentro del proceso de edición y/o lectura de los documentos, porque a través de la edición y lectura de documentos de formato abierto en el cliente web del eXcriba, se agilizará en gran medida dicho proceso.

Requerimientos: no habrá necesidad de tener instalado en las computadoras clientes, dependencias de *software* para procesar los documentos que se quieren leer y/o modificar.

Recursos: las computadoras clientes, no deberán contar con grandes prestaciones, se ahorrará fundamentalmente en espacio de disco duro.

A lo largo del trabajo se utilizan varios métodos científicos, basándose en las especificaciones identificadas anteriormente. Entre los métodos teóricos se utiliza el histórico-lógico, para determinar los enfoques que existen en cuanto a la edición y lectura de documentos de formato abierto en la web, así como su evolución y desarrollo hasta la actualidad. El análisis-sintético para procesar la información obtenida, que posibilitará una mejor comprensión de la edición y lectura de documentos de formato abierto en la web y su posible solución. El inductivo-deductivo para lograr la constatación del problema identificado en el presente trabajo, así como para la interpretación de la información obtenida en todo el proceso de investigación, a través de la descripción del resultado de las observaciones realizadas, que permitirá definir conceptos, teorías, y a su vez extraer consecuencias lógicas de estos enunciados.

La **justificación de la investigación** se puede describir de la siguiente forma: con la implementación de un módulo para el Gestor de Documentos Administrativos eXcriba se podrá editar y leer documentos de formato abierto desde el cliente web mediante el uso de tecnologías para la web. Esto permitirá agilizar y facilitar el proceso de edición y lectura de documentos de formato abierto desde el eXcriba y mejorar la experiencia del usuario, influyendo en factores como: usabilidad, tiempo, requerimientos y recursos.

La presente investigación ha sido estructurada de la siguiente forma, para lograr un mejor entendimiento tanto de la problemática planteada, como de la solución que se propone. Por esta razón consta de: introducción, cuatro capítulos, conclusiones, referencias bibliográficas, bibliografía, recomendaciones y anexos.

Capítulo 1:"Fundamentación teórica", contiene los principales conceptos e ideas que se tratan a lo largo del proceso de creación de un módulo que permita al Gestor de Documentos Administrativos eXcriba editar y leer documentos de formato abierto desde el cliente web. Se hace una propuesta de la herramienta CASE a utilizar, se describe la metodología seleccionada, los lenguajes de programación, el marco de trabajo, y las tecnologías.

Capítulo 2:"Procesos de negocio", se explica el flujo de todos los procesos involucrados, a través de la definición del modelo de dominio, el levantamiento de los requisitos funcionales y no funcionales del sistema, la definición de los actores y casos de usos del sistema, la interacción que existe entre ellos, así como la solución propuesta para el módulo que se desea diseñar.

Capítulo 3:"Diseño del subsistema", se exhibe a través de un conjunto de artefactos la solución al problema planteado, dentro de ellos son básicos, el diagrama de secuencia y el diagrama de clases del diseño.

Capítulo 4:"Implementación y pruebas del módulo propuesto", en este capítulo se especifica la implementación del sistema, y las pruebas que se usarán en el mismo. La ordenación en clases y componentes que garanticen su operatividad, las limitaciones y pruebas realizadas al sistema a lo largo del ciclo de vida del producto.

Capítulo 1 “Fundamentación Teórica”

1.1 Introducción

En este capítulo se analizan los principales conceptos e ideas que servirán de base para el proceso de creación de un módulo que permita al Gestor de Documentos Administrativos eXcriba editar y leer documentos de formato abierto desde el cliente web. Además, se hace una propuesta de la herramienta CASE a utilizar, se describe la metodología seleccionada, los lenguajes de programación, el marco de trabajo y las tecnologías.

1.2 Ofimática

La Ofimática, de los términos oficina e informática, hace alusión al conjunto de técnicas, aplicaciones y herramientas informáticas que se utilizan en los procesos comunes de las oficinas, para optimizar, automatizar y mejorar los procedimientos o tareas relacionadas. Es posible debido a una combinación entre *hardware* y *software* que permite crear, manipular, almacenar y enviar digitalmente la información que se necesita en una oficina. (Sáez, 1990)

Debido a las demandas de los usuarios de obtener la información al instante, se requiere establecer mecanismos automatizados, y es aquí donde la informática comienza a desempeñar un papel principal; a partir de la evolución que sufren las computadoras en sus dos dimensiones (*hardware* y *software*) y la evolución de internet y sus tecnologías. Estos mecanismos automatizados ofrecen la oportunidad de utilizar herramientas de última generación en las oficinas, para facilitar el trabajo de las personas y que a su vez el mismo sea realizado con mayor eficiencia.

Un elemento importante es que los medios donde se almacenará toda la información generada de la actividad en las oficinas tendrán un vínculo inherente a los procesadores de texto, los que desempeñan un papel fundamental en el procesamiento de toda esta información. La mayoría de estas actividades de oficina han sido automatizadas de forma digital, hasta llevar a un nivel superior el trabajo en las oficinas.

En las oficinas, es necesario destacar la información como un elemento clave, así como, la forma en que ha evolucionado su administración, con el objetivo de hacer más eficiente las diferentes actividades que

se realizan en un ámbito donde cada día se hacen necesarias las formas de obtener los resultados en el menor tiempo posible.

Las aplicaciones que se incluyen en la ofimática tienen un amplio ámbito de utilización, por tanto, estarán siempre orientadas a personal no especializado en informática, abarcando un amplio abanico de funciones típicas de la gestión de una oficina.

La siguiente figura recoge los principales conceptos de la ofimática, haciendo referencia a la historia de la ofimática y mostrando la relación que existe entre los diferentes conceptos.

Ver anexo A.1 Figura 1, de la versión extendida de este documento.

1.3 Herramientas Ofimáticas

Entre estas aplicaciones, caracterizadas por tener un mayor uso en los procesos comunes de las oficinas, se encuentran las herramientas ofimáticas; aplicaciones o programas que ofrecen la posibilidad de gestionar en su totalidad los archivos y documentos. Estas pueden adquirirse por separado o en un paquete ofimático.

Dentro de este paquete estándar de herramientas ofimáticas se encuentran, procesadores de texto, procesadores de hojas electrónicas, gestores de presentaciones y otras especializadas en diferentes tipos de documentos y funciones, muy necesarias para procesar y gestionar toda la información que se genera en las oficinas, lo que facilita y agiliza el trabajo en las mismas.

Cualquier actividad que pueda hacerse manualmente en una oficina puede ser automatizada o ayudada por herramientas ofimáticas, dígase: dictado, mecanografía, archivado, fax, microfilmado, gestión de archivos y documentos.

En la figura que se muestra a continuación, se detallan los diferentes tipos de aplicaciones informáticas, que juntas conforman los llamados paquetes ofimáticos.

Ver anexo A.1 Figura 2, de la versión extendida de este documento.

Existen algunos requerimientos que deben cumplir las herramientas ofimáticas, para una mejor usabilidad y aceptación por parte del usuario final:

- Fácil manejo.
- Interfaz de usuario sencilla y personalizable.
- Necesidad de formación mínima.
- Compatibilidad con los productos que ya se poseen.
- Interoperabilidad con otras aplicaciones.
- Facilidad para las comunicaciones con otros entornos operativos.
- Seguridad de los datos.
- Soporte de distintas plataformas físicas y sistemas operativos.
- Soporte de los dispositivos requeridos.

1.4 Documentos ofimáticos

Los archivos y documentos, resultado del trabajo con las herramientas ofimáticas, son los llamados documentos ofimáticos, los cuales cambian, en dependencia de su uso, en: documentos de texto, hojas de cálculo, presentaciones o documentos portables. Estos documentos están relacionados con varios tipos de formatos, que se pueden agrupar en formato abierto y propietario. Dentro del formato propietario se encuentran agrupados diferentes tipos de documentos como: DOC y PPT, generados por la herramienta ofimática Microsoft Office. Y como parte del formato abierto se encuentran agrupados los siguientes tipos: PDF (Documento de Formato Portable) y ODF (Documento de Formato Abierto), estos son llamados documentos de formato abierto. La mayoría de estos formatos varían según la herramienta ofimática que se use para trabajar con el documento.

1.4.1 Documentos de formato abierto

Un documento de formato abierto es un documento ofimático cuyo formato; modo en que se representan sus datos, entra dentro de la clasificación de formato abierto, que se define como el modo de representación transparente de los datos del documento y porque la especificación del formato está disponible públicamente. Los formatos abiertos son, ordinariamente, estándares determinados por

autoridades públicas o instituciones internacionales cuyo objetivo es establecer normas para la interoperabilidad de *software*. No obstante, hay casos de formatos abiertos promovidos por compañías que eligen hacer la especificación de los formatos usados por sus productos disponibles públicamente. (Puigpinos, 2013)

1.4.2 PDF (Formato de Documento Portable)

El formato PDF es el formato abierto y nativo de la familia de productos Adobe Acrobat. PDF se basa en un lenguaje de descripción de páginas, heredero de PostScript[3], para describir texto y gráficos de forma independiente de la plataforma y de la resolución. (Graells, 2008)

PDF define un formato más estructurado que PostScript para facilitar una visualización interactiva e incluye otros elementos como anotaciones, vínculos o formularios –incluso JavaScript en las últimas versiones– pensados exclusivamente para el entorno digital. A partir de la versión 1.4 los archivos PDF incorporan estructura en la descripción del documento mediante etiquetas similares a HTML. Adobe facilita además una versión pública del *software* de lectura, Adobe Reader, que en sus últimas versiones ha incorporado una serie de funciones para facilitar la accesibilidad. (Graells, 2008)

1.4.3 ODF (Formato de Documento Abierto)

La Organización Internacional para la Estandarización ha elegido el formato *OpenDocument* como un estándar para el almacenamiento e intercambio de texto con formato, contenido en documentos ofimáticos tales como hojas de cálculo, textos, gráficas y presentaciones. Las extensiones de formato de archivo identificativas en los archivos *OpenDocument* incluye: ODT (Documento de Texto Abierto) para documentos de texto, ODS (Documento de Hojas de cálculo Abierto) para hojas de cálculo, ODP (Documento de Presentación Abierto) para presentaciones, ODG (Documento de Gráficos Abierto) para gráficos y ODB (Documento de Base de datos Abierto) para bases de datos. Un documento ODF es un formato basado en XML[4]; un fichero en este lenguaje es un fichero de texto que no requiere ninguna aplicación especial para leerlo. Por esta razón es multiplataforma: un documento ODF es independiente

[3] Es un lenguaje de descripción de páginas (en inglés PDL, *page description language*), desarrollado por Adobe y utilizado en muchas impresoras y de manera usual, como formato de transporte de archivos gráficos en talleres de impresión profesional.

[4] Es un lenguaje de marcas desarrollado por el *World Wide Web Consortium* (W3C). Deriva del lenguaje SGML y permite definir la gramática de lenguajes específicos para estructurar documentos grandes.

no solamente de la aplicación con la que se ha creado, sino también del sistema operativo. Además, sigue fielmente el principio de la separación de contenido y formato tan característico de XML. En concreto, un documento ODF es un archivo comprimido que contiene varios ficheros cuyos nombres son bastante indicativos: el contenido en sí mismo (content.xml), los metadatos sobre el documento (meta.xml), el estilo (styles.xml) y características adicionales del mismo (settings.xml), así como archivos gráficos si el documento contiene ilustraciones. ODF se ha desarrollado de forma expresa para que el código fuente de los documentos se pueda interpretar; es decir, no solo que pueda ser visto sino entendido a simple vista. (Codina, 2008)

El formato de archivo de documentos de *OpenDocument* más utilizado es ODT. Un estándar abierto para documentos electrónicos, especial para documentos de editores de texto. Tiene similar función que los DOC y DOCX de Microsoft Office. Fue desarrollado por Sun Microsystems y OASIS, es una extensión del XML y sigue el estándar ISO/IEC 26300:2006. (Weir, 2012)

Toda esta variedad de documentos y de formatos genera un flujo continuo de información y surge la necesidad de gestionarla de forma adecuada, es aquí donde pasa a jugar un papel determinante la gestión documental. Es debido a esos inmensos volúmenes de información que se procesan en muchas organizaciones y a los adelantos tecnológicos de la actualidad, que se deben utilizar equipos de cómputo adecuados para darle solución a los diversos problemas que trae consigo el manejo de la información. Es en este ámbito donde el empleo de herramientas ofimáticas, que posibiliten a los usuarios acceder a la información de forma eficaz, garantiza el desarrollo normal de las actividades y el cumplimiento de las tareas.

Dentro de la gran variedad de herramientas ofimáticas como: AbiWord, Atlantis Word Processor, IBM Lotus Symphony, KOffice, Microsoft Office, NeoOffice, Okular, OpenOffice, WordPad, Zoho Office Suite, se destacan, y son de gran utilidad para el presente trabajo el OpenOffice y el Adobe Reader, por ser herramientas ofimáticas de gran uso y comercialización en el mundo, y que a su vez gestionan los formatos de documentos de mayor utilización en el ámbito social en que se desempeñan, ya sea en universidades, centros laborales e instituciones de desarrollo, donde se están creando, compartiendo y consultando elementos documentales.

1.5 OpenOffice

OpenOffice es un paquete ofimático, es decir, es un conjunto de programas básicos requeridos para la utilización habitual de un ordenador que realiza tareas básicas de oficinas. Incluye herramientas como un procesador de textos, una hoja de cálculo, presentaciones, herramientas para el dibujo vectorial y base de datos. (Martín, 2013)

Este producto se puede utilizar con diversas plataformas, como Microsoft Windows, Linux, BSD, Solaris y Mac OS X. OpenOffice es compatible además con Microsoft Office, aunque inicialmente se creó para ofrecer una alternativa abierta y de alta calidad a este producto. (Martín, 2013)

La ventaja más importante que posee es que puede ser usado de manera gratuita, un gran aporte para este trabajo investigativo, además existe un mejor acceso a la información sobre el desarrollo e implementación de dicha herramienta, así como del uso y trabajo con el formato ODF, con el cual se deberá trabajar para darle solución a la problemática planteada en esta investigación.

1.6 Adobe Reader

Adobe Reader es un *software* que permite leer, navegar e imprimir los ficheros de documentos en el formato PDF, de forma independiente o desde el navegador. Algunas de las opciones de Adobe Reader son la integración con los navegadores, la mejora de impresión en PCL[5] y Adobe PostScript, soporte para documentos extensos, soporte para todas las versiones de PostScript (incluyendo la 3.0), soporte para el formato PDF 1.3, buen manejo del color (con soporte para plantillas ICC[6]) y una total compatibilidad con todas las versiones previas. Permite leer y organizar eBooks[7], extraer imágenes de presentaciones creadas con Adobe Photoshop[8], ver diferentes capas en documentos PDF creados para tal efecto. (Leurs, 2001)

---

[5] *Printer Command Language* (PCL), es un lenguaje de descripción de páginas muy sofisticado desarrollado por Hewlett Packard para impresoras láser.

[6] Es un conjunto de datos que caracteriza a un dispositivo de entrada o salida de color, o espacio color, según los estándares promulgados por el Consorcio Internacional del Color (ICC)

[7] Es una versión electrónica o digital de un libro o un texto publicado en la *World Wide Web* o en otros formatos electrónicos.

[8] Una aplicación informática en forma de taller de pintura y fotografía que trabaja sobre un "lienzo" y que está destinado a la edición, retoque fotográfico y pintura a base de imágenes de mapa de bits.

La integración del Adobe Reader con los navegadores web es un aspecto importante a analizar a la hora de darle solución a la problemática del trabajo investigativo, porque uno de los tópicos a resolver en el módulo propuesto, es la lectura de documentos en el formato PDF desde el cliente web del eXcriba, por lo que el estudio y análisis de cómo el Adobe Reader se integra con los navegadores web, será de gran ayuda y servirá de punto de partida para la realización de la solución final.

1.7 Evolución de Internet y de las tecnologías web

Con la evolución de Internet y de las tecnologías web las herramientas ofimáticas han dejado de ser simples aplicaciones de escritorio en las computadoras de oficinas. Se han convertido en potentes herramientas que gestionan toda la información que se despliega diariamente en la red de redes, y en cualquier ámbito donde exista una infraestructura de red, ya sea en las mismas oficinas, en las escuelas, universidades e instituciones. De modo que, el ser humano se ha visto en la necesidad y ha requerido que se implementen versiones de estas herramientas ofimáticas para la web.

Las ventajas más inherentes específicas de estas herramientas *online* son:

- La movilidad total en el acceso a la información de trabajo desde cualquier lugar, independientemente de donde se encuentre la conectada la computadora.
- El ahorro económico, pues no es necesario adquirir ningún paquete o herramienta ofimática.
- No es necesaria la instalación de *software* adicional en la computadora, lo que supone una ventaja para discos duros con poca capacidad o procesadores más lentos.
- Compartir o editar documentos entre varios usuarios (o simplemente visualizarlos) en tiempo real.

Para acceder a estas herramientas solo se necesita de una conexión a la web y de un navegador en el lugar donde se desee trabajar con los documentos.

Existen gran variedad de herramientas ofimáticas *online* entre las que se encuentran: Zoho, Stilus, Peepel, Gliffy, Picnik, y Office Live Workspace, pero definitivamente el de mayor uso y aceptación por parte de los usuarios es Google Docs.

1.7.1 Google Docs

Google Docs es un conjunto de productos que permiten crear distintos tipos de documentos, trabajar en ellos con otros usuarios en tiempo real y almacenar documentos y otros archivos, todo *online* y de forma gratuita. Con una conexión a Internet, se puede acceder a los documentos y archivos desde cualquier ordenador. Incluso es posible realizar algunas tareas sin necesidad de conectarse a Internet. (Docs, 2013)

Los usuarios de esta herramienta tienen la posibilidad de manejar gran cantidad de formatos que ella soporta, con su procesador de textos se logra editar documentos de Microsoft Office, OpenOffice, Adobe Reader y guardarlos con el mismo formato u otros distintos.

Por las características vistas en esta herramienta se hace necesario su estudio y análisis, para conocer a profundidad cómo se realiza la edición y lectura de documentos en línea, que logre desarrollar una solución más eficiente y robusta de la problemática planteada en este trabajo investigativo.

En la evolución de Internet ha jugado un rol determinante el surgimiento de nuevas tecnologías web, que posibilitan realizar una gran variedad de tareas y actividades dentro de la web. Esto también ha sido posible por la aparición de nuevos lenguajes, o la actualización de los ya existentes a versiones superiores, que permiten un mejor desarrollo de las aplicaciones web y la exploración de nuevos campos de utilidad para estas tecnologías, dentro de estos lenguajes se hallan JavaScript y HTML5.

1.7.2 PDF.js

Dentro de las tecnologías web que existen se encuentra PDF.js desarrollada por Andreas Gal[9]. Es una tecnología que analiza las matrices de bytes primas en flujos de código de bytes (*bytecode*) de PDF, compila el *bytecode* en los programas de JavaScript, y entonces ejecuta los programas. El efecto secundario de estos programas es la de dibujar un HTML5 <canvas>[10]. Los comandos en el código de

[9] Vicepresidente de ingeniería de *Mozilla Mobile*. Anteriormente, trabajó como científico del proyecto de la Universidad de California, Irvine, trabajando en los sistemas de seguros y Laboratorio de Idiomas Profesor Michael Franz. Su formación es en los sistemas de seguridad, lenguajes de tipo seguro, compilación dinámica, y las máquinas virtuales.

[10] El elemento HTML5 <canvas> se utiliza para dibujar gráficos, sobre la marcha, a través de secuencias de comandos (normalmente JavaScript).El elemento <canvas> es solo un contenedor para gráficos. Se debe utilizar una secuencia de comandos para dibujar en realidad los gráficos. Canvas tiene varios métodos para rutas de dibujo, cajas, círculos, personajes e imágenes agregando.

bytes incluyen cosas simples como "dibujar una curva", "dibujar este texto aquí", y las cosas más complicadas como rellenar áreas con "patrones de sombreado" especificadas por las funciones personalizadas. Además, el flujo de comandos en sí, y otros datos embebidos como fuentes e imágenes, podría ser comprimido y/o codificado en la matriz de bytes en bruto. PDF.js tiene soporte básico para descomprimir algunos de estos flujos, todo el código está escrito en JavaScript.

