Universidad de las Ciencias Informáticas

Facultad 2

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

Título: Herramienta para trabajo colaborativo en la gestión documental con Nuxeo.

Autores:

Roberto Pérez Santos.

Andy Díaz Montesino.

Tutores:

Ing. Wilson Alba Cal.

Ing. Gilberto Enrique González Hidalgo

La Habana, 2013

…todos los días hay que luchar porque ese amor a la humanidad viviente se transforme en hechos concretos, en actos que sirvan de ejemplo, de movilización.

Ernesto Che Guevara.

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de este trabajo y autorizo a la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste, firmo la presente a los ____ días del mes de _____ del año 2013.

_____ _____

Roberto Pérez Santos **Andy Díaz Montesino**

_____ _____

Ing. Wilson Alba Cal **Ing. Gilberto E. González Hidalgo**

DEDICATORIA

Este trabajo va dedicado a mis padres y a mi hermano.

Roberto

A mis padres y a mi hermano.

Andy

AGRADECIMIENTOS

A mis padres y a mi hermano por su apoyo y amor incondicional.

A mi Abuela Noelia y a mi tía Bury por su dedicación a mi persona.

A toda mi familia por el apoyo que me han brindado.

A Laura por estar siempre a mi lado.

A todas aquellas personas que he conocido y con las que he compartido estos 5 años de universidad.

Roberto.

A mi mamá, mi papá y mi hermano por siempre estar presente en mi vida, por todo su apoyo, dedicación y amor incondicional.

A mis abuelos y tíos por apoyarme

A mi cuñada y a todos los amigos que me ayudaron y estuvieron conmigo en todos los momentos.

Andy

RESUMEN

En la actualidad los sistemas de gestión documental han tomado gran auge siendo cada vez más utilizados por las empresas. Estos sistemas han tenido un gran impacto ya que cuentan con múltiples ventajas, entre ellas la preservación de la información y el control sobre la misma. El centro de informatización de seguridad ciudadana (ISEC) en la Universidad de las Ciencias Informáticas utiliza el sistema gestor documental Nuxeo, el cual brinda una serie de servicios, sin embargo no existe una herramienta de trabajo colaborativo que permita la sincronización de archivos con el servidor Nuxeo y la notificación de cambios de contenidos. Teniendo en cuenta la situación anteriormente expuesta, en el presente trabajo se propone el diseño e implementación de una herramienta que elimine estas deficiencias. Dicha herramienta mejorará la interacción del usuario con el servidor.

Índice

ÍNDICE

ÍNDICE

Índice de tablas

INTRODUCCIÓN

El avance de las Tecnologías de la Informática y las Comunicaciones (TIC) ha ampliado las capacidades físicas, mentales y las posibilidades de desarrollo social de todo ser humano. Las TIC no solamente incluyen la telemática, multimedia y los medios de comunicación, sino también la informática y sus tecnologías lo que ha provocado cambios numerosos en la forma de estructurar la información. Estos cambios radican fundamentalmente en la modernización y mejora en la gestión y conservación de la información ya sean documentos en soporte de papel o electrónicos, posibilitando un control sobre dichos documentos y los procesos a los que son sometidos.

En la actualidad la gestión documental está presente en disímiles ámbitos, fundamentalmente en el empresarial, ya que surge como una alternativa para tener toda la información digitalizada y centralizada disponible selectivamente en función del usuario, en la cual se puede realizar el acceso y edición simultánea a los documentos por varios usuarios a la vez. Todo esto supone un ahorro en tiempo y papel (y por tanto también de espacio), ya que no hay retrasos en tratamientos repetitivos, largos y costosos de documentos, búsquedas lentas o pérdidas de datos.