Hay varias formas de escribir un PDF renderizado en la parte superior de la plataforma web, la implementación PFD.js actual, dibujando con canvas, es solo una forma. Se eligió canvas inicialmente porque es la manera más rápida para que se pueda dibujar en la pantalla. Se quiere que el primer dibujo de las páginas sea rápido para que el arranque de PDF.js sea enérgico, y los usuarios puedan comenzar a leer el contenido lo antes posible.

A pesar de las ventajas aportadas por canvas, no cuenta con algunas de las características necesarias para renderizar los archivos PDF. Se han añadido algunas de estas a la plataforma web, cuando se haga necesario su uso. Otra limitación es que el agente de usuario no dispone de información suficiente para permitir a los usuarios seleccionar texto o navegar usando interfaces de accesibilidad, aun cuando permite de inmediato un modo de diseño con un mínimo de gastos.

La plataforma web ya ofrece una solución potencial a estos problemas, sin embargo: SVG [11] está ricamente destacado, con modo retenido, y tiene su propio DOM[12]. En teoría, los agentes de usuario deben tener selección de texto y soporte de impresión para SVG. Así, SVG proporciona los elementos que faltan.

Sobre la base de las ventajas que ofrece canvas y SVG, se puede hacer un rápido primer dibujo de páginas con canvas, y al mismo tiempo la construcción de un documento SVG para la página en segundo plano, y cuando el documento SVG esté listo, se muestra completamente. (Pitchin, 2013)

[11] *Scalable Vector Graphics* (SVG) es el modelo de imagen basado en XML estándar de W3C que permite que los desarrolladores, diseñadores y usuarios de la web superen las limitaciones del HTML y creen contenido visual atractivo e interactividad mediante un sencillo modelo de programación de declaraciones.

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

1.7.3 WebODF

Otra de las tecnologías web existentes es: WebODF, tiene como principal característica que puede mostrar un documento de Office en el formato *OpenDocument* en un navegador web. Esta hace posible al usuario tener la misma experiencia en el navegador como si se estuviera trabajando en el escritorio.

Con WebODF, se combinan las ventajas que posibilita “la nube”[13] –*the cloud* en inglés- y el control del escritorio de código abierto. Se puede acceder a los documentos de oficina desde cualquier lugar, pero con las condiciones del usuario. No es necesario estar de acuerdo con los términos de servicio, ya que puede proporcionar su propio servicio o elegir una empresa de servidores web que se prefiera y seguir utilizando la versión de WebODF.

WebODF se beneficia de normas cada vez más compatibles, como CSS[14], HTML[15], DOM y JavaScript, colectivamente se conocen con los nombres de HTML5 y AJAX. Esto hace que sea posible llevar a cabo las funciones de oficina con pequeñas cantidades de código, que a su vez, hacen a WebODF compacto y fácil de entender.

El conjunto de características que se necesitan para representar documentos ODF en el navegador es compatible con la última versión de navegadores. Un navegador puede hacer mucho más que simplemente visualizar una página web. Las páginas web pueden ser aplicaciones completas. Además de eso, los navegadores pueden mostrar archivos XML genéricos. La visualización se realiza entonces con CSS, aunque hay que prestar cuidado y usar siempre las características de CSS que más compatibles sean entre todos los navegadores. (WebODF, 2013)

Por las potencialidades vistas en estas dos tecnologías la solución final de este trabajo investigativo se basará en gran medida en cómo están desarrolladas estas, aprovechando las libertades que brindan ambas, al ser tecnologías libres y de código abierto.

---

[13] Permite una separación funcional entre los recursos que se utilizan y los recursos de tu computadora, esto es: se utilizan recursos en un lugar remoto y que se acceden por Internet.

[14] Es un lenguaje usado para definir la presentación de un documento estructurado escrito en HTML o XML (y por extensión en XHTML).

[15] Es un lenguaje de programación que se utiliza para el desarrollo de páginas web. Se trata de la sigla que corresponde a *Hyper Text Markup Language*, es decir, Lenguaje de Marcas de Hipertexto, que podría ser traducido como Lenguaje de Formato de Documentos para Hipertexto.

Las ventajas que ofrece la edición y lectura de documentos en línea (*online*), como se ha venido analizando hasta aquí, ha planteado la necesidad de mejorar los Gestores de Contenido Empresarial (ECM).

1.8 Gestores de Contenido Empresarial (ECM)

Los Gestores de Contenido Empresarial, en sus siglas en inglés ECM (*Enterprise Content Management*) hacen referencia al manejo de contenido empresarial, ya sea como medio impreso o electrónico.

La Asociación para la Información y Gestión de la imagen (en inglés *Association for Information and Image Management*) lo definió en el año 2010 de la siguiente manera:

“*Enterprise Content Management* (ECM) son las estrategias, métodos y herramientas usadas para capturar, manejar, salvaguardar, preservar y entregar contenido y documentos relacionados con procesos organizacionales. ECM cubre la gestión de la información dentro del ámbito completo de una empresa, ya sea que esa información esté en forma de documento de papel, un archivo electrónico, una base de datos impresa e incluso un email.” (Management, 2013)

En la actualidad el ECM es usado en muchas empresas, dado que ayuda en el proceso organizacional y, como tal, aumenta la rentabilidad de la empresa, mediante el mejoramiento de los aspectos relacionados con la organización y manejo de documentación.

El ECM también combina varios elementos de la gestión de documentación, por ejemplo el manejo, captura, búsqueda de archivos digitales. Entre los aspectos principales involucrados en un ECM se incluye el uso y la preservación de información privilegiada para una empresa. Este a su vez se puede entender como una evolución de la Gestión de Contenido, ya que plantea una nueva forma de gestión, además de incluir nuevos formatos, también implica una nueva forma de manejo de información, como protocolos para el ingreso y salida de información, generación de metadatos con relación a la documentación y algunos aspectos de automatización.

Es decir, que con el Enterprise Content Management se hace frente a un mundo donde la gestión de la información tiene fronteras tecnológicas, ya que este nuevo enfoque implica un manejo computarizado de los datos.

La siguiente figura recoge, los principales elementos que componen a los ECM.

Ver anexo A.1 Figura 3, de la versión extendida de este documento.

1.8.1 Alfresco

Entre los Enterprise Content Management (ECM) se destaca Alfresco, un producto que en el mercado de los ECM ha ganado gran aceptación al promover soluciones de gestión documental a las empresas usando tecnologías de código libre y estándares abiertos.

Con Alfresco las empresas clientes pueden reducir el costo, minimizar los riesgos y adquirir ventajas competitivas adoptando las bases de las tecnologías de código abierto. Se pueden reducir los costos de adquirir soluciones y *software*, así como los costos de mantenimiento.

Las principales funcionalidades que presenta Alfresco son:

- Catalogar y clasificar los documentos por múltiples criterios.
- Extraer metadatos de los documentos automáticamente.
- Asignarles propiedades y características.
- Búsquedas complejas, incluyendo el contenido del documento, metadatos, categorías.
- Enviar correos y notificaciones a otros usuarios incluyendo referencias al documento.
- Crear flujos de trabajo, aprobación, supervisión.
- Control de versiones y auditoría.
- Colaboración en el desarrollo de documentos, bloqueo de documentos en la edición, edición *online* y *offline*.
- Seguridad de documentos mediante usuarios, roles y carpetas.
- Ejecutar acciones, como conversiones, cambios de formato, envíos automáticos.
- Compartir documentos con otros usuarios
- Incluir documentos en *blogs* y web de forma sencilla.

- Carga de documentos a través del propio explorador de archivos de Windows (CIFS).
- Posibilidad de empezar a trabajar con esta herramienta de forma rápida y efectiva. Instalación, completamente configurada y adaptada a sus necesidades.
- Separación del repositorio y base de datos del motor.
- Configuración del acceso mediante FTP, CIFS e IMAP.
- Configuración de integración con clientes de correo y paquetes *office*.
- Configuración de *backup* y restauración segura.

1.9 Edición y lectura de documentos de formato abierto en los Gestores de Contenido Empresarial.

La edición y lectura de documentos de formato abierto en los ECM se realiza generalmente de dos formas: a través de la integración de los mismos, con herramientas ofimáticas en línea (*online*), como son: Zoho y Google Docs, es decir edición *online* de documentos, y otra a través de la edición *offline* de documentos.

La edición *offline* o en desconexión de documentos desde los ECM permite trabajar con el documento de forma local (en la computadora del usuario) mediante el empleo de herramientas ofimáticas de escritorio, y subir el documento modificado una vez terminado. Para ello es necesario la descarga del documento hacia la computadora del usuario, a través de una pantalla de descarga de archivos, para poder editarlo de forma local. Al realizar la descarga el sistema crea una copia de trabajo (documento que se descarga), de esta forma, el original no se toca y las modificaciones se realizan sobre la copia de trabajo.

Cuando el usuario termina de introducir las modificaciones en la copia de trabajo puede subir la misma hacia el repositorio documental y dejarla abierta para que otros usuarios verifiquen los cambios, de lo contrario puede simplemente actualizar el documento original con los cambios realizados a la copia de trabajo, la que se elimina luego de este proceso.

Por otra parte, la edición *online* de documentos desde los ECM, particularmente desde Alfresco, está basada principalmente en la integración de los mismos con herramientas ofimáticas en línea, dentro de las que se destaca Google Docs, la cual ofrece a los usuarios acceso a una completa herramienta de edición en línea, lo que posibilitará que estos no hagan uso de aplicaciones ofimáticas de escritorio, como Microsoft Office, OpenOffice y Adobe Reader, para modificar o actualizar los contenidos gestionados en

un repositorio de Alfresco. La integración permite a los usuarios de Alfresco abrir contenidos directamente en Google Docs para su edición conjunta en tiempo real, al tiempo que añade más contenidos a un entorno de gestión integral de ECM ya de por sí lleno de utilidades y funcionalidades como la gestión de metadatos y seguridad de Alfresco.

Además de la integración con herramientas ofimáticas, que posibilita al Alfresco la edición de documentos en línea, esta herramienta brinda acceso al repositorio documental a través del protocolo WebDAV, que proporciona funcionalidades para crear, cambiar y mover documentos en un servidor remoto (generalmente un servidor web), y el cual es utilizado sobre todo para permitir la edición de los documentos almacenados en un servidor web, pero puede también aplicarse a sistemas de almacenamiento generales basados en web, a los que puede accederse desde cualquier lugar.

Otra forma de trabajar con los documentos la provee Alfresco Share[16], que ofrece la funcionalidad de ver el contenido de los documentos almacenados en el mismo en un visor flash[17], lo que permite a los usuarios visualizar el contenido independientemente de la aplicación o versión de producto con la que se genera. Esta solución para visualizar documentos que brinda el Alfresco Share, tiene la desventaja que depende que en el navegador web sobre el cual trabaja el usuario, tenga instalado los últimos plugins[18] de flash, para trabajar con esta tecnología, lo que crea dependencia. (Bhandari, 2012)

Alfresco Share también provee la integración con conjunto de visores de contenido para visualizar cualquier documento o elemento de contenido seleccionado en un panel web. Para el caso de los documentos brinda integración con los visores PDF.js y WebODF (Abson, 2013), cuyas APIs de desarrollo serán tenidas a consideración en la implementación del módulo final que se propone.

Como parte de la gestión de los documentos que realiza Alfresco, el mismo tiene la capacidad de transformar el contenido entre diferentes tipos MIME[19]. El usuario puede hacerlo de forma manual o mediante la invocación de una regla sobre los contenidos o sobre un espacio. El proceso de transformación de contenidos se realiza mediante el uso de bibliotecas de terceros, tales como Apache

[16] Es una plataforma moderna basada en la Web para la colaboración y la gestión social de contenidos.

[17] Es una tecnología creada por Adobe para el desarrollo de distintas funciones, que permite la creación de animaciones vectoriales y muchas aplicaciones en la web.

[18] Es un módulo de *software* que añade una característica o un servicio específico a un sistema más grande.

[19] Es un estándar que clasifica los recursos y provee información (a los programas) acerca de cómo manejarlos.

PDFBox [20], y aplicaciones como OpenOffice (que se ejecuta en el servidor). (Shariff, 2010) Esta funcionalidad permite convertir gran variedad de documentos de diferentes formatos a uno o varios formatos específicos, esto es de gran aporte para la investigación, ya que brinda una vía de cómo trabajar los diferentes documentos que se encuentran almacenados en el repositorio y que no entran en la categoría de documentos de formato abierto, y que a través de esta funcionalidad del Alfresco, pueden ser convertidos a un formato abierto y ser editados y visualizados usando el módulo que se propone, como solución de la investigación.

Hasta este momento de la investigación se ha realizado un estudio y análisis del sistema homólogo Alfresco, así como del sistema Google Docs y de tecnologías libres para la web, con el fin de comprender a profundidad cómo se desarrolla y se implementa la edición de documentos de formato abierto en la web. Lo que posibilita hacer una valoración de cuáles de estos sistemas, aporta más a la solución final que se propone en este trabajo.

El análisis de la herramienta ofimática *online* Google Docs, permitió ver un ejemplo concreto, de cómo visualizar un documento ofimático en un navegador web, lo que constituye un aporte a esta investigación. También permitió observar, cómo el acceso a esta herramienta *online* se realiza a través de Internet, sin embargo no es esta una opción viable para la solución final, por las limitaciones existentes en el uso de Internet, además que el acceso al código fuente de la misma está restringido por licencias privativas. Por lo anteriormente planteado esta herramienta es de utilidad para la investigación, aunque no será utilizada para el desarrollo de la solución final.

Otro de los sistemas analizados es PDF.js, un API de JavaScript que posibilita la visualización de documentos con formato PDF en un navegador web, usando el motor de renderizado del propio navegador, canvas, HMTL5, y el propio lenguaje JavaScript. El análisis y estudio de este sistema, permitió entender con facilidad cómo está implementado el mismo y su facilidad de integración con otros sistemas, lo cual es un gran aporte al desarrollo de la solución final.

Un sistema muy similar al PDF.js, y sobre el cual también se realizó un estudio, es el WebODF, un API de JavaScript que al igual que PDF.js permite la visualización de documentos en un navegador web, pero en este caso particular permite la visualización y edición de documentos con formato ODF. Este API es

[20] Es una herramienta de código abierto en Java para trabajar con documentos PDF.

implementado en JavaScript, y aprovecha las posibilidades que brinda el lenguaje HMTL5 y los motores de renderizado de los navegadores. El mismo será de gran utilidad para desarrollar la solución final.

El acceso libre y sin restricciones al código fuente de las dos API de JavaScript analizadas con anterioridad, permitió comprender en profundidad cómo están implementadas, lo que ha demostrado la importancia y aporte de las mismas a la solución final del trabajo investigativo, así como la importancia del uso de lenguajes como: JavaScript y HMTL5, el uso de canvas, y de los motores de renderizado de los navegadores web.

El único sistema homólogo estudiado fue el Gestor de Contenidos Empresariales Alfresco, sistema que cuenta con una variedad de funcionalidades para gestionar de forma adecuada toda la documentación generada por las empresas u organizaciones. Una de las funcionalidades que contempla es la visualización y edición de documentos, la cual puede ser realizada de diversas formas, entre ellas están: la edición *online* de documentos, la edición *offline* de documentos, la edición de documentos a través del protocolo WebDAV, así como a través de visores de contenidos integrados al mismo y mediante la integración con herramientas ofimáticas. Producto al libre acceso que existe al código fuente de este sistema, y que además el Gestor de Documentos Administrativos eXcriba, usa como núcleo Alfresco, el estudio de este sistema es de gran aporte para la investigación y para el desarrollo de la solución final.

Después de exponer los fundamentos sobre los que se sustenta la investigación, se hace necesario analizar las herramientas, tecnologías, lenguajes de programación y metodologías para el desarrollo de la solución.

1.10 Metodología de desarrollo de *software*

Las metodologías de desarrollo de *software* son un conjunto de herramientas, métodos y procesos con un enfoque de calidad para el desarrollo de productos *software*, en el que se van indicando paso a paso todas las actividades a realizar para lograr el producto informático desaseado y con la calidad requerida, indicando además qué personas deben participar en el desarrollo de las actividades y qué papel deben de tener. Además, detallan la información que se debe producir como resultado de una actividad y la información necesaria para comenzarla. (IAGP, 2005/2006)

1.10.1 *Rational Unified Process* (RUP) con nivel 2 de CMMI

El *Rational Unified Process* (RUP) es un proceso de desarrollo de *software*, este comprende un conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema de *software*. Este

define claramente quién, cómo, cuándo y qué debe hacerse en un proyecto. El proceso unificado combinado con UML como lenguaje de modelado conforma la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientado a objetos. (Jacobson, 2004)

CMMI es un modelo de integración que constituye una de las mejores prácticas de los modelos de mejora de procesos. El empleo de RUP con Nivel 2 de CMMI permite incrementar la satisfacción de los usuarios internos mediante una correcta implementación de productos de *software* de calidad, dentro del tiempo y costo estimado.

RUP se distingue por tres características:

- Dirigido por casos de usos

Un caso de uso es un fragmento de funcionalidad del sistema que proporciona al usuario un resultado importante. Los casos de uso representan los requisitos funcionales que no solo inician el proceso de desarrollo sino que le proporcionan un hilo conductor. Dirigido por casos de uso quiere decir que el proceso de desarrollo sigue un hilo que avanza a través de una serie de flujos de trabajo que parten de los casos de uso. Los casos de uso se especifican, se diseñan, y los casos de uso finales son la fuente a partir de la cual los ingenieros de prueba construyen sus casos de prueba. (Jacobson, 2004)

- Centrado en la arquitectura

Abarca diferente vistas del sistema: estructural, funcional, dinámica, la plataforma en que se va a desarrollar y la forma del sistema. La arquitectura muestra la visión común del sistema completo en la que el equipo de proyecto y los usuarios deben estar de acuerdo, por lo que describe los elementos del modelo que son más importantes para su construcción, los cimientos del sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo económicamente. RUP propone el desarrollo del *software* mediante iteraciones comenzando por los casos de uso relevantes desde el punto de vista de la arquitectura. (Jacobson, 2004)

- Iterativo e Incremental

El desarrollo de un producto *software* supone un gran esfuerzo que puede durar entre varios meses hasta posiblemente un año o más. Es práctico dividir el trabajo en partes más pequeñas o mini proyectos. Cada mini proyecto es una iteración que resulta en un incremento. Las iteraciones hacen referencia a pasos en el flujo de trabajo, y los incrementos, al crecimiento del producto. Para una efectividad máxima, las

iteraciones deber estar controladas; deben seleccionarse y ejecutarse de una forma planificada. (Jacobson, 2004)

Se ha elegido esta, como la metodología a aplicar en el desarrollo del módulo que le dará solución al objetivo de este trabajo investigativo, porque es la que se definió en el proyecto eXcriba para generar los artefactos y módulos con los que cuenta el sistema, además por todas las características antes expuesta de la metodología y la amplia documentación que existe sobre la misma.

1.11 Tecnologías

1.11.1 REST

REST define un conjunto de principios arquitectónicos por los cuales se puede diseñar servicios web que se centran en los contenidos del sistema, incluyendo cómo los estados de los contenidos se dirigen y se transfieren a través de HTTP por una amplia gama de clientes escritos en diferentes idiomas. Si se mide por el número de servicios web que lo utilizan, REST ha surgido en los últimos años como un modelo de diseño Web predominante. De hecho, REST ha tenido un impacto tan grande en la web que se ha desplazado principalmente en SOAP y WSDL basado en el diseño de interfaces, porque es un estilo mucho más fácil de usar. (Rodriguez, 2008)

Una implementación concreta de un servicio web REST sigue cuatro principios básicos de diseño:

- Utilice métodos HTTP explícitamente.
- Exponer la estructura de directorios de tipo URI.
- Transferencia XML, *JavaScript Object Notation* (JSON), o ambos.

El uso de este servicio web será de gran utilidad para la solución final de la investigación, ya que posibilitará el trabajo con el Gestor de Contenidos Empresariales Alfresco, permitiendo el acceso a los recursos almacenados en dicho gestor, en este caso particular a los documentos de formato abierto que en este se gestionan. Además porque es la forma en la que se encuentra implementado el sistema.

1.12 Lenguajes de programación

1.12.1 JavaScript

JavaScript es un lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas. Una página web dinámica es aquella que incorpora efectos como texto que aparece y desaparece, animaciones, acciones que se activan al pulsar botones y ventanas con mensajes de aviso al usuario. Técnicamente, JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios. (Eguíluz, 2009)

Por las facilidades vistas anteriormente que brinda JavaScript y por ser un lenguaje del lado del cliente, es decir se ejecuta directamente en las computadoras de los usuarios, no desde un servidor web, se utilizará la versión 1.5 de este lenguaje para implementar gran parte de las principales funcionalidades que requerirá el módulo, que dará solución a la problemática planteada en esta investigación.

1.12.2 PHP

PHP es un lenguaje "del lado del servidor" (esto significa que PHP funciona en un servidor remoto que procesa la página web antes de que sea abierta por el navegador del usuario) especialmente creado para el desarrollo de páginas web dinámicas. Puede ser incluido con facilidad dentro del código HTML, y permite una serie de funcionalidades tan extraordinarias que se ha convertido en el favorito de millones de programadores en todo el mundo. (Gallego, 2003)

Entre sus características fundamentales están:

Gratuito. Al tratarse de *software* libre, puede descargarse y utilizarse en cualquier aplicación, personal o profesional, de manera completamente libre.

Gran popularidad. Existe una gran comunidad de desarrolladores y programadores que continuamente implementan mejoras en su código.

Integración con múltiples bases de datos.

Versatilidad. PHP puede usarse con la mayoría de sistemas operativos.

Gran número de funciones predefinidas. A diferencia de otros lenguajes de programación, PHP fue diseñado especialmente para el desarrollo de páginas web dinámicas. Por ello, está dotado de un gran número de funciones que simplificarán enormemente tareas habituales como descargar documentos, enviar correos, trabajar con cookies y sesiones, etc.

Se utilizará PHP como lenguaje de programación del lado del servidor, ya que el Gestor de Documentos Administrativos eXcriba, sistema para el cual se implementará un módulo que permita la edición y lectura de documentos de formato abierto, tiene como restricción el uso de este lenguaje de programación, y también por las características y facilidades que brinda el mismo.

1.13 Lenguaje de modelado

1.13.1 UML

Es un lenguaje gráfico que ofrece un modo estándar de visualizar, especificar, construir, documentar y comunicar los artefactos de un sistema muy basado en el *software*. (Jacobson, 2004)

UML entrega una forma de modelar cosas conceptuales como lo son procesos de negocio y funciones de sistema, además de cosas concretas como lo son escribir clases en un lenguaje determinado, esquemas de base de datos y componentes de *software* reusables. (Jacobson, 2004)

En el desarrollo de la solución propuesta en este trabajo investigativo se hará uso de UML en su versión 2.0, para modelar los artefactos que se creen en el proceso de desarrollo del *software*. La elección de este lenguaje viene dada por la metodología de desarrollo que se seleccionó (RUP), ya que el lenguaje de modelado que viene asociado a la misma es UML.

1.14 Herramientas

1.14.1 ZendStudio

Zend Technologies ofrece Zend Studio, un IDE líder para PHP. Cubre todas las típicas características de un IDE más un conjunto de características especiales para el entorno PHP con el objetivo de ayudar a la productividad del desarrollador de PHP.(Aulke, 2007) Esta lista muestra alguna de las características especiales de Zend Studio:

- SOAP: generación del WSDL a partir de la definición de clases de PHP (incluyendo la evaluación de comentarios PHPDoc).

- SOAP: autocompletado de código para clientes SOAP basado en un archivo WSDL (local o remoto).

- SCM: soporte integrado para Subversion y CVS.

- DB: visor de la estructura de base de datos y de contenidos, cliente SQL para diferentes tipos de bases de datos.

- Editor: gestión de archivos vía FTP o SFTP para una edición remota rápida.

- Editor: coloreo de la sintaxis. Coloreo de errores.

- Editor: autocompletado de código para funciones PHP y código propio basado en comentarios PHPDoc.

- Editor: analizador de Código para encontrar problemas típicos en código PHP.

- Editor: plantillas de código PHP.

- Depurador: depuración Local o Remota.

- Depurador: profiling Local o Remoto.

- Depurador: barra del navegador que permite empezar el depurado de errores en cualquier lugar de su aplicación.

- Integración: autocompletado de código Java para el uso de Java Bridge.

Por las características de esta herramienta, su fácil integración con el framework de desarrollo CodeIgniter, utilizado para implementar el eXcriba, y sobre el cual se implementará el módulo que dará solución al objetivo de este trabajo investigativo, y además por ser la herramienta que se definió en la arquitectura del proyecto, se elegirá la misma para el desarrollo de la solución final.

1.14.2 Visual Paradigm

Una de las líderes del mercado de las llamadas herramientas CASE. Permite el desarrollo de aplicaciones utilizando modelado UML, ideal para Ingenieros de Software, Analistas de Sistemas y Arquitectos de sistemas que están interesados en construcción de sistemas a gran escala y necesitan confiabilidad y estabilidad en el desarrollo orientado a objetos. (Paradigm, 2013) A continuación se ofrece una lista de las características principales que se tuvieron en cuenta para la selección del mismo:

Soporte para la versión 2.1 de UML: la versión más reciente de este lenguaje es la de mayor uso a nivel mundial y la que más documentación posee.

Interoperabilidad entre diagramas: Visual Paradigm es capaz de exportar los diagramas de un modelo a otro con mucha facilidad. Esto ahorra considerables cantidades de tiempo.

Generación de código ActionScript 3.0 desde los diagramas: Visual Paradigm es una de las pocas herramientas capaz de generar esta versión del lenguaje ActionScript.

Visual Paradigm se empleará para especificar y construir los diversos artefactos, ya que el producto terminado debe ser entregado al usuario con toda la documentación y el código fuente.

1.15 Marco de Trabajo

1.15.1 CodeIgniter

CodeIgniter es un marco de trabajo de aplicación web de código abierto escrito en el lenguaje PHP. Tiene muchas características que lo hacen destacar entre la multitud. Diferente a algunos otros frameworks PHP la documentación es muy minuciosa y completa, que cubre todos los aspectos de la estructura. (Griffiths, 2010)

El código fuente de CodeIgniter está escrito en PHP4, por lo que es compatible con PHP4 y PHP5, y se puede ejecutar en la mayoría de los servidores web. También utiliza el patrón de diseño Modelo Vista Controlador (MVC), que es una manera de organizar su aplicación en tres partes diferentes: la vista, la lógica y acceso a la base de datos. En el núcleo, CodeIgniter también hace un amplio uso del patrón de diseño Singleton. Esta es una manera para cargar las clases de modo que si se les llama varias veces, la misma instancia de la clase será devuelta. (Griffiths, 2010)

CodeIgniter viene con una serie de bibliotecas muy útiles y otros conjuntos de funciones que le ayudan a construir sus aplicaciones. Esto hace que usted se centre en la pequeña parte de la aplicación que es único, en lugar de la parte que se utiliza a través de todos los proyectos, como las consultas de bases de datos y los datos de análisis. (Griffiths, 2010)

Se decide utilizar CodeIgniter en su versión 1.7.2 del lado del servidor como marco de trabajo para implementar la lógica de negocio de la solución a desarrollar, porque se definió en la arquitectura del sistema y cumple con todo los requisitos necesarios para desarrollar dicha solución.

1.15.2 jQuery

JQuery es un framework para el lenguaje JavaScript y CSS. Implementa una serie de clases (de programación orientada a objetos) que permiten programar soluciones informáticas que son compatibles con cualquier navegador independientemente del que utilice el usuario, ya que funcionan de forma exacta en todas las plataformas más habituales.

JQuery ofrece una infraestructura con la que se tiene mucha mayor facilidad para la creación de aplicaciones complejas del lado del cliente. Por ejemplo, con jQuery se obtiene ayuda en la creación de interfaces de usuario, efectos dinámicos y aplicaciones que hacen uso de Ajax. Simplemente deben conocerse las librerías del framework y programar utilizando las clases, propiedades y métodos para la consecución de los objetivos. (Angel, 2013)

Se elegirá este framework para ayudar en el desarrollo de la solución de este trabajo investigativo, porque es una restricción para la implementación del sistema, y por las facilidades y particularidades que brinda para trabajar con el lenguaje JavaScript. Además en el desarrollo del Gestor de Documentos Administrativos eXcriba, se utiliza una versión del mismo, por lo que se aprovechará su existencia en el eXcriba.

1.16 Conclusiones del capítulo

El análisis realizado sobre el contexto actual en que se realiza la edición y lectura de documentos de formato abierto en el Gestor de Documentos Administrativos eXcriba, permitió confirmar la necesidad de desarrollar un módulo para el Gestor de Documentos Administrativos eXcriba que permita la edición y lectura de estos documentos desde el cliente web mediante el uso de tecnologías para la web, y a su vez permitió arribar a la conclusión de que es un objetivo alcanzable a través de las diferentes variantes expuestas anteriormente, considerando la más aceptada para esta investigación la de utilizar las tecnologías PDF.js y WebODF respectivamente. La variante de integrar el ECM Alfresco con alguna herramienta ofimática en línea (*online*) es factible para darle solución a la problemática de este trabajo investigativo, porque no se cuenta con el acceso necesario a Internet, que permita interactuar con dichas herramientas. También se seleccionaron las metodologías, tecnologías, lenguajes y herramientas para el desarrollo de la solución.

Capítulo 2 “Procesos de negocio”.

2.1 Introducción

En el presente capítulo se comienza a definir cómo debe funcionar el sistema, por lo que es necesario enfocarse en la definición del modelo de dominio, así como en la captura de requisitos. Es en este momento donde se definen los requisitos funcionales y no funcionales de la aplicación, los actores que interactúan con la misma y las relaciones que pudieran establecerse entre ellos, alcanzando como resultado el diagrama de casos de uso del sistema, junto con la descripción textual de cada uno de los requisitos identificados, los cuales son incluidos también en este capítulo.

2.2 Problema y situación problémica

El problema y situación problémica que se definen en esta investigación y que se explican en la introducción de este trabajo, están dados por los siguientes aspectos:

- El uso de la ofimática en línea (*online*) para la edición y lectura de documentos en la web, ha venido ganando grandes niveles de utilización y aceptación por parte de los usuarios de la misma.
- Es necesario ampliar su utilización hacia las diferentes plataformas que se usan para la gestión de documentos.
- En la Universidad de las Ciencias Informáticas (UCI), existe el desarrollo de una herramienta para la gestión de documentos, el Gestor de Documentos Administrativos eXcriba.
- Este Gestor no cuenta con la funcionalidad de editar y leer documentos de formato abierto de manera *online*, es decir desde el cliente web, por lo que es necesario implementar dicha funcionalidad, que facilite y agilice la gestión de documentos de formato abierto sobre el mismo.

2.3 Propuesta de solución

Como propuesta de solución al problema explicado anteriormente, en este trabajo investigativo, se propone el desarrollo de un módulo para el Gestor de Documentos Administrativos eXcriba que permita la edición y lectura de documentos de formato abierto desde el cliente web del mismo. Para facilitar y agilizar en gran medida el proceso de edición y lectura de documentos de formato abierto, mediante la edición o

lectura de los documentos *online*, es decir, directamente desde el cliente web del Gestor de Documentos Administrativos eXcriba, eliminando así los diferentes factores que retrasan y complejizan dicho proceso.

Mediante el uso de las tecnologías para la web, analizadas en el capítulo anterior: PDF.js y WebODF, se desarrollará un nuevo módulo para el eXcriba, utilizando los marcos de trabajo CodeIgniter y jQuery, y los respectivos lenguajes que corresponden a cada marco de trabajo, PHP y JavaScript.

La realización del módulo en el Gestor de Documentos Administrativos eXcriba permitirá:

- Visualizar documentos de formato PDF y ODF embebidos dentro del eXcriba.

  Una vez seleccionado algún documento de formato PDF y ODF, dentro del explorador del eXcriba, el sistema generará una nueva vista embebida dentro del eXcriba que visualice el documento seleccionado.

- Editar documentos de formato ODF dentro del eXcriba.

  Al seleccionar un documento de formato ODF dentro del explorador del eXcriba, el sistema generará una vista que posibilite al usuario editar la información contenida dentro del mismo, la cual una vez terminada la edición será actualizada en el Alfresco.

- Gestionar el tamaño de visualización de los documentos con formato PDF.

  Cuando se muestre la vista del documento con formato PDF seleccionado, se le permitirá al usuario gestionar el tamaño de visualización del mismo a través de botones de acción o de una lista desplegable con tamaños predefinidos.

- Descargar documentos de formato PDF.

  En la vista que muestra el documento con formato PDF previamente seleccionado, existirá un botón de acción que permita descargar el mismo, hacia la computadora cliente, para un posible traslado o trabajo fuera del cliente web del eXcriba.

2.4 Integración de la propuesta de solución con el GDA eXcriba

El GDA eXcriba es un sistema que está compuesto por un núcleo que es el excriba-core[21] y 15 módulos independientes. Este núcleo cuenta con subsistemas previamente implementados, los cuales son horizontales para todos los módulos entre los que se encuentran: Gestión de eventos, Gestión de acciones, Control de acceso y Comunicación con la capa de acceso al repositorio. A continuación se describe cada uno de estos subsistemas y se establece la relación de estos con la solución que se propone.

Gestión de eventos: Los eventos que ocurren en el sistema o sobre el módulo propuesto, que son provocados por un usuario, serán manejados por este subsistema. Manejar la edición o visualización de los diferentes formatos de documentos.

Gestión de acciones: Permite visualizar en la capa de presentación cualquier acción ejecutada por el usuario correspondiente al módulo que se propone. Una vez que el usuario elija el documento que desea leer o visualizar, este subsistema se encargará de gestionar todas las acciones necesarias.

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

Comunicación con la capa de acceso al repositorio: Le permite al sistema poder consumir los servicios REST que provee la capa de Alfresco. Esto le permite al módulo que se propone, hacer uso de estos servicios para obtener los documentos que serán leídos o visualizados por el usuario, que se encuentran almacenados en el Alfresco.

2.5 Modelo de dominio

El modelo de dominio es una representación visual de los conceptos u objetos del mundo real, significativos para un problema o área de interés. Representa clases conceptuales del dominio del problema que vienen a ser las ideas u objetos físicos y el/los enlaces de unos objetos con otros, ayudando de esta forma a la elaboración del glosario de términos, facilitando la comunicación entre los

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

desarrolladores del sistema y un mayor entendimiento del contexto en que se desarrolla el sistema, producto del empleo de un lenguaje común.

Representación del Modelo de Dominio

Ver anexo A.1 Figura 4, de la versión extendida de este documento.

2.6 Captura de requisitos

Un requisito es algo que el sistema tiene que hacer o una cualidad que debe tener. Un requisito existe ya sea porque el tipo de producto exige ciertas funciones o cualidades o porque el cliente quiere que el requisito sea parte del producto entregado. La especificación de requisitos de *software* establece la base para el acuerdo entre los clientes y los desarrolladores, de lo que deberá hacer el producto de *software*, así como lo que no se espera que este haga.

2.6.1 Definición de las técnicas de obtención de requisitos

Los requisitos son la base para todo lo que hay que seguir en el ciclo de desarrollo de un producto, por lo tanto, es lógico pensar que estos deben ser entregados a diseñadores y desarrolladores correctamente descritos, de ahí la vital importancia del uso de una adecuada técnica para la captura de requisitos de manera eficaz, que ayude a obtener una visión correcta del sistema. Las técnicas utilizadas para la captura de requisitos del módulo son las siguientes:

1. Sistemas existentes: esta técnica consiste en analizar distintos sistemas ya desarrollados que estén relacionados con el sistema a ser construido. Se pueden analizar las interfaces de usuario, observando el tipo de información y cómo se maneja. También es útil analizar las distintas salidas que los sistemas producen (listados, consultas), porque siempre pueden surgir nuevas ideas sobre la base de estas. Esto puede ser útil para descubrir requisitos importantes, que tal vez el cliente o el usuario hayan fallado en comunicar.

2. Brainstorming (Lluvia de ideas): una sesión de tormenta de ideas es una reunión de personas interesadas, cuya misión es generar nuevas ideas para el producto final. Esta técnica aprovecha el efecto de grupo, es decir, reúne a un grupo de personas para generar tantas ideas como sea posible para el nuevo producto. Las ideas que se exponen en esta técnica son todas aceptables y siempre debe existir el espacio para criticarlas o debatirlas.

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

Estas tres técnicas combinadas dieron lugar a la captura de los requisitos del módulo. Primeramente, se hizo un estudio de homólogos, donde se realizó un análisis de cómo se realiza la edición y lectura de documentos de formato abierto en la web, evidenciando la técnica de sistemas existentes. Después de realizado el estudio de homólogos, se aplicó la técnica de lluvias de ideas para analizar y generar ideas sobre el desarrollo del módulo propuesto. Por último se utilizó la técnica de prototipo para detallar de forma más clara algunos de los requisitos funcionales identificados, elemento esencial que dio lugar a la obtención de importantes requerimientos funcionales.

2.7 Especificación de requisitos

Requisitos funcionales

Los requisitos funcionales es la definición de los servicios que el sistema debe proporcionar, cómo debe reaccionar a una entrada particular y cómo se debe comportar ante situaciones particulares. Expresan la naturaleza del funcionamiento del sistema (cómo interacciona el sistema con su entorno y cómo van a ser su estado y funcionamiento).

Es por ello que se debe analizar cuáles son las funcionalidades del sistema que cumplan con los objetivos que se definieron, especificando las acciones que la aplicación debe ser capaz de realizar.

RF1 Visualizar documento.

- RF1.1 Visualizar documento PDF.
- RF1.2 Visualizar documento ODF.
- RF1.3 Cambiar tamaño de visualización del documento.
- RF1.4 Navegar entre páginas.

RF2 Editar documento.

RF3 Imprimir documento.

RF4 Descargar documento.

Requisitos no funcionales

Los requisitos no funcionales son restricciones que afectan a los servicios o funciones del sistema, tales como: restricciones de tiempo, sobre el proceso de desarrollo y estándares. Son propiedades o cualidades que el producto debe tener, que lo harán atractivo, usable y rápido. Marcan la diferencia entre un producto bien aceptado y otro con poca aceptación.

A continuación se muestran algunas categorías para clasificar los requisitos no funcionales.

Usabilidad

RNF1: El funcionamiento del sistema será intuitivo y no requerirá de grandes conocimientos para su uso. [Las funcionalidades del sistema para trabajar con los documento son de fácil acceso y visualización.]

RNF2: El sistema será flexible y muy fácil de usar. [El acceso a la interfaz de visualización de los documentos se hace de forma sencilla, a través de un click sobre el documento.]

RNF3: Se actualizará el manual de usuario del eXcriba incorporándole los pasos a seguir para ejecutar las funcionales que ofrece el módulo. [Para una mejor utilización de las funcionalidades que brinda el módulo, se incorporará al manual de usuario del eXcriba, una serie de pasos de cómo hacer uso de ellas.]

Confiabilidad

RNF4: Salvas. [El sistema trabaja sobre una copia temporal del documento descargada del servidor, para que en caso de perder la conexión con el servidor, el usuario pueda continuar trabajando sin interrupciones, y una vez reiniciada la conexión, si es necesario, actualizar el fichero almacenado en el servidor, y el eliminar el temporal.]

Seguridad

RNF5: Control de Acceso. [El sistema permite el acceso a los documentos que se quieren visualizar o editar, siempre y cuando el usuario autenticado tengo los permisos necesarios para acceder al mismo.]

Soporte

RNF6: Plataforma web. [Se debe tener instalado los navegadores Mozilla Firefox 16 o superior y Google Chrome 23 o superior.]

RNF7: Documentación. [Se documentará el sistema con un manual de usuario con el objetivo de explicar su uso y estará disponible como parte del sistema.]

Restricciones de diseño

RNF8: Lenguajes de programación. [Los lenguajes de programación a ser usado para implementar la solución serán PHP versión 5.3 y JavaScript versión 1.5, además de herramientas que se distribuyan bajo licencias libres.]

RNF9: Marco de trabajo. [Se utilizará CodeIgniter 1.7.2 y jQuery 1.3.2 como ayuda el diseño de la interfaz del usuario final.]

RNF10: Servidor web. [Se utilizará servidor web Apache 2.2.]

RNF11: Metodología. [Se utilizará la metodología de desarrollo de *software* RUP con nivel 2 de CMMI, usando el lenguaje de modelación UML 2.0.]

RNF12: Librerías [Las librerías a usar no deben ser propietarias, ejemplo WebODF y PDF.js]

RNF13: Forma de acceso [El sistema podrá ser accedido a través de un navegador web desde los sistemas operativos Windows y Gnu/Linux.]

Requisitos para la documentación de usuarios en línea y ayuda del sistema.

RNF14: Manual de usuario. [El sistema incluirá un manual de usuario para aclarar dudas en cuanto al funcionamiento del mismo.]

Componentes Comprados

RNF15: Componentes. [Todos los componentes del sistema deben ser de código abierto (open-source).]

Interfaz

Interfaces de usuario

RNF16: Navegadores. [Para interactuar con el sistema debe usarse el navegador Mozilla/Firefox en su versión 16 o superior y el Google Chrome en su versión 23 o superior. No se garantiza la correcta visualización en otros navegadores.]

RNF17: Soporte. [Para acceder al sistema el navegador debe tener habilitado el soporte para Java Script.]

Interfaces Hardware

RNF18: Red. [El hardware donde se instalará el sistema debe poseer al menos una Interfaz de red cuya velocidad de transferencia iguale o supere los 100 Mbps.]

Requisitos de Licencia

RNF19: Licencia. [El *software* no debe usar componentes, bibliotecas de clases u otro elemento que posea licencias privativas.]

Requisitos Legales, de Derecho de Autor y otros.

RNF20: Herramientas. [Las herramientas seleccionadas para el desarrollo del producto están respaldadas por licencias libres, bajo las condiciones de *software* libre. A excepción del IDE de desarrollo ZendStudio.]

2.8 Definición de los Casos de Uso

Para la correcta captura de los requisitos identificados anteriormente, se han creado durante esta fase una serie de artefactos, los cuales serán expuestos a continuación:

Definición de los actores:

En el sistema que está siendo modelado solo interactúa un actor denominado “Usuario”, el cual es el encargado de interactuar con el sistema:

Actores	Justificación
Usuario	Es la persona que se encarga de interactuar con el sistema y elegir cuál documento desea visualizar o editar.

Tabla 1 Definición de los actores.

Lista de los Casos de Uso

A continuación una breve descripción de los casos de usos que conforman el sistema.

CU-1	Visualizar documento
Actor	Usuario
Descripción	Permite que el usuario pueda visualizar un documento.
Referencia	RF1, RF1.1, RF1.2, RF1.3, RF1.4

Tabla 2 Breve descripción Caso de Uso 1.

CU-2	Visualizar documento PDF
Actor	Usuario

Descripción	Permite que el usuario pueda visualizar un documento de formato PDF.
Referencia	RF1.1

Tabla 3 Breve descripción Caso de Uso 2.

CU-3	Visualizar documento ODF
Actor	Usuario
Descripción	Permite que el usuario pueda visualizar un documento de formato ODF.
Referencia	RF1.2

Tabla 4 Breve descripción Caso de Uso 3.

CU-4	Editar documento
Actor	Usuario
Descripción	Permite que el usuario edite un documento de formato ODF, una vez visualizado el mismo.
Referencia	RF2

Tabla 5 Breve descripción Caso de Uso 4.

CU-5	Imprimir documento
Actor	Usuario
Descripción	Permite que el usuario imprima un documento, una vez visualizado el mismo.
Referencia	RF3

Tabla 6 Breve descripción Caso de Uso 5.

CU-6	Descargar documento
Actor	Usuario
Descripción	Permite que el usuario descargue un documento, una vez visualizado el mismo.
Referencia	RF4

Tabla 7 Breve descripción Caso de Uso 6.

Diagrama de Casos de Usos

Ver anexo A.1 Figura 5, de la versión extendida de este documento.

Descripción expandida de los Casos de Uso

Ver Anexo B de la versión extendida del documento para observar las otras descripciones de los casos de uso.

CU 1. Visualizar documento

Objetivo	Visualizar un documento dentro del Gestor de Documentos Administrativos eXcriba.	
Actores	**El Usuario:** Inicia el caso de uso, haciendo click sobre el documento que desea visualizar en el explorador de archivos del Gestor de Documentos Administrativos eXcriba.	
Resumen	Debe permitirle al usuario visualizar el documento.	
Complejidad	Alta	
Prioridad	Critico	
Precondiciones	El usuario tiene que estar autenticado en el sistema.	
Postcondiciones		
Flujo de eventos		
Flujo básico <Nombre del flujo básico>		
	Actor	**Sistema**

1.	El usuario selecciona dentro del explorador de archivos del Gestor de Documentos Administrativos eXcriba el que desea visualizar.	
2.		El sistema muestra la visualización del documento seleccionado y las siguientes opciones para interactuar con el mismo: • Cambiar tamaño de visualización del documento. *[Ver Sección 1:”Cambiar tamaño”.]* • Navegar entre páginas. *[Ver Sección 2:”Navegar entre páginas”.]*
3.		Finaliza el evento cuando el usuario salga de la vista que visualiza el documento.
Sección 1: “Cambiar tamaño”		
Flujo básico <Nombre del flujo básico>		
	Actor	**Sistema**
1.	El usuario elige si aumentar o disminuir el tamaño de visualización del documento a través de una lista desplegable con varios tamaños predefinidos o introduciendo el tamaño deseado en un campo de texto.	
2.		El sistema muestra la nueva visualización del documento con el tamaño elegido por el usuario.
Sección 2: “Navegar entre páginas”		
Flujo básico <Nombre del flujo básico>		
	Actor	**Sistema**
1.	El usuario elige la página hacia la cual quiere desplazarse, insertando el número de la página en un cuadro de texto, o desplazándose a la misma a través de una mini-visualización desplegable de las páginas ubicadas a la izquierda de la pantalla.	
2.		El sistema muestra la página elegida por el usuario.

Requisitos no funcionales	RnF 1, RnF 2, RnF 3, RnF7, RnF 20, RnF 21, RnF 22, RnF 23

Tabla 8 Descripción expandida del Caso de Uso 1.

2.9 Conclusiones del capitulo

La realización de este capítulo permitió analizar las principales características del sistema, definir el modelo de dominio para comprender la estructura y la dinámica de los procesos involucrados, plantear los requisitos funcionales y no funcionales del sistema, los actores y casos de usos del sistema, la relación existentes entre ellos, y por último permitió realizar la descripción textual de cada uno de los casos de uso del sistema. De esta manera quedan planteadas, las condiciones y características del sistema propuesto.

Capítulo 3 “Diseño del subsistema”.

3.1 Introducción

En el presente capítulo se modelará el sistema para que soporte todos los requisitos, incluyendo los no funcionales y las restricciones que se le suponen, todo a través de un conjunto de artefactos y diagramas de clases de diseño; dentro de ellos se destacan: el diagrama de clases del diseño y el diagrama de secuencia, además de, la descripción de la arquitectura y los diferentes patrones de diseños utilizados.

3.2 Modelo de diseño

El modelo de diseño es utilizado para modelar los aspectos dinámicos del sistema. Consta de un conjunto de objetos que describen las realizaciones de los casos de uso y sus relaciones, incluyendo además los mensajes que pueden enviarse entre ellos. Se centra en los impactos que producen en el sistema los requisitos funcionales y no funcionales. De manera general el modelo de diseño constituye una abstracción al modelo de implementación y del código fuente, o sea, es la entrada o el punto de partida para las posteriores actividades de implementación del sistema que se desea desarrollar. (Jacobson, 2004)

3.2.1 Diagrama de clases de diseño

Los diagramas de clases son los más utilizados en el modelado de sistemas orientados a objetos. Un diagrama de clases en sí, muestra un conjunto de clases, interfaces y colaboraciones, así como sus relaciones. Los mismos se utilizan para reflejar la vista de diseño estática de un sistema. Son la base para los posteriores diagramas de componentes y los de despliegue. Su importancia radica en que permitirá construir sistemas ejecutables aplicando ingeniería directa e inversa. (Jacobson, 2004)

Ver anexo A.1 Figura 6, de la versión extendida de este documento.

3.2.2 Descripción de las clases

A continuación se hará una descripción de las principales clases que modela el sistema.

Nombre: C_visual_docs

Tipo de clase: Controladora	
Atributo	**Tipo**
$document	
Para cada responsabilidad:	
Nombre:	update_document()
Descripción:	Actualiza el documento en el repositorio documenta, después de editado el mismo.

Tabla 9 Descripción de la clase C_visual_docs.

Nombre: visual_docs	
Tipo de clase: Interfaz	
Atributo	**Tipo**
$pagina	
$texto	
$document	
Para cada responsabilidad:	

Tabla 10 Descripción de la clase visual_docs.

Nombre: Visuals_docs

Tipo de clase: Controladora	
Atributo	**Tipo**
Para cada responsabilidad:	
Nombre:	__construct()
Descripción:	Se captura la acción y se ejecuta el método correspondiente.
Nombre:	print_view()
Descripción:	Se imprime la vista con el documento seleccionado.
Nombre:	get_ajax_context()
Descripción:	Es el encargado de incluir los ficheros JavaScript.

Tabla 11 Descripción de la clase Visual_docs.

3.3 Diagrama de secuencia.

Los diagramas de interacción se utilizan para la preparación de un buen diseño, explican gráficamente las interacciones existentes entre las instancias (las clases). Esta interacción se puede expresar en diagramas de colaboración y de secuencia.

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

Ver anexo A.1 Figura 7, de la versión extendida de este documento.

3.4 Descripción de la arquitectura

La arquitectura de *software* consiste en un conjunto de patrones que proporcionan un marco de referencia necesario para guiar la construcción de un *software*, permitiendo a los programadores, analistas y todo el conjunto de desarrolladores del *software* compartir una misma línea de trabajo y cubrir todos los objetivos y restricciones de la aplicación. (Casanovas, 2013)

3.4.1 Arquitectura en capas

En el desarrollo de la presente solución se hará uso de la arquitectura en capas, producto a que el sistema, al cual se le incorporará el módulo ya utiliza la misma, lo cual constituye una restricción del diseño. Esta arquitectura del *software* va a contar con tres capas, estas son: Presentación, Aplicación y Acceso a repositorio.

Capa de presentación: Dentro de esta capa es donde se encuentran el conjunto de interfaces de usuario, que hará posible establecer la comunicación entre el cliente y la aplicación, así como la manipulación de los datos, y la representación en términos de componentes visuales de toda la información necesaria, consultada y/o generada por el usuario y la aplicación. Las vistas que se diseñarán para el módulo edición y lectura de documentos de formato abierto serán implementadas en esta capa de presentación, basándose en la API de eXcriba para jQuery, la cual está basada en el Framework jQuery, que brinda un conjunto de funcionalidades que facilitan la implementación.

Capa de aplicación: En esta capa se ejecutan todos los procesos de negocio que han sido implementados con antelación, se preparan a su vez las transformaciones de datos, que sirven como un intermediario entre las demandas del cliente y las respuestas de los datos. Además, esta capa controla y dirige el flujo de la aplicación en sentido general y es la encargada de comunicarse con los servicios ubicados en la capa inferior. En el módulo para edición y lectura de documentos de formato abierto se necesita acceder a un conjunto de subsistemas ubicados en esta capa, entre los que se encuentran: Gestión de acciones, Control de acceso, Comunicación con la capa de acceso a repositorio y Gestión de eventos. Estos eventos son los que validan los permisos que tiene el usuario de acceder al documento solicitado, las acciones y eventos que se pueden ejecutar, y los que permiten la comunicación con los servicios que brinda Alfresco. También en esta capa estarán las clases controladoras que se encargarán de manejar la lógica del negocio, implementadas a través del Framework CodeIgniter.

Capa de acceso a repositorio: Esta capa es la encargada de gestionar los datos que se encuentran en el repositorio de archivos a través de la implementación de los servicios o mediante el uso de los servicios que se encuentran implementados en la API de WebScript de Alfresco.

En la siguiente figura se muestra un diagrama con la descripción de la arquitectura del sistema.

Ver anexo A.1 Figura 8, de la versión extendida de este documento.

3.5 Patrones de diseño

Los patrones de diseño son la base para la búsqueda de soluciones a problemas comunes en el desarrollo de *software* y otros ámbitos referentes al diseño de interacción o interfaces.

En la investigación se utilizó la familia de patrones GRASP (*General Responsibility Asignment Software Patterns*) el cual describe los principios fundamentales de la asignación de responsabilidades a objetos, expresados en forma de patrones. (Larman, 2004)

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

Ejemplo 1: La aplicación de este patrón se evidencia en la clase x_aplication_helper la cual tiene la información necesaria para hacer la llamada al servicio.

Ejemplo 2: La aplicación de este patrón se evidencia en la clase x_explorer_helper la cual tiene la información necesaria para generar el explorador del eXcriba.

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

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

Ejemplo 1: La aplicación de este patrón se evidencia en la clase visual_docs_odf la cual tiene las mismas funciones básicas de las librerías del sistema pero los métodos tendrán diferentes implementaciones. Lo que evidencia la poca dependencia entre las clases.

Ejemplo 2: La aplicación de este patrón se evidencia en la clase visual_docs_pdf la cual tiene las mismas funciones básicas de las librerías del sistema pero los métodos tendrán diferentes implementaciones. Lo que evidencia la poca dependencia entre las clases.

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

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

Ejemplo 1: La aplicación de este patrón se evidencia en la clase visual_docs_odf la cual aunque cuenta con diferentes implementaciones de sus métodos, tiene las mismas funciones básicas de las librerías del sistema. Lo que evidencia como están relacionadas entre sí las clases y a su vez tienen diferentes implementaciones, además esto permite una reutilización del código.

Ejemplo 2: La aplicación de este patrón se evidencia en la clase visual_docs_pdf la cual aunque cuenta con diferentes implementaciones de sus métodos, tiene las mismas funciones básicas de las librerías del sistema. Lo que evidencia como están relacionadas entre sí las clases y a su vez tienen diferentes implementaciones, además esto permite una reutilización del código.

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

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

Ejemplo 1: La aplicación de este patrón se evidencia en la clase x_visual_docs la cual está encargada de realizar las operaciones del sistema y está ubicada en la capa del dominio.

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

3.6 Conclusiones del capítulo

Con la conclusión del capítulo se han dado los primeros pasos necesarios para comenzar la implementación del sistema. Esto ha sido posible mediante la transformación paulatina de los requisitos, tanto funcionales como no funcionales a una especificación que describa como implementar el sistema. En el transcurso del capítulo se ha obtenido una visión sobre lo que debe hacer el sistema a desarrollar, centrándose en los requisitos funcionales y por otro lado cómo el sistema debe cumplir con los objetivos propuestos, enfocándose así en los requisitos no funcionales. Una vez concluida esta etapa puede decirse que se ha refinado y definido la arquitectura del sistema lo cual permitirá adaptar dicho diseño para que sea consistente con el entorno de implementación.

Capítulo 4 “Implementación y pruebas del módulo propuesto”.

4.1 Introducción

En este capítulo se especifica cómo los elementos de diseño se implementan en términos de componentes y se organizan en nodos específicos en el diagrama de despliegue, además se describen las diferentes pruebas realizadas al sistema a lo largo del ciclo de vida del producto. Otros de los propósitos de la implementación es desarrollar la arquitectura y el sistema como un todo.

4.2 Diagrama de despliegue

El modelo de despliegue es utilizado para capturar los elementos de configuración del procesamiento y las diferentes conexiones existentes entre ellos. El mismo incluye un artefacto fundamental, el diagrama de despliegue, usado para visualizar la distribución de los componentes de *software* en los nodos físicos.

Ver anexo A.1 Figura 9, de la versión extendida de este documento.

El diagrama de despliegue del *software* eXcriba cuenta con tres nodos. El primero es ordenador cliente, que es el nodo donde se localizarán las estaciones de trabajo que el usuario utilizará para acceder a la aplicación, a través de un navegador web. Como segundo nodo se tiene gestor documental, el cual hace referencia al núcleo del sistema, donde reside la lógica de la aplicación. Para lograr la conexión del sistema con el ordenador cliente se utiliza el protocolo de comunicación HTTP. Dentro del nodo del gestor documental residen dos servidores web: apache que contiene el cliente web de eXcriba 2.0 y apache tomcat que contiene el ECM Alfresco Labs 3.0 Stable. El último nodo es el servidor de bases de datos, que es donde se encuentra almacenada toda la información gestionada por el Alfresco. Este servidor de base de datos está especificado por el gestor PostgreSQL.

4.3 Diagrama de componentes

El diagrama de componentes muestra un conjunto de componentes relacionados entre sí. Su uso principal es estructurar el modelo de implementación en términos de subsistemas de implementación y mostrar las relaciones de los elementos de implementación, para modelar la vista estática del sistema.

Ver anexo A.1 Figura 10: Diagrama de componentes, de la versión extendida de este documento.

4.4 Estándar de codificación

Los estándares de codificación son un complemento a la programación, cuyo propósito es que el código fuente tenga una arquitectura y un estilo consistente con lo cual se pueda facilitar su lectura y modificación por cualquier miembro del equipo de desarrollo. (Zend, 2013)

Al comenzar un proyecto de *software* se debe establecer un estándar de codificación para que todos los implicados trabajen de forma coordinada. En cada grupo de desarrollo se definen cuáles serán los aspectos a estandarizar y qué estilos se aplicarán a cada uno de ellos. Dichos estándares empleados para el desarrollo del módulo son los propuestos por el arquitecto del GDA eXcriba que son los que establece CodeIgniter versión 1.7.2. (Griffiths, 2010)

Ver anexo C.1 de la versión extendida de este documento.

4.5 Pruebas de *software*

Un elemento fundamental para garantizar la calidad de un sistema son las pruebas del *software*. Estas pruebas verifican que el *software* funcione como se diseñó y que los requerimientos son satisfechos, además de brindar soporte para encontrar y documentar defectos del sistema. (Pressman, 2005)

4.5.1 Criterios de validación de requisitos

La validación del *software* se logra mediante una serie de pruebas que demuestran que se cumple con los requisitos. (Pressman, 2005) Para validar los requisitos obtenidos en el módulo para la gestión de metadatos en el eXcriba se utilizan criterios propuestos por el proceso de mejora basado en el segundo nivel de CMMI que se lleva a cabo en la UCI para la metodología RUP, dichos criterios contienen varias interrogantes que permiten validar si el requisito puede o no aprobarse, estas interrogantes se exponen a continuación. Para ver detalladamente estos artefactos, consultar el expediente del proyecto eXcriba.

- ¿El proveedor del requisito es un proveedor válido?
- ¿El requisito está identificado como único?
- ¿El requisito es modificable?
- ¿El requisito no es ambiguo?

- ¿El requisito está completo?
- ¿El requisito es congruente con otros requisitos relacionados?
- ¿El requisito puede ser implementado?
- ¿El requisito puede ser probado?
- ¿El resultado de la evaluación de impacto es positivo?
- ¿El requisito está correcto?
- ¿El requisito es traceable?

Al aplicarse estos criterios sobre los requisitos del módulo se obtuvo como resultado el 100 % de requisitos aprobados.

4.5.2 Técnicas de validación de requisitos

En Pressman se definen un grupo de técnicas que permiten que el proceso de validación tenga una mejor calidad, de ellas se utilizaron: (Pressman, 2005)

- **Matriz de trazabilidad:** se realizó una matriz de trazabilidad para comprender cómo cada uno de los diferentes casos de uso con que cuenta el sistema satisfacen todos los requerimientos, y a su vez analizar la vida de un requerimiento tanto hacia atrás como hacia delante durante todo el ciclo de vida de un proyecto.
- **Construcción de prototipos:** se mostró un modelo ejecutable del sistema a los clientes para que los mismos interactuaran con este y determinaran si cumplía o no con sus necesidades.
- **Generación de casos de prueba:** se realizaron los diseños de casos de pruebas para cada uno de los requisitos especificados, que permitieron verificar que todos se pudieran probar; e identificar, en dependencia de la complejidad del diseño de caso de prueba, requisitos que deberían ser reconsiderados y aquellos más difíciles de implementar.

4.5.3 Resultado de la revisión de los requerimientos

Durante la revisión de requerimientos se realizaron verificaciones en el documento “Especificaciones de requisitos”, este proceso comprendió las:

- **Verificaciones de validez:** los requisitos deben cumplir con las necesidades del cliente. Luego de realizar algunos análisis y razonamientos surgieron funciones adicionales y cambios en las que ya estaban identificadas.

- **Verificaciones de consistencia:** los requisitos no deben contradecirse con las especificaciones escritas, no deben haber restricciones o descripciones que estén opuestas a las reglas definidas.

- **Verificaciones de completitud:** los requerimientos deben incluir todas las funcionalidades propuestas por el cliente, satisfacer de manera general todas las necesidades acordadas.

- **Verificaciones de realismo:** a partir del conocimiento previo de la tecnología se realizó una revisión de los requisitos para identificar su posible implementación.

- **Verificabilidad:** para evitar posibles discrepancias entre los miembros del equipo del proyecto y el cliente se revisó que los requisitos estuvieran descritos de manera que puedan ser verificables, o sea que se puedan diseñar casos de pruebas orientadas a estos y que demuestren que el sistema a entregar responde a las necesidades del cliente.

Al concluir el proceso de revisión se detectaron algunas inconsistencias que fueron erradicadas de inmediato. Entre las más comunes se pueden citar:

- Se interpretaron de forma incorrecta algunas de las funcionalidades y características solicitadas por el cliente.

- Descripciones poco detalladas de algunos de los requisitos.

- Errores ortográficos.

4.5.4 Prueba de caja blanca

Dentro de los métodos de diseño de casos de prueba se encuentran las pruebas de caja blanca que utilizan la estructura de control de diseño procedimental para obtener los casos de prueba. Es mediante los métodos de caja blanca que se garantiza que se ejerciten al menos una vez todos los caminos independientes de cada módulo y todas las decisiones lógicas en sus vertientes verdaderas y falsa.

Además de que se ejecuten todos los bucles en sus límites y se ejerciten las estructuras internas de datos para asegurar su validez. (Pressman, 2005)

Existen varios tipos de prueba de caja blanca entre los que se destacan:

Prueba de condición: es un método de diseño de casos de prueba que ejercita las condiciones lógicas contenidas en el módulo de un programa.

Prueba de flujo de datos: se seleccionan caminos de prueba de un programa de acuerdo con la ubicación de las definiciones y los usos de las variables del programa.

Prueba de bucles: comprueba la validez de las construcciones de los bucles.

Prueba del camino básico: permite obtener una medida de la complejidad lógica de un diseño.

Para comprobar la precisión del código se seleccionó la prueba de camino básico. Esta permitirá comprobar la complejidad dicromática de los métodos del módulo y verificar que todos los caminos se ejecutaban de forma correcta. A continuación se muestran los pasos realizados para validar el correcto funcionamiento el método update_document.

La siguiente figura contiene la representación del código fuente del algoritmo update_document().

Ver anexo A.1 Figura 11, de la versión extendida de este documento.

La próxima figura muestra el diagrama del flujo asociado al algoritmo update_document().

Ver anexo A.1 Figura 12, de la versión extendida de este documento.

Fórmulas para calcular la complejidad ciclomática:

V (G) = (A – N) + 2

V (G) = (11 - 9) + 2

V (G) = 4

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

V (G) = P + 1

V (G) = 3 + 1

V (G) = 4

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

El resultado obtenido mediante las fórmulas anteriores representa los posibles caminos por los que transitará el flujo, así como la cantidad mínima de casos de pruebas que se deben realizar para el procedimiento escogido. A continuación se representa los caminos básicos que se identificaron en el flujo:

Camino 1	1, 9
Camino 2	1, 2, 4, 9
Camino 3	1, 2, 3, 5, 7, 8, 9
Camino 4	1, 2, 3, 5, 6, 5, 7, 8, 9

Tabla 12 Caminos básicos.

Se procede a realizar los casos de pruebas, teniendo en cuenta que se debe realizar al menos un caso de prueba por cada camino básico identificado:

Casos de prueba para el camino básico 1

Camino 1: [1, 9]

Descripción: los datos entrantes permiten la actualización del documento editado en el repositorio documental.

Entrada: $uuid =”c734sdfd-de33f787”.

Resultados esperados: si ocurre algún error dentro del algoritmo el sistema captura el mismo y muestra un mensaje de información de dicho error.

Casos de prueba para el camino básico 2

Camino 2: [1, 2, 4, 9]

Descripción: los datos entrantes permiten la actualización del documento editado en el repositorio documental.

Entrada: $uuid =”c734sdfd-de33f787”.

Resultados esperados: si existe algún error en la actualización del documento el sistema muestra el siguiente mensaje “Ha ocurrido un problema actualizando el documento”.

Casos de prueba para el camino básico 3

Camino 3: [1, 2, 3, 5, 7, 8, 9]

Descripción: los datos entrantes permiten la actualización del documento editado en el repositorio documental.

Entrada: $uuid =”c734sdfd-de33f787”.

Resultados esperados: si no existe ningún error en la actualización del documento el sistema muestra el siguiente mensaje “El documento ha sido actualizado correctamente”.

Casos de prueba para el camino básico 4

Camino 4: [1, 2, 3, 5, 6, 5, 7, 8, 9]

Descripción: los datos entrantes permiten la actualización del documento editado en el repositorio documental.

Entrada: $uuid =”c734sdfd-de33f787”.

Resultados esperados: si no existe ningún error en la actualización del documento el sistema muestra el siguiente mensaje “El documento ha sido actualizado correctamente”.

4.5.5 Prueba de caja negra

Las pruebas de caja negra se centran principalmente en los requisitos funcionales del *software*. Estas pruebas permiten obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. Intentando encontrar errores dentro de las siguientes categorías:

funciones incorrectas o ausentes; errores de interfaz, en estructuras de datos o en acceso a bases de datos externas; errores de rendimiento, de inicialización y de terminación. (Pressman, 2005)

Para comprobar la calidad de la solución planteada, se empleó el método de caja negra aplicándose la técnica de partición equivalente, pues permite examinar los valores válidos e inválidos de las entradas existentes en el *software*, además se descubre de forma inmediata los errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico, y además la utilización de esta técnica permitirá reducir el número de clases de prueba que hay que desarrollar. (Pressman, 2005)

A continuación se presentan y describen los casos de prueba desarrollados para cada casos de uso definido, especificando la información de entrada, los resultados obtenidos una vez ejecutado el caso de prueba y las condiciones que deben cumplirse para que este se ejecute.

Escenario Visualizar documento

Entrada	Resultado	Condiciones
El usuario elige dentro del explorador del eXcriba el documento que desea visualizar.	El sistema visualiza el documento elegido por el usuario.	El usuario tiene que estar autenticado en el sistema. El documento tiene que estar almacenado dentro del repositorio documental.

Tabla 13 Escenario visualizar documento.

Escenario Visualizar documento PDF

Entrada	Resultado	Condiciones
El usuario elige dentro del explorador del eXcriba el documento con formato PDF que desea visualizar.	El sistema visualiza el documento con formato PDF elegido por el usuario.	El usuario tiene que estar autenticado en el sistema. El documento con formato PDF tiene que estar almacenado dentro del repositorio documental.

Tabla 14 Escenario visualizar documento PDF.

Escenario Visualizar documento ODF

Entrada	Resultado	Condiciones
El usuario elige dentro del explorador del eXcriba el	El sistema visualiza el documento con formato ODF elegido por el	El usuario tiene que estar autenticado en el sistema. El documento con formato ODF tiene

documento con formato ODF que desea visualizar.	usuario.	que estar almacenado dentro del repositorio documental.

Tabla 15 Escenario visualizar documento ODF.

Escenario Editar documento

Entrada	Resultado	Condiciones
El usuario elige dentro del explorador del eXcriba el documento que desea editar.	El sistema visualiza el documento elegido por el usuario y muestra las acciones predeterminadas que el usuario podrá realizar sobre el documento.	El usuario tiene que estar autenticado en el sistema. El documento tiene que estar almacenado dentro del repositorio documental.

Tabla 16 Escenario editar documento.

Escenario Descargar documento

Entrada	Resultado	Condiciones
El usuario elige dentro de la vista de visualización del documento la acción descargar documento.	El sistema muestra una ventana donde el usuario elige la ubicación y el nombre del documento que el usuario desea descargar.	El usuario tiene que estar autenticado en el sistema. El documento tiene que estar almacenado dentro del repositorio documental.

Tabla 17 Escenario descargar documento.

Escenario Imprimir documento

Entrada	Resultado	Condiciones
El usuario elige dentro de la vista de visualización del documento la acción imprimir documento.	El sistema muestra una ventana con las opciones de impresión del documento.	El usuario tiene que estar autenticado en el sistema. El documento tiene que estar almacenado dentro del repositorio documental.

Tabla 18 Escenario imprimir documento.

Una vez que se realizan las pruebas de caja negra, se obtienen los resultados siguientes:

No conformidades	Iteración 1	Iteración 2	Iteración 3

Alta	-	-	-
Baja	4	3	-

Tabla 19 Resultados de las pruebas de caja negra.

En el flujo de prueba se realizaron las descripciones de los casos de prueba pertenecientes a los casos de uso documentados en el trabajo, para comprobar las funcionalidades del módulo; las pruebas realizadas se llevaron a cabo en tres iteraciones, en la primera iteración se detectaron cuatro no conformidades, todas ellas fueron bajas (no significativas); en la segunda iteración se detectaron tres no conformidades todas ellas baja(no significativas) que fueron resueltas inmediatamente, en la última iteración ya no se encontraron no conformidades, por lo tanto se puede concluir que las pruebas de funcionalidad en el módulo fueron realizadas satisfactoriamente.

4.6 Conclusiones del capítulo

Con la modelación de los diagramas de componentes se logró tener una vista general de todas las dependencias y funcionalidades necesarias para el correcto funcionamiento del módulo. Se implementaron las clases y objetos en ficheros de código fuente y ejecutables, dando como resultado final un módulo para la edición y lectura de documentos de formato abierto en el GDA eXcriba. La realización de pruebas a la solución, mediante la selección de las técnicas de caja negra y caja blanca permitió corregir algunos errores detectados en la implementación, logrando que dicha solución cumpla con las necesidades del cliente Los resultados arrojados por las pruebas realizadas, permitieron la validación de la calidad, eficiencia y correcto funcionamiento del módulo.

Conclusiones

A través del análisis del proceso de edición y lectura de documentos de formato abierto desde el Gestor de Documentos Administrativos eXcriba y el estudio de diversos sistemas, que permitieron observar el desarrollo y evolución de la edición y lectura de documento *online*, se expuso la necesidad de implementar un módulo para la edición y lectura de documentos de formato abierto desde el cliente web del eXcriba, que permitió arribar a las siguientes conclusiones:

Los documentos constituyen un aspecto importante dentro de la gestión documental, ya que ellos, como soporte de información, son la base sobre la que se sustenta la gestión documental, de ahí que la edición y lectura de los mismos debe realizarse a través de un proceso rápido y eficiente que permita el ahorro de tiempo en las tareas de oficinas.

La implementación del módulo propuesto permitió agilizar el proceso de edición y lectura de documentos de formato abierto desde el eXcriba, y eliminar las dependencias de *software* ofimáticos.

La utilización de la técnica partición equivalente y la prueba de camino básico permitieron validar el correcto funcionamiento del módulo.

Recomendaciones

Mejorar la implementación de la edición de documentos, basándose en futuras implementaciones de la API WebODF, que permita realizar a los usuarios una edición avanzada de los documentos de formato abierto, almacenados en el repositorio documental.

Referencias bibliográficas

Abson, Will. Media Viewers. [En línea]. 2013. [Consultado el: 24 de mayo de 2013.]. Disponible en: http://code.google.com/p/share-extras/wiki/MediaViewers.

Aldaz, María José. ECM como estrategia empresarial. [En línea]. 2013. [Consultado el: 8 de febrero de 2013.]. Disponible en: http://archivistica.blogspot.com/2012/06/ecm-como-estrategia-empresarial.html.

Angel, Miguel. Manual de jQuery. [En línea]. 2013. [Consultado el: 6 de febrero de 2013.]. Disponible en: http://www.desarrolloweb.com/manuales/manual-jquery.html.

Aulke, Gaylord. Zend Studio y Zend Platform en un entorno de equipo. s.l. Biblioteca Zend Technologies, 2007.

Bhandari, Amita. Alfresco Share. Birmingham, UK. Packt Publishing Ltd. 2012. ISBN 978-1-84951-710-2.

Casanovas, Josep. Usabilidad y arquitectura del software. [En línea]. 2013. [Consultado el: 19 de 3 de 2013.]. Disponible en: http://www.desarrolloweb.com/articulos/1622.php.

Codina, Lluís. OpenOffice y el formato OpenDocument. [En línea]. 2008. [Consultado el: 22 de mayo de 2013.]. Disponible en: http://www.lluiscodina.com/ooODF.pdf.

Docs, Google. Una Introducción a Google Docs. [En línea]. 2013. [Consultado el: 5 de febrero de 2013.]. Disponible en: http://support.google.com/drive/bin/answer.py?hl=es&answer=49008.

EEES, Alfin. Aplicaciones Ofimáticas. [En línea]. 2013. [Consultado el: 8 de febrero de 2013.]. Disponible en: http://www.mariapinto.es/alfineees/ofimatica/mapa.htm.

Eguíluz, Javier. Introducción a JavaScript. [En línea]. 2009. [Consultado el: 19 de noviembre de 2012.]. Disponible en: http://www.librosweb.es/javascript.

Gallego, José Antonio. Desarrollo Web con PHP y MySQL. Madrid. Anaya Multimedia (Grupo Anaya). 2003. ISBN: 84-415-1525-5.

Graells, Miquel. Estudio de la accesibilidad de los documentos científicos en soporte digital. [En línea]. Revista Española de Documentación Científica. 2008. [Consultado el: 25 de septiembre de 2012.]. Disponible en: http://redc.revistas.csic.es/index.php/redc/article/viewPDFInterstitial/443/455.

Griffiths, Adam. CodeIgniter 1.7 Professional Development. Birmingham, B27 6PA, UK. Packt Publishing Ltd. 2010. ISBN 978-1-849510-90-5.

IAGP. Metodologías de desarrollo de software. [En línea] 2005/2006. [Consultado el: 6 de febrero de 2013.]. Disponible en: http://www.um.es/docencia/barzana/IAGP/Iagp2.html.

Jacobson, Ivar. El Proceso Unificado de Desarrollo de Software. La Habana. Félix Varela. 2004. ISBN: 84-7829-036-2.

Larman, Craig. UML y Patrones Introducción al análisis y diseño orientado a objetos. La Habana. Félix Varela. 2004. ISBN: 842-053-438-2.

Leurs, Laurens. La historia del PDF. [En línea]. 2001. [Consultado el: 4 de febrero de 2013.]. Disponible en: http://gusgsm.com/que_es_el_formato_pdf.

Management, Enterprise Content. Enterprise Content Management ¿Qué es?. [En línea]. 2013. [Consultado el: 5 de febrero de 2013.]. Disponible en: http://herramientasempresariales.com.mx/2011/08/enterprise-content-management-%C2%BFque-es/.

Martín, Miguel. Manual OpenOffice. [En línea]. 2013. [Consultado el: 4 de febrero de 2013.]. Disponible en: http://www.legistdf.gov.ar/servicios/varios/Manual%20OpenOffice.pdf.

MoReq. Model Requirements for the Management of Electronic Records. Bruselas - Luxemburgo. s.n., 2001.

Paradigm, Visual. Visual paradigm for UML user's guide. [En línea]. 2013. [Consultado el: 20 de 1 de 2013.]. Disponible en: http://www.visual-paradigm.com/support/documents/vpumluserguide.jsp.

Pitchin, Chris. Mozilla hacks and suchlike. [En línea]. 2013. [Consultado el: 5 de 2 de 2013.]. Disponible en: http://blog.mozilla.org/cjones/2011/06/15/overview-of-pdf-js-guts/.

Pressman, Roger S. Ingeniería del Software Un enfoque práctico. La Habana. Félix Varela. 2005. ISBN: 970-105-473-3.

Puigpinos, Julio. Formatos Abiertos (1ª Parte). [En línea]. 2013. [Consultado el: 22 de 5 de 2013.]. Disponible en: http://www.ant.org.ar/revista/num0/formatos%20abiertos.pdf.

Ramirez, Neri. Conceptos Básicos: La Ofimática. [En línea]. 2012. [Consultado el: 8 de 2 de 2012.]. Disponible en: http://mitecnologia-neri.blogspot.com/2012/03/conceptos-basicos-la-ofimatica.html.

Rodriguez, Alex. RESTful Web services: The basics. [En línea] 2008. [Consultado el: 6 de 2 de 2013.]. Disponible en: http://www.ibm.com/developerworks/webservices/library/ws-restful/.

Sáez, F. Ofimática Compleja. [En línea]. 1990. [Consultado el: 6 de 2 de 2013.]. Disponible en: http://www.ecured.cu/index.php/Ofim%C3%A1tica.

Shariff, Munwar. Alfresco 3 Web Content Management. Birmingham, UK. Packt Publishing Ltd. 2010. ISBN 978-1-847198-00-6.

Valdés, María. Análisis conceptual de las principales interacciones entre la gestión de información, la gestión documental y la gestión del conocimiento. La Habana. ACIMED. 2008. ISSN 1561-2880.

WebODF. Acerca de nosotros: WebODF. [En línea]. 2013. [Consultado el: 5 de 2 de 2013.]. Disponible en: http://webodf.org/about/.

Weir, Rob. OASIS Open Document Format for Office Applications (OpenDocument) TC. [En línea]. 2012. [Consultado el: 19 de noviembre de 2012.]. Disponible en: https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=office.

Zend. Guía para Implementar Estándares de Codificación. Versión 1.0. [En línea]. 2013. [Consultado el: 19 de abril de 2013.]. Disponible en: http://framework.zend.com/manual/1.12/en/coding-standard.coding-style.html.

Bibliografía

Abson, Will. Media Viewers. [En línea]. 2013. [Consultado el: 24 de mayo de 2013.]. Disponible en: http://code.google.com/p/share-extras/wiki/MediaViewers.

Aldaz, María José. ECM como estrategia empresarial. [En línea]. 2013. [Consultado el: 8 de febrero de 2013.]. Disponible en: http://archivistica.blogspot.com/2012/06/ecm-como-estrategia-empresarial.html.

Angel, Miguel. Manual de jQuery. [En línea]. 2013. [Consultado el: 6 de febrero de 2013.]. Disponible en: http://www.desarrolloweb.com/manuales/manual-jquery.html.

Aulke, Gaylord. Zend Studio y Zend Platform en un entorno de equipo. s.l. Biblioteca Zend Technologies, 2007.

Bhandari, Amita. Alfresco Share. Birmingham, UK. Packt Publishing Ltd., 2012. ISBN 978-1-84951-710-2, 2012.

Casanovas, Josep. Usabilidad y arquitectura del software. [En línea]. 2013. [Consultado el: 19 de 3 de 2013.]. Disponible en: http://www.desarrolloweb.com/articulos/1622.php.

Codina, Lluís. OpenOffice y el formato OpenDocument. [En línea]. 2008. [Consultado el: 22 de mayo de 2013.]. Disponible en: http://www.lluiscodina.com/ooODF.pdf.

Eisenberg, David. OASIS OpenDocument Essentials. [En línea]. 2013. [Consultado el: 16 de febrero de 2013.]. Disponible en: http://books.evc-cit.info/.

Fernández, Luis. Gestión Documental. En Sociedad de la Información. [En línea]. 2007. [Consultado el: 14 de marzo de 2013.]. Disponible en: http://www.sociedadelainformacion.com/12/Gestion %20Documental.pdf. 12.

Gallego, José Antonio. Desarrollo Web con PHP y MySQL. Madrid. Anaya Multimedia (Grupo Anaya). 2003. ISBN: 84-415-1525-5.

Graells, Miquel. Estudio de la accesibilidad de los documentos científicos en soporte digital. [En línea]. Revista Española de Documentación Científica. 2008. [Consultado el: 25 de septiembre de 2012.]. Disponible en: http://redc.revistas.csic.es/index.php/redc/article/viewPDFInterstitial/443/455.

Griffiths, Adam. CodeIgniter 1.7 Professional Development. Birmingham, B27 6PA, UK. Packt Publishing Ltd. 2010. ISBN 978-1-849510-90-5.

Hernández, Rolando. El proceso de investigación científica. La Habana. Universitaria. 2011. ISBN: 9789591913073. 110p.

Hernández, Roberto. Metodología de la investigación. México D.F. McGraw-Hill : 4ta Edición. 2006. ISBN: 9701036322.

IAGP. Metodologías de desarrollo de software. [En línea] 2005/2006. [Consultado el: 6 de febrero de 2013.]. Disponible en: http://www.um.es/docencia/barzana/IAGP/Iagp2.html.

Jacobson, Ivar. El Proceso Unificado de Desarrollo de Software. La Habana. Félix Varela. 2004. ISBN: 84-7829-036-2.

Laporte, Marc. Acerca de nosotros: WebODF. [En línea]. 2013. [Consultado el: 5 de febrero de 2013.]. Disponible en: http://webodf.org/about/.

Larman, Craig. UML y Patrones Introducción al análisis y diseño orientado a objetos. La Habana. Félix Varela. 2004. ISBN: 842-053-438-2.

Leurs, Laurens. La historia del PDF. [En línea]. 2001. [Consultado el: 4 de febrero de 2013.]. Disponible en: http://gusgsm.com/que_es_el_formato_pdf.

Management, Enterprise Content. Enterprise Content Management ¿Qué es?. [En línea]. 2013. [Consultado el: 5 de febrero de 2013.]. Disponible en: http://herramientasempresariales.com.mx/2011/08/enterprise-content-management-%C2%BFque-es/.

Mansfield, Ron. Guia completa para Office de Microsoft. México, D.F. Ventura. 2006. ISBN 968-7393-01-7..

Martin. Suite Ofimática. [En línea]. 2013. [Consultado el: 8 de 2 de 2013.]. Disponible en: http://martincipriruiz.blogspot.com/2011/11/tarea-9-suite-ofimatica_08.html.

Martín, Miguel. Manual OpenOffice. [En línea]. 2013. [Consultado el: 4 de febrero de 2013.]. Disponible en: http://www.legistdf.gov.ar/servicios/varios/Manual%20OpenOffice.pdf.

MoReq. Model Requirements for the Management of Electronic Records. Bruselas - Luxemburgo. s.n., 2001.

Nieto, Iván. Curso de JavaScript. [En línea]. 2009. [Consultado el: 20 de 3 de 2013.]. Disponible en: http://www.elcodigo.net/tutoriales/javascript/javascript1.

Paradigm, Visual. Visual paradigm for UML user's guide. [En línea]. 2013. [Consultado el: 20 de 1 de 2013.]. Disponible en: http://www.visual-paradigm.com/support/documents/vpumluserguide.jsp.

Pitchin, Chris. Mozilla hacks and suchlike. [En línea]. 2013. [Consultado el: 5 de 2 de 2013.]. Disponible en: http://blog.mozilla.org/cjones/2011/06/15/overview-of-pdf-js-guts/.

Pressman, Roger S. Ingeniería del Software Un enfoque práctico. La Habana. Félix Varela. 2005. ISBN: 970-105-473-3.

Puigpinos, Julio. Formatos Abiertos (1ª Parte). [En línea]. 2013. [Consultado el: 22 de 5 de 2013.]. Disponible en: http://www.ant.org.ar/revista/num0/formatos%20abiertos.pdf.

Ramirez, Neri. Conceptos Básicos: La Ofimática. [En línea]. 2012. [Consultado el: 8 de 2 de 2012.]. Disponible en: http://mitecnologia-neri.blogspot.com/2012/03/conceptos-basicos-la-ofimatica.html.

Rodriguez, Alex. RESTful Web services: The basics. [En línea] 2008. [Consultado el: 6 de 2 de 2013.]. Disponible en: http://www.ibm.com/developerworks/webservices/library/ws-restful/.

Sáez, F. Ofimática Compleja. [En línea]. 1990. [Consultado el: 6 de 2 de 2013.]. Disponible en: http://www.ecured.cu/index.php/Ofim%C3%A1tica.

Shariff, Munwar. Alfresco 3 Web Content Management. Birmingham, UK. Packt Publishing Ltd. 2010. ISBN 978-1-847198-00-6.

Tenzer, Simón. Archivos, formatos y extensiones. [En línea]. 2007. [Consultado el: 19 de 11 de 2013.]. Disponible en: http://www.ccee.edu.u/Fensenian/catcomp/material/ArchivosFormatosExtensiones.pdf.

Valdés, María. Análisis conceptual de las principales interacciones entre la gestión de información, la gestión documental y la gestión del conocimiento. La Habana. ACIMED. 2008. ISSN 1561-2880.

WebODF. Acerca de nosotros: WebODF. [En línea]. 2013. [Consultado el: 5 de 2 de 2013.]. Disponible en: http://webodf.org/about/.

Weir, Rob. OASIS Open Document Format for Office Applications (OpenDocument) TC. [En línea]. 2012. [Consultado el: 19 de noviembre de 2012.]. Disponible en: https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=office.

Zend. Guía para Implementar Estándares de Codificación. Versión 1.0. [En línea]. 2013. [Consultado el: 19 de abril de 2013.]. Disponible en: http://framework.zend.com/manual/1.12/en/coding-standard.coding-style.html.

Glosario de términos

A

API (*Application Programming Interface*): Es una llave de acceso a funciones que permiten hacer uso de un servicio web provisto por un tercero, dentro de una aplicación web propia, de manera segura.

C

CASE *Computer Aided Software Engineering* (Ingeniería de Software Asistida por Ordenador): aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de *software* reduciendo el coste de las mismas en términos de tiempo y de dinero.

F

Fiabilidad: probabilidad de buen funcionamiento de algo.

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

G

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

H

***Hardware*:** dispositivos que están conectados físicamente al ordenador.

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

I

IDE (*Integrate Development Enviroment*): entorno de desarrollo integrado. Herramienta que se usa para facilitar el desarrollo de software.

S

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

U

UCI: Universidad de las Ciencias Informáticas.

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

Anexo A

Primer Apéndice

A.1. Figura 1: Principales conceptos de la Ofimática

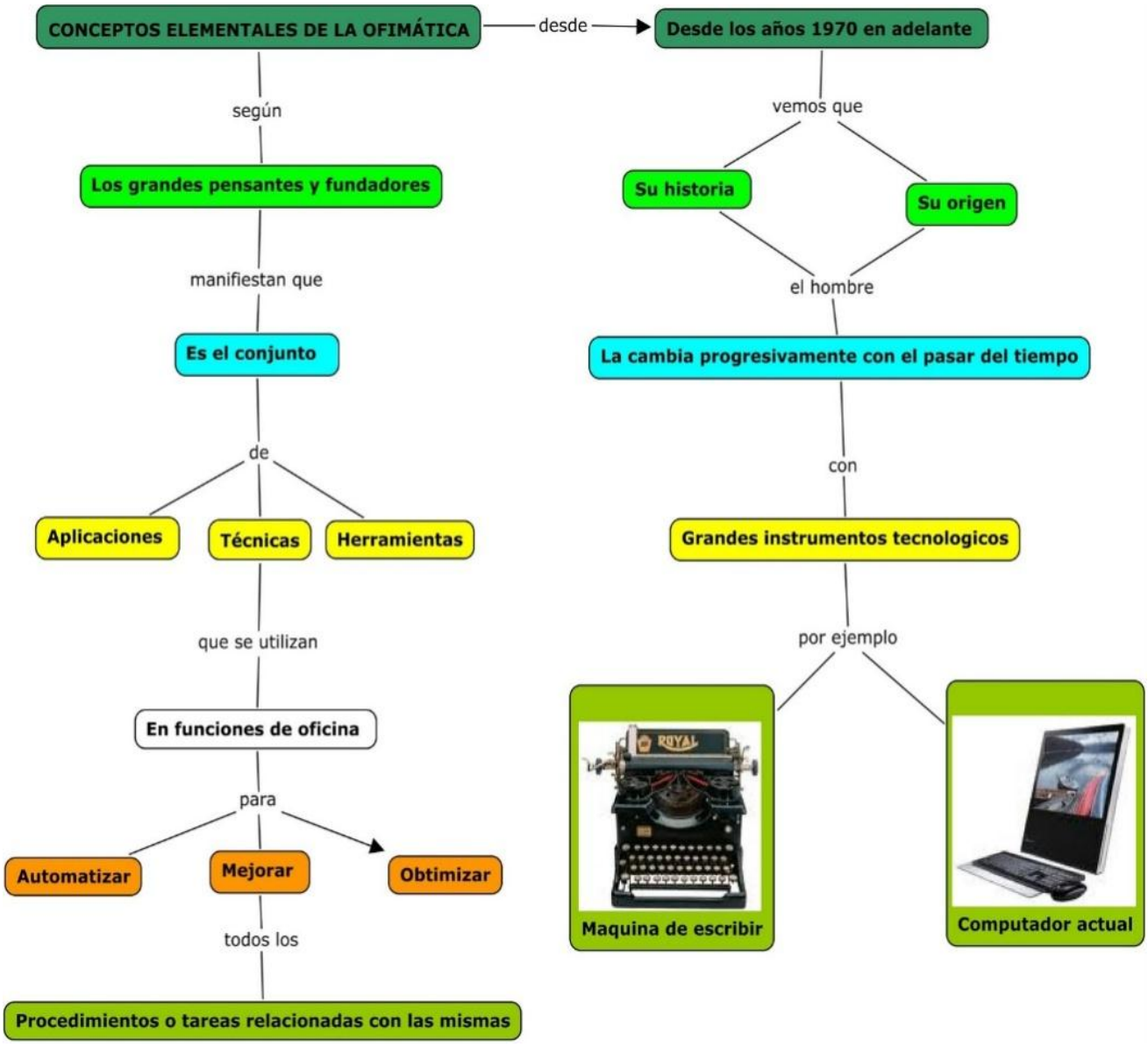

Figura 1 Principales conceptos de la Ofimática. (Ramirez, 2012)

A.1. Figura 2: Tipos de aplicaciones ofimáticas

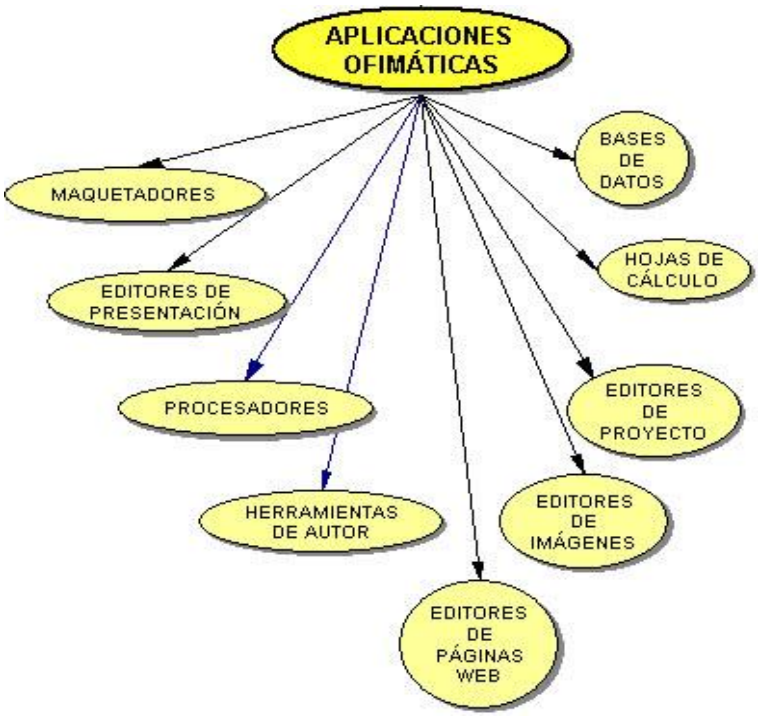

Figura 2 Tipos de aplicaciones ofimáticas. (EEES, 2013)

A.1. Figura 3: Principales elementos de los ECM

Figura 3 Principales elementos de los ECM. (Aldaz, 2013)

A.1. Figura 4: Modelo de dominio

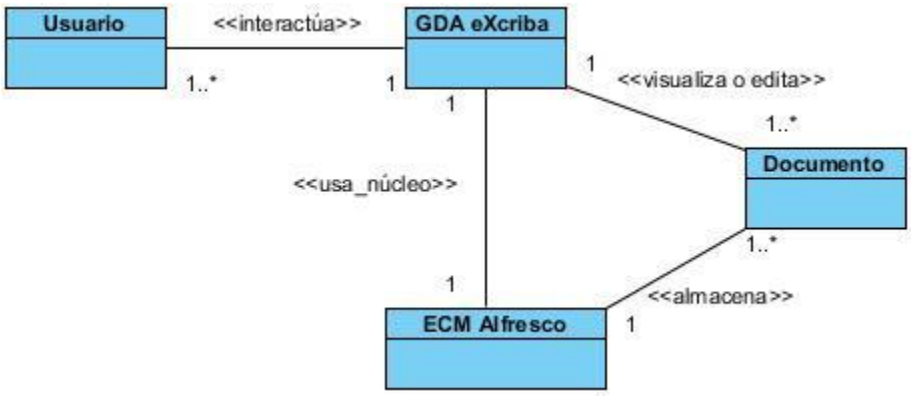

Figura 4 Modelo de dominio.

A.1. Figura 5: Diagrama de casos de usos del sistema

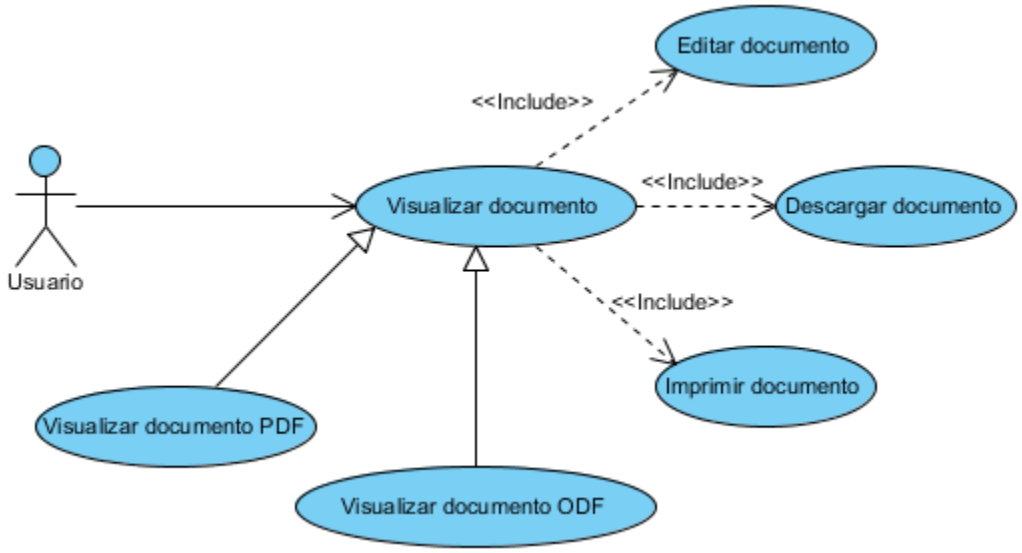

Figura 5 Diagrama de casos de uso del sistema.

A.1. Figura 6: Diagrama de clases del diseño

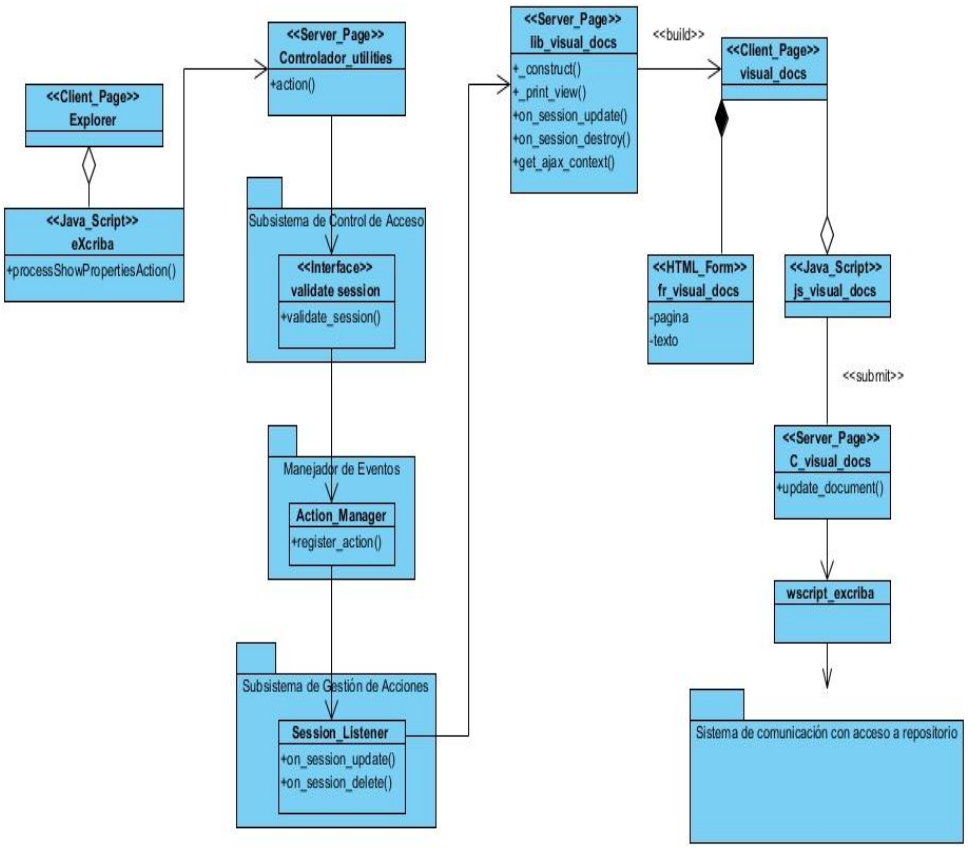

Figura 6 Diagrama de clases del diseño.

A.1. Figura 7: Diagrama de secuencia

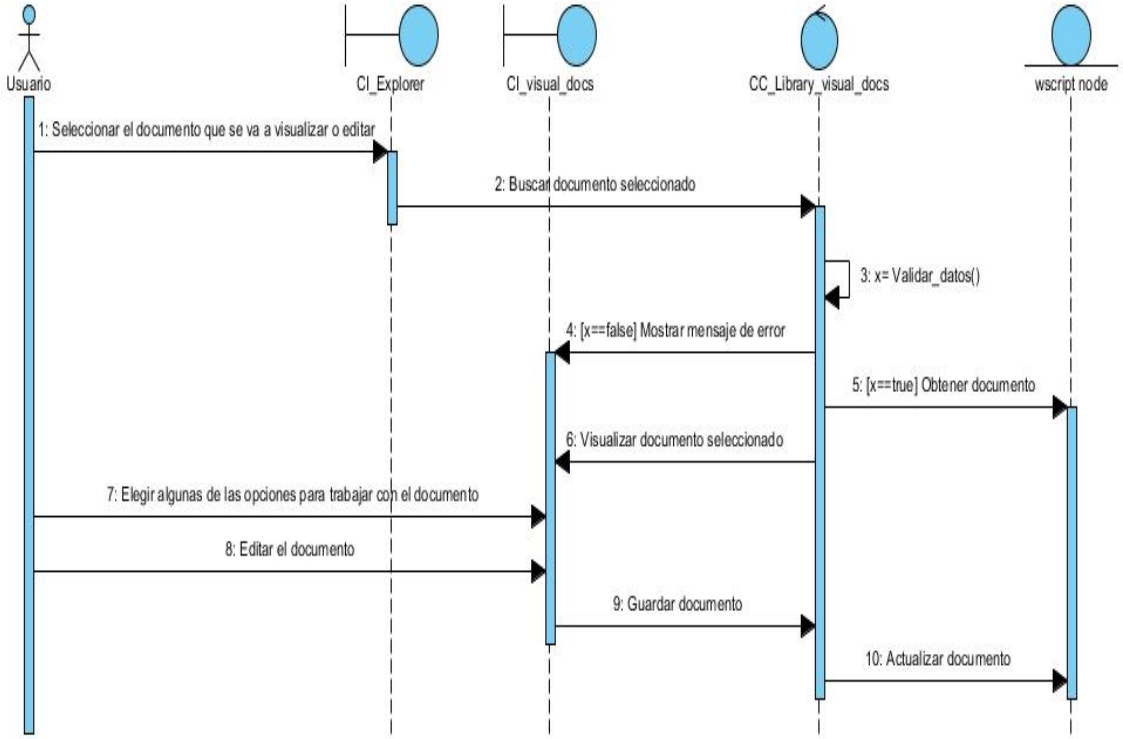

Figura 7 Diagrama de secuencia.

A.1. Figura 8: Descripción de la arquitectura

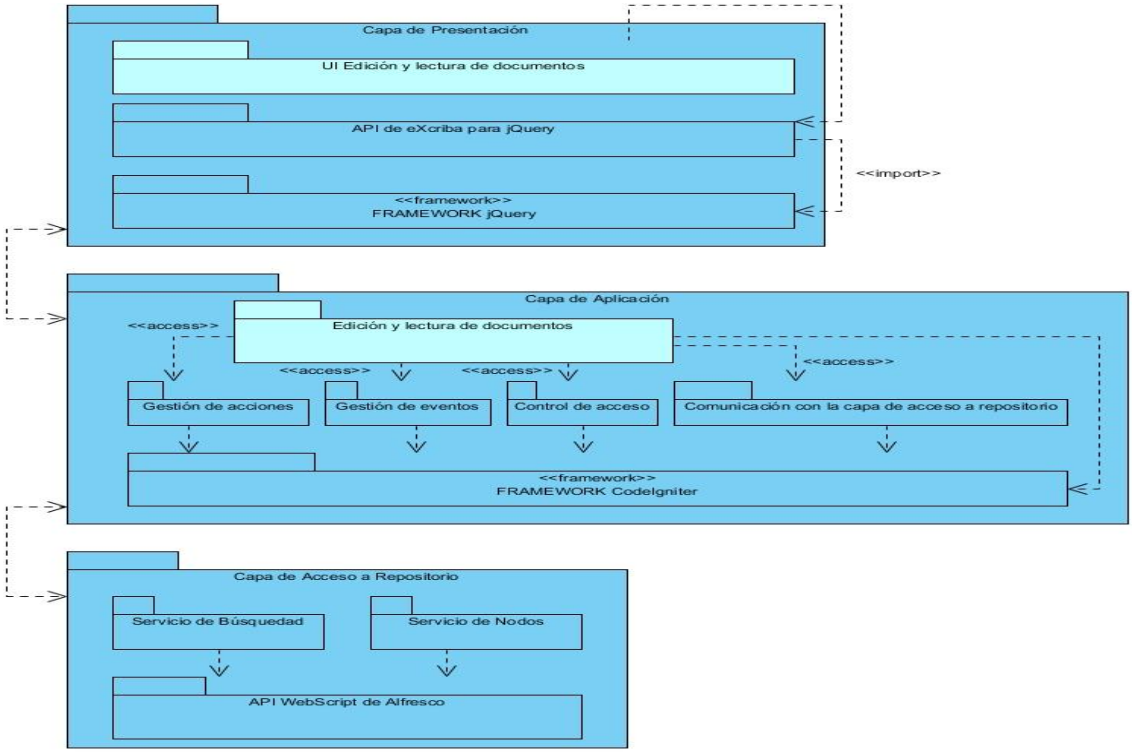

Figura 8 Descripción de la arquitectura.

A.1. Figura 9: Diagrama de componentes

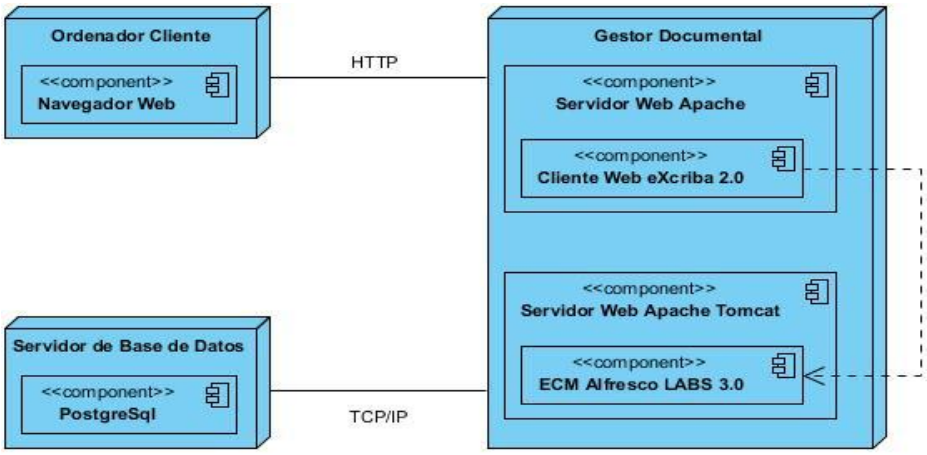

Figura 9 Diagrama de despliegue.

A.1. Figura 10: Diagrama de componentes

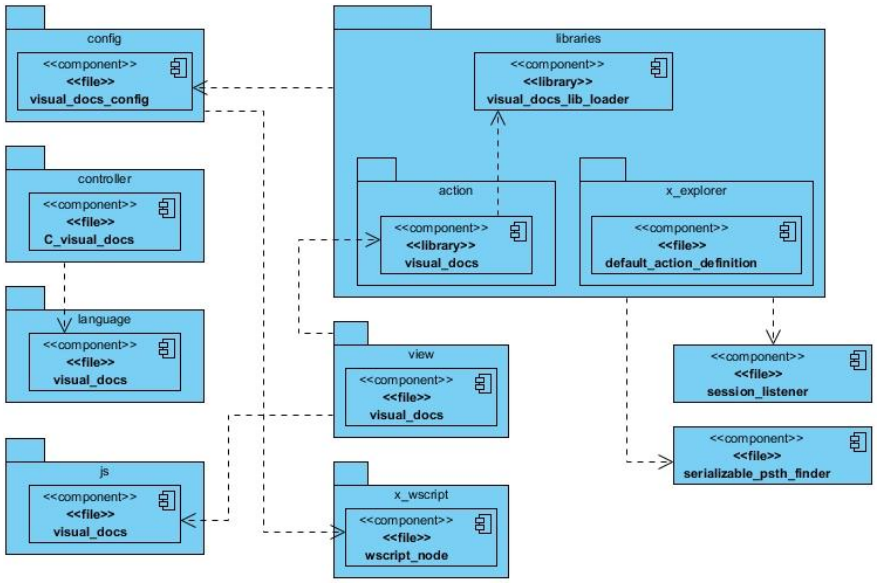

Figura 10 Diagrama de componentes.

A.1. Figura 11: Representación del algoritmo update_document()

```
15      public function update_document($uuid)
16      {
17          try
18          {
19              validate_session ();
20              $messages = load_text ( 'x_visual_docs' );
21              $node = Wscript_node::get_gen_props($uuid);
22
23              $extra_datas = array ('file' => '@' . realpath ( 'public/media/temp/'.$node['node.name'] ),
24                                    'name' => $node['node.name'],
25                                    'mimetype' => $node['node.mimetype'],
26                                    'title' => $node['node.title'],
27                                    'desc' => $node['node.description'],
28                                    'node_uuid' => $node['node.uuid'],
29                                    'contentType' => 'cm:content' );
30
31              $result = Wscript_node::update_upload_file ( $extra_datas );
32
33              if ($result ['msg'] == 0)
34              {
35                  //Limpiar directorio
36                  $dir = "public/media/temp/";
37
38                  $handle = opendir ( $dir );
39                  while ( $file = readdir ( $handle ) ) {
40                      if (is_file ( $dir . $file )) {
41                          unlink ( $dir . $file );
42                      }
43                  }
44
45                  $action_manager = get_resource_manager ( 'Action_manager' );
46                  $action_manager->register_action ( 'return_back',
47                          get_from_session ( 'current.id' ) );
48                  notify ( $messages ['action.update_document.validate.success'] );
49              }
50              else
51              {
52                  notify ( $messages ['action.update_document.validate.error.' . $result ['msg']],
53                          LEVEL_ERROR );
54              }
55          }
56          catch(Excriba_exc $e)
57          {
58              notify ( $e->get_local_message(), LEVEL_ERROR );
59          }
60      }
61
```

Figura 11 Representación del algoritmo update_document().

A.1. Figura 12: Diagrama del flujo asociado al algoritmo update_document()

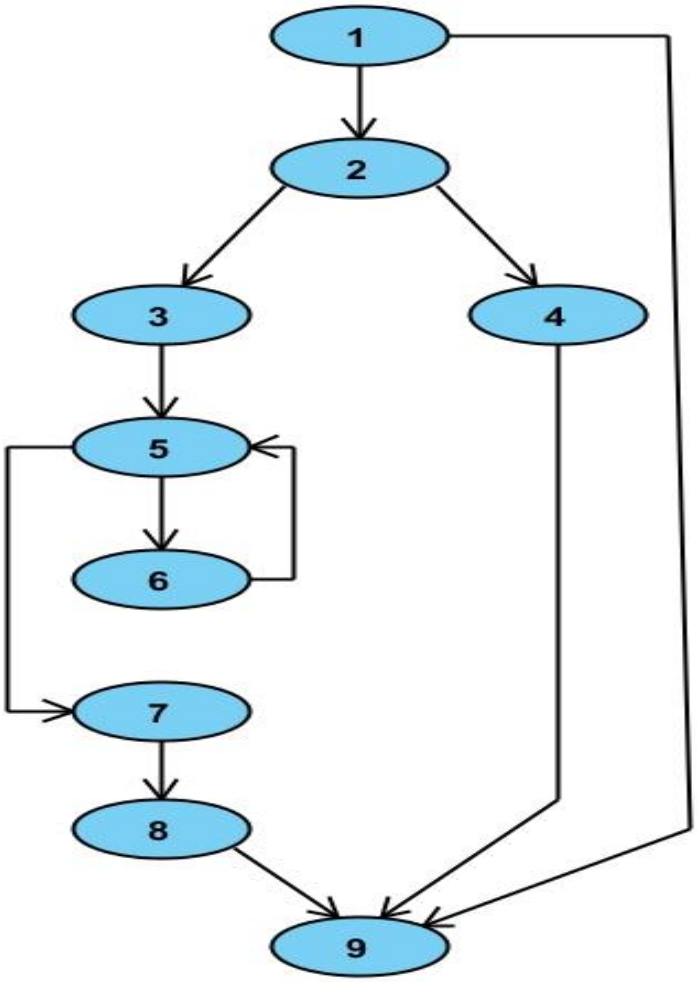

Figura 12 Diagrama del flujo asociado al algoritmo update_document().

Anexo B

Segundo Apéndice

B.1. Descripción de los casos de usos del sistema

CU 2. Visualizar documento PDF

Objetivo	Visualizar un documento de formato PDF dentro del Gestor de Documentos Administrativos eXcriba.	
Actores	**El Usuario:** Inicia el caso de uso, haciendo click sobre el documento de formato PDF que desea visualizar en el explorador de archivos del Gestor de Documentos Administrativos eXcriba.	
Resumen	Debe permitirle al usuario visualizar el documento de formato PDF.	
Complejidad	Alta	
Prioridad	Critico	
Precondiciones	El usuario tiene que estar autenticado en el sistema.	
Postcondiciones		
Flujo de eventos		
Flujo básico <Nombre del flujo básico>		
	Actor	**Sistema**
4.	El usuario selecciona dentro del explorador de archivos del Gestor de Documentos Administrativos eXcriba el documento con formato PDF que desea visualizar.	
5.		El sistema muestra la visualización del documento seleccionado.
6.		Finaliza el evento cuando el usuario salga de la vista que visualiza el documento.
Requisitos no funcionales	RnF 1, RnF 2, RnF 3, RnF7, RnF 20, RnF 21, RnF 22, RnF 23	

Tabla 20 Visualizar documento PDF.

CU 3. Visualizar documento ODF

Objetivo	Visualizar un documento de formato ODF dentro del Gestor de Documentos Administrativos eXcriba.	
Actores	**El Usuario:** Inicia el caso de uso, haciendo click sobre el documento de formato ODF que desea visualizar en el explorador de archivos del Gestor de Documentos Administrativos eXcriba.	
Resumen	Debe permitirle al usuario visualizar el documento de formato ODF.	
Complejidad	Alta	
Prioridad	Critico	
Precondiciones	El usuario tiene que estar autenticado en el sistema.	
Postcondiciones		
Flujo de eventos		
Flujo básico <Nombre del flujo básico>		
	Actor	**Sistema**
7.	El usuario selecciona dentro del explorador de archivos del Gestor de Documentos Administrativos eXcriba el documento con formato ODF que desea visualizar.	
8.		El sistema muestra la visualización del documento seleccionado.
9.		Finaliza el evento cuando el usuario salga de la vista que visualiza el documento.
Requisitos no funcionales	RnF 1, RnF 2, RnF 3, RnF7, RnF 20, RnF 21, RnF 22, RnF 23	

Tabla 21 Visualizar documento ODF.

CU 4. Editar documento

Objetivo	Editar un documento de dentro del Gestor de Documentos Administrativos eXcriba.	
Actores	**El Usuario:** Inicia el caso de uso, haciendo click sobre el documento que desea editar en el explorador de archivos del Gestor de Documentos Administrativos eXcriba.	
Resumen	Debe permitirle al usuario editar el documento.	
Complejidad	Alta	
Prioridad	Critico	
Precondiciones	El usuario tiene que estar autenticado en el sistema.	
Postcondiciones	Los cambios realizados en el documento son actualizados en el repositorio documental.	
Flujo de eventos		
Flujo básico <Nombre del flujo básico>		
	Actor	**Sistema**
1.	El usuario selecciona dentro del explorador de archivos del Gestor de Documentos Administrativos eXcriba el documento que desea editar.	
2.		El sistema muestra la visualización del documento seleccionado permite al usuario modificar el documento, de acuerdo a sus criterios.
3.		Finaliza el evento cuando el usuario salga de la vista que visualiza el documento.
Requisitos no funcionales	RnF 1, RnF 2, RnF 3, RnF7, RnF 20, RnF 21, RnF 22, RnF 23	

Tabla 22 Editar documento.

CU 5. Imprimir documento

Objetivo	Imprimir un documento visualizado dentro del Gestor de Documentos Administrativos eXcriba.	
Actores	El Usuario: Inicia el caso de uso, haciendo click sobre la funcionalidad que ejecuta la acción de imprimir dentro de la vista que muestra el documento.	
Resumen	Debe permitirle al usuario imprimir el documento.	
Complejidad	Alta	
Prioridad	Critico	
Precondiciones	El usuario tiene que estar autenticado en el sistema.	
Postcondiciones		
Flujo de eventos		
Flujo básico <Nombre del flujo básico>		
	Actor	**Sistema**
1.	El usuario selecciona dentro de la vista de visualización del documento el botón que ejecuta la acción de imprimir.	
2.		El sistema muestra una visualización previa de cómo quedaría impreso el documento.
3.	El usuario acepta imprimir el documento.	
4.		Finaliza el evento con la impresión del documento.
Requisitos no funcionales	RnF 1, RnF 2, RnF 3, RnF7, RnF 20, RnF 21, RnF 22, RnF 23	

Tabla 23 Imprimir documento.

CU 6. Descargar documento.

Objetivo	Descargar un documento visualizado dentro del Gestor de Documentos Administrativos eXcriba.	
Actores	**El Usuario:** Inicia el caso de uso, haciendo click sobre la funcionalidad que ejecuta la acción de descargar dentro de la vista que muestra el documento.	
Resumen	Debe permitirle al usuario descargar el documento.	
Complejidad	Alta	
Prioridad	Critico	
Precondiciones	El usuario tiene que estar autenticado en el sistema.	
Postcondiciones		
Flujo de eventos		
Flujo básico <Nombre del flujo básico>		
	Actor	**Sistema**
5.	El usuario selecciona dentro de la vista de visualización del documento el botón que ejecuta la acción de descargar.	
6.		El sistema muestra una ventana, donde el usuario debe elegir la ubicación donde desea descargar el documento, y si desea modificar el nombre del documento.
7.	El usuario acepta descargar el documento.	
8.		Finaliza el evento cuando el documento es descargado.
Requisitos no funcionales	RnF 1, RnF 2, RnF 3, RnF7, RnF 20, RnF 21, RnF 22, RnF 23	

Tabla 24 Descargar documento.

Anexo C

Tercer Apéndice

C.1. Estándar de codificación

Convención de nomenclatura

Variables locales:

Los nombres de las variables se rigen por la nomenclatura *CamelCase*[22], específicamente por el tipo *lowerCamelCase* que es cuando la primera letra de cada palabra se escribe con mayúscula a excepción de la primera palabra que se escribe con minúscula. Ejemplo:

```
$variable;
$variableNombreCompuesto;
```

- Los nombres de algunas variables locales, como los iteradores o los contadores, pueden especificarse en minúscula y de forma abreviada, siempre que su contexto sea específicamente local y su lectura sea intuitiva.

  ```
  Ejemplos: $cont, $i, $j.
  ```

- Al hacer asignaciones, debe existir un espacio a ambos lados del signo igual (=), esto funciona tanto para asignar un valor fijo, de otra variable o del resultado de una función.
- En el caso de un bloque de asignaciones relacionadas entre sí (por ejemplo, al inicializar un script), se pueden alinear los signos (=) agregando espacios extra, para mejorar la legibilidad.

Variables globales (constantes): los nombres de variables globales deben ser siempre en mayúsculas, separando las palabras con guiones bajos (“_”). Ejemplo:

```
define($CONSTANTE, valor);
define($CONSTANTE COMPUESTA, valor);
```

[22] *CamelCase* es un estilo de escritura que se aplica a frases o palabras compuestas. El nombre se debe a que las mayúsculas a lo largo de una palabra en *CamelCase* se asemejan a las jorobas de un camello. El término case se traduce como "caja tipográfica", que a su vez implica si una letra es mayúscula o minúscula.

Clases:

- El nombre debe ser descriptivo, evitando abreviaturas y siempre comienzan con mayúscula. En caso de nombres compuesto, se usará el carácter subrayado "_" para separar las palabras. Ejemplo:

```
Class Clase {
      // Bloque de instrucciones
}
```

```
Class Clase_nombre_compuesto {
      // Bloque de instrucciones
}
```

- La llave de inicio de la clase en la línea siguiente, identada correctamente.

Funciones:

- El nombre debe ser lo más descriptivo posible, evitando el uso de abreviaturas y empleando la convención *lowerCamelCase*.
- Los parámetros o argumentos son separados por espacios luego de la coma que los separa y aquellos con valores por defecto deben ser colocados al final de la lista.
- Siempre intentar retornar un valor significativo.
- La llave de inicio de la función se coloca en la línea siguiente, identada correctamente.

```
Function funcion($param1, param2 = array())
{
      // Bloque de instrucciones
}
Function funcionNombreCompuesto($param1, param2 = array())
{
      // Bloque de instrucciones
}
```

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

- *Vistas:* intuitivo y relacionado con el formulario y/o vista que representa.
- *Modelos:* con el mismo nombre de la clase que representa que contiene en el nombre el sufijo *_mdl* o *_base* en caso de ser modelos base.
- *Librerías:* con el mismo nombre de la clase que representa que contiene en el nombre el sufijo *_lib*.
- *Controladoras:* con el mismo nombre de la clase que representa.

Etiquetas de bloque PHP

Siempre utilizar **<?php** y **?>** para iniciar y terminar un bloque de código PHP, no las variantes **<?** y **?>** o **<%** y **%>**. Esto asegura compatibilidad entre diversas configuraciones de equipos.

Identación, llaves de apertura y cierre, tamaño de líneas

- Se identará con 4 espacios, sin tabulador, para que cualquier editor de texto reconozca correctamente la identación. Por otro lado, si bien existen editores que realizan corte automático de línea, es recomendable hacerlo en forma manual. Por esta razón la longitud de las líneas será de aproximadamente 75-80 caracteres.
- El uso de llaves de apertura y cierre será siempre en una nueva línea.

Estructuras de control

- Incluye **if**, **for**, **while**, **switch**, etc. Deben tener un espacio entre la palabra clave y el paréntesis de apertura, para diferenciarlos de las llamadas a funciones. Se recomienda utilizar siempre llaves de apertura y cierre, incluso en situaciones en las que técnicamente son opcionales, esto aumenta la legibilidad y disminuye la probabilidad de errores lógicos.
- Si las condiciones son muy largas que sobrepasan el tamaño de la línea, estas se dividen en varias líneas. En el mejor de los casos cuando la condición es muy extensa, se puede dividir esta en variables y compararlas dentro de la estructura de control. Ejemplo:

```
if($condicion = 1)
{
      // Bloque de instrucciones
}
else
{
      // Bloque de instrucciones
}
```

Documentación

Todos los archivos deben de tener la documentación asociada al mismo. Para esto debe de cumplir con el siguiente bloque al principio de cada clase. Ejemplo:

```
/**
* Breve descripción de la clase
* @package nombre del paquete o modulo al que pertenece
* @subpackage nombre del subpaquete (si es el caso)
* @category categoría de la clase (controladora, librería, modelo)
* @author nombre y apellidos del autor (se puede agregar el e-mail)
*/
```

De la misma forma las funciones de cada clase deberán ser documentadas como se muestra a continuación.

```
/**
* Breve descripción de la función
* @param nombre tipo de dato y nombre del parámetro (por cada uno)
* @return tipo de dato que retorna
* @author nombre y apellidos del autor (se puede agregar el e-mail)
*/
```

Comentarios

Se aconseja el uso de comentarios en línea para facilitar la comprensión del código, sobre todo en procedimientos complejos. Los comentarios pueden ser con fin documental o bien como “ayuda-memoria”. Para ello se recomienda utilizar los estilos de C (/* */) y C++ (//).

Anexo D

Cuarto Apéndice

D.1. Matriz de trazabilidad

	CU-1	CU-2	CU-3	CU-4	CU-5	CU-6
RF 1	X			X		
RF 1.1		X				
RF 1.2			X			
RF 1.3	X			X		
RF 1.4	X			X		
RF 2				X		
RF 3	X			X	X	
RF 4	X			X		X

Tabla 25 Matriz de trazabilidad.

D.2. Prototipos de los requisitos funcionales

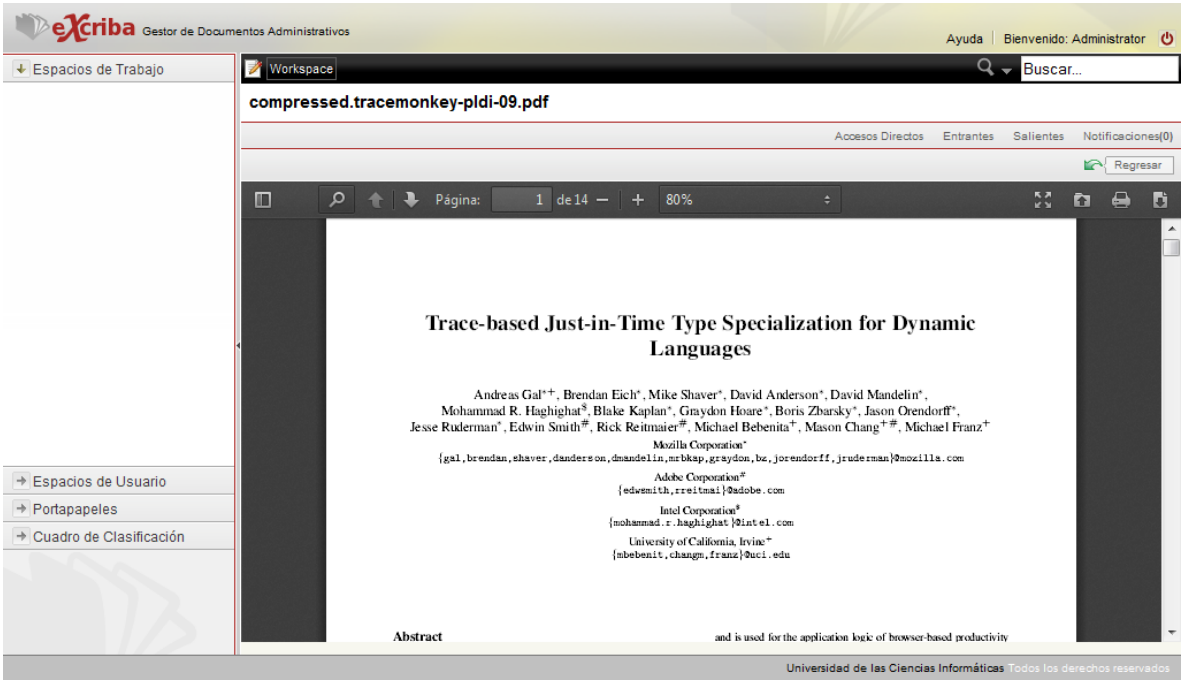

Figura 13 Prototipo del RF 1.1.

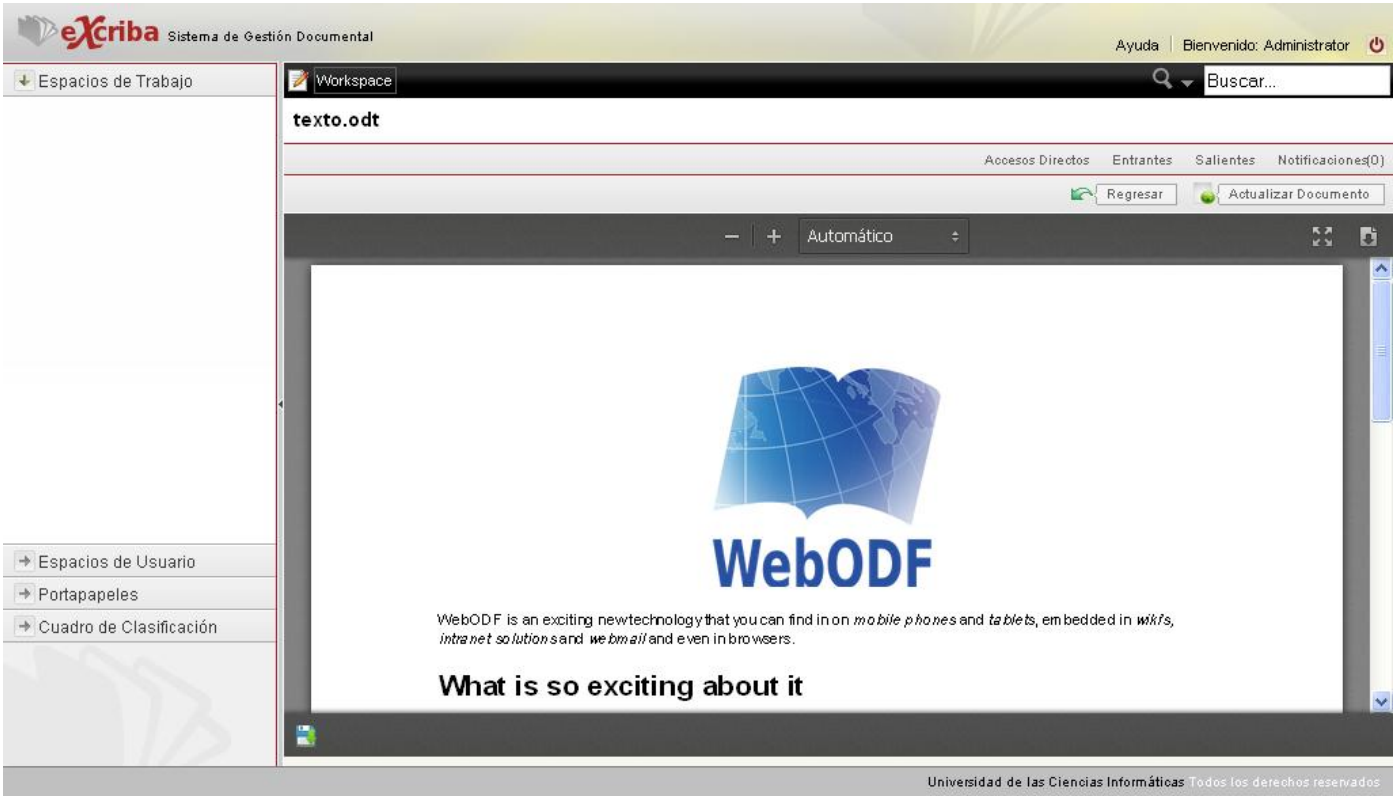

Figura 14 Prototipo del RF 1.2.

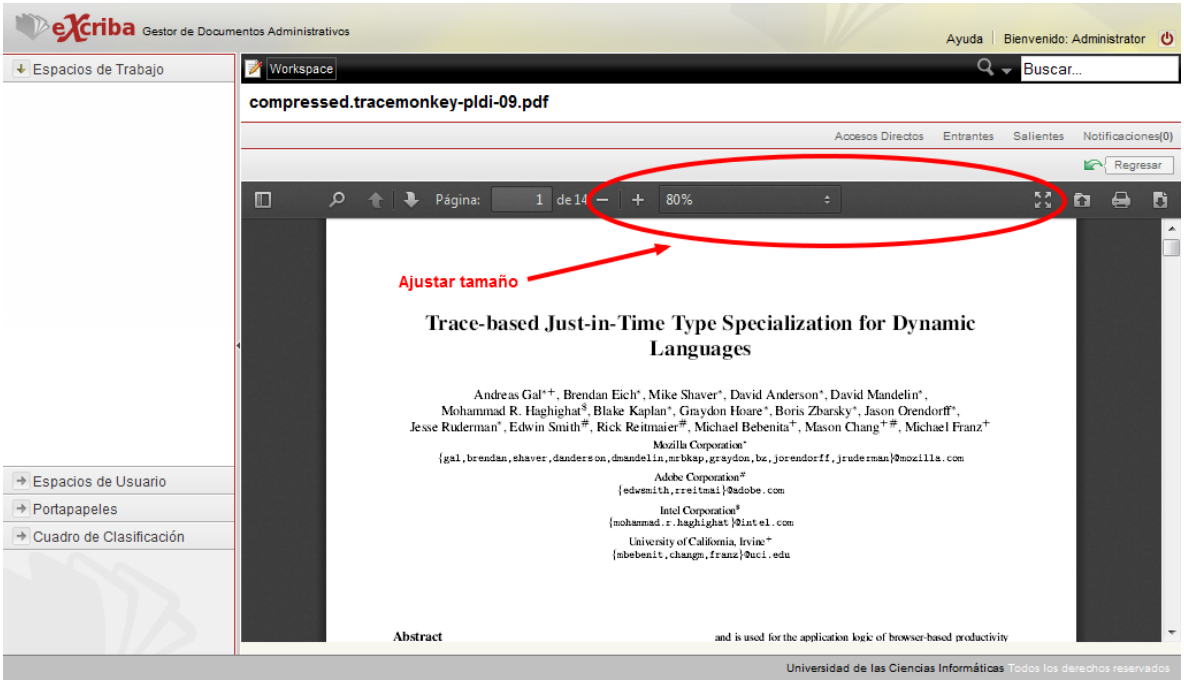

Figura 15 Prototipo del RF 1.3.

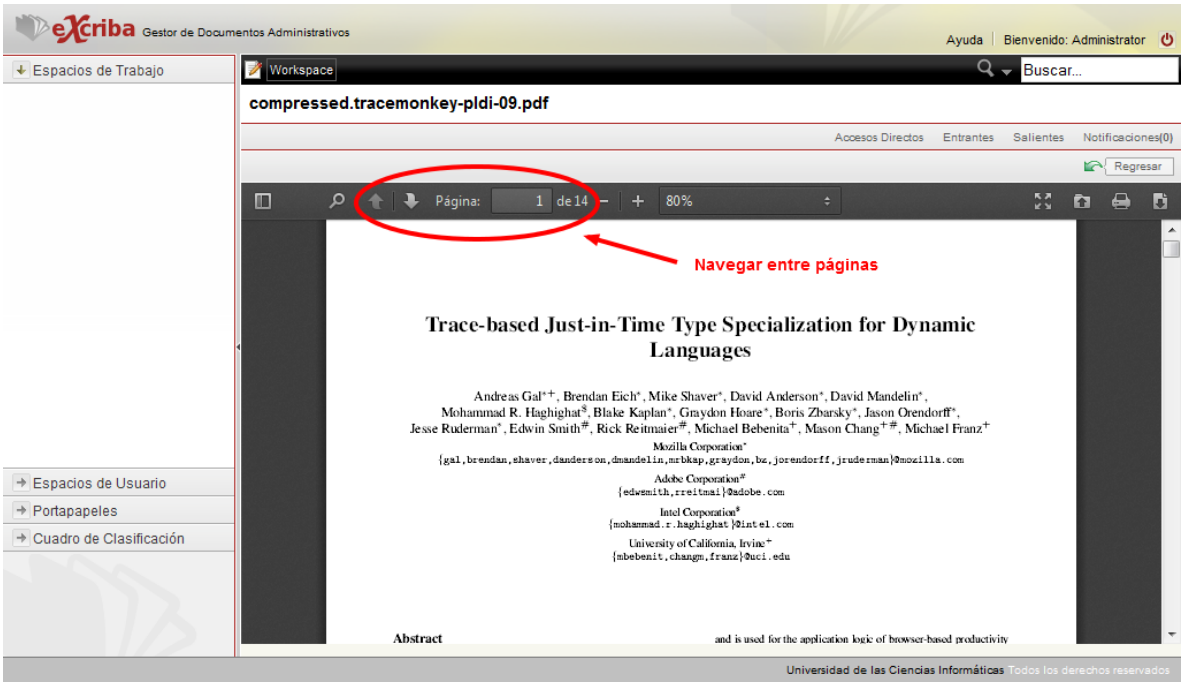

Figura 16 Prototipo del RF 1.4.

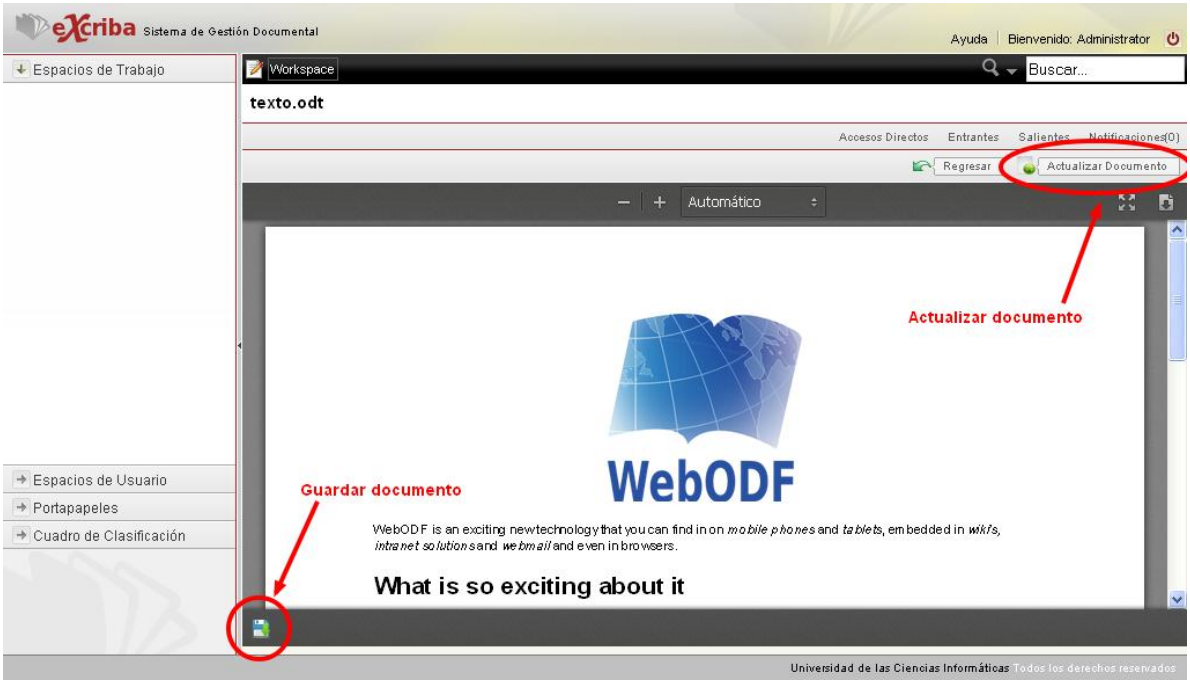

Figura 17 Prototipo del RF 2.

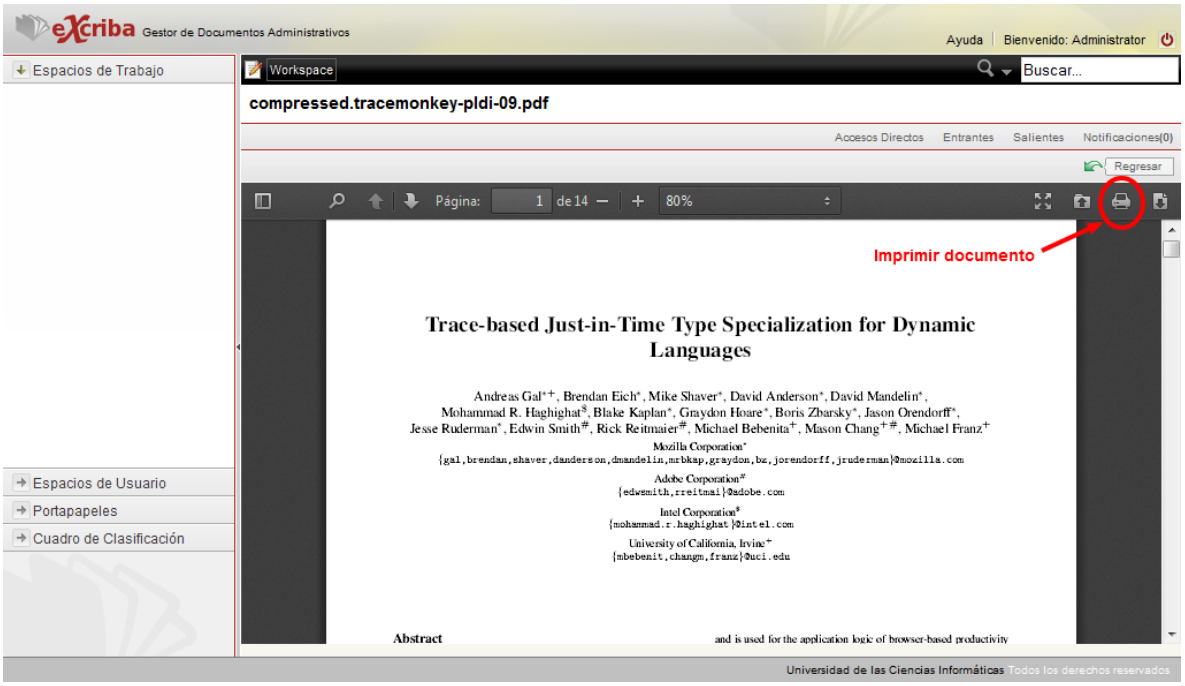

Figura 18 Prototipo del RF 3.

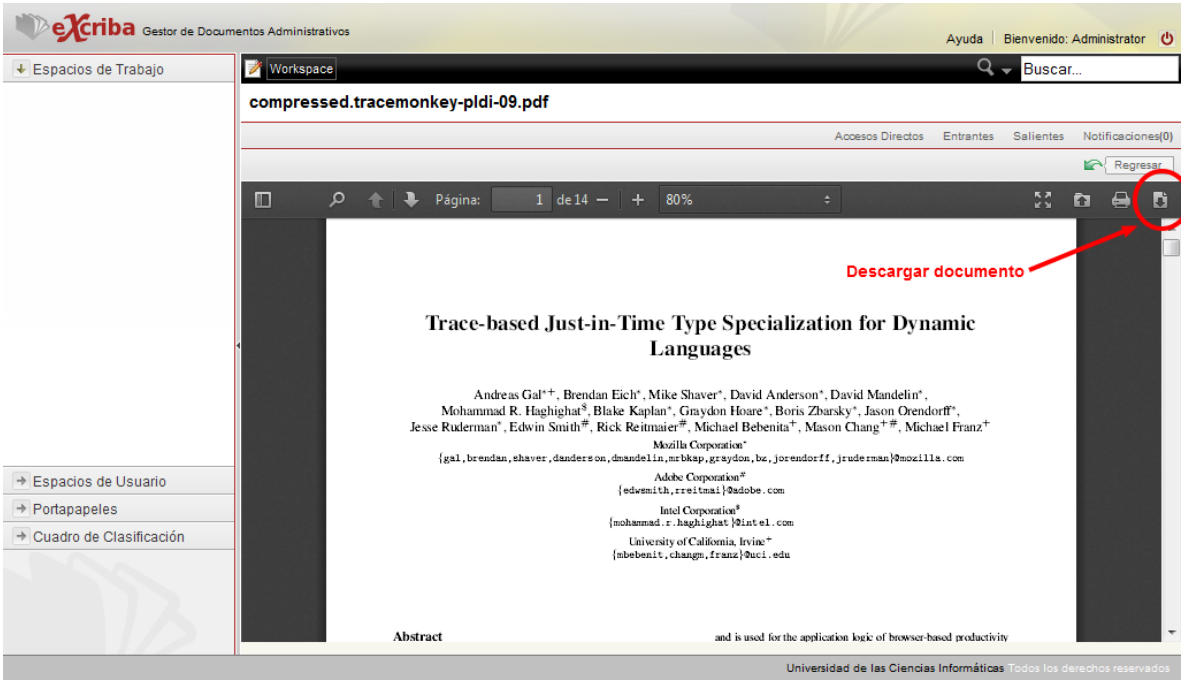

Figura 19 Prototipo del RF4.