Cuba no se encuentra ajena a estos cambios y se ha introducido gradualmente en la creación y utilización de sistemas de gestión documental. En el centro de desarrollo ISEC de la facultad 2, en la Universidad de las Ciencias Informáticas se utiliza un sistema de gestión documental llamado Nuxeo, el cual no posee las funcionalidades de sincronización de contenidos de manera automática, lo que provoca que el usuario cada vez que realice un cambio en el archivo, tenga que revisar en la aplicación Web la fecha de modificación de los archivos y ver si ha ocurrido algún cambio. También presenta problema en la visualización de alertas de cambios durante el trabajo colaborativo o sea cuando dos o más personas trabajan sobre el mismo documento no se le notifica sobre algún cambio ocurrido, al no ser que esta notificación sea de forma personal o mediante otras vías de notificaciones pudiendo no llegarle al usuario, provocando pérdida de la información. Esto se debe a que la interfaz de usuario de dicho gestor documental es web, la cual no permite la automatización de dichas tareas.

Considerando la situación anteriormente mencionada se identifica como **problema a resolver:** ¿Cómo garantizar la sincronización automática de contenidos y visualización de alertas de cambios durante el trabajo colaborativo en la gestión documental con Nuxeo?

El **objeto de estudio** estaría centrado en la gestión documental. Enmarcando el **campo de acción** el trabajo colaborativo en la gestión documental con Nuxeo, tomando como **objetivo general:** Desarrollar una herramienta que permita la sincronización automática de los contenidos generados en los procesos de gestión documental a través de Nuxeo y la visualización de alertas de cambios de contenidos, garantizando el trabajo colaborativo.

Dicho objetivo a su vez está desglosado en los siguientes **objetivos específicos**:

- Definir los mecanismos para la comunicación con el servidor de gestión documental Nuxeo, para la sincronización de los contenidos y la notificación de cambios en los contenidos.
- Diseñar e implementar una herramienta informática que cumpla con los requerimientos y mecanismos antes definidos.
- Validar la solución propuesta.

Para llevar a término el resultado de la investigación, se expone como **idea a defender** que con el desarrollo y puesta en práctica de una aplicación escritorio para el trabajo colaborativo en la gestión documental con Nuxeo, se garantizará la sincronización automática de contenidos y la visualización de alertas de cambios durante el trabajo colaborativo.

Se trazan las siguientes **tareas de la investigación** para dar cumplimento al objetivo principal:

1. Definición de los mecanismos a utilizar para la comunicación con el servidor Nuxeo.
2. Definición de los mecanismos a utilizar para la sincronización de los contenidos.
3. Definición de los mecanismos a utilizar para la notificación de cambios en los contenidos.

4. Caracterización de las principales metodologías de desarrollo de software utilizadas y su adaptabilidad al sistema que se desarrollará.
5. Análisis sobre las tecnologías y selección de las más adecuadas para el desarrollo.
6. Diseño de la aplicación que soporte los mecanismos de interacción definidos.
7. Implementación de la solución propuesta.
8. Realización de pruebas funcionales.

Al desarrollar este trabajo se obtendrá como **resultado** una herramienta informática que posibilite la sincronización automática de los contenidos generados en los procesos de gestión documental a través de Nuxeo, como parte del trabajo colaborativo. Dicha aplicación deberá también visualizar la información básica referente a los cambios registrados.

Métodos de Investigación Científica

Entre los métodos teóricos que se emplearán para la investigación están:

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

La Modelación: Se describe el sistema a realizar a través de las historias de usuario, además se representan las clases y sus responsabilidades mediante la confección de las tarjetas Clase – Responsabilidad - Colaborador

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

Como método empírico utilizado para cumplir con las tareas se empleará:

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

La investigación quedará estructurada de la siguiente manera:

Capítulo 1: Fundamentación Teórica.

Se indaga en el estado de la gestión documental, se describen los elementos de la teoría que se necesitan para resolver el problema; se analizan, describen y fundamentan las herramientas y tecnologías que se utilizarán para la confección de la herramienta.

Capítulo 2: Características del sistema, Exploración y Planificación.

Se analizan y describen las características del sistema como el objeto de estudio; el entorno de trabajo en el que se desarrollará el sistema, los requerimientos funcionales y no funcionales, se plantean las Historias de Usuarios (HU) y el plan de iteraciones.

Capítulo 3: Diseño del sistema.

En este capítulo se define la arquitectura del sistema, así como los patrones de diseño empleados. Además se procederá a construir las Tarjetas CRC (Clase - Responsabilidad -Colaborador).

Capítulo 4: Implementación y Pruebas.

En este capítulo se describen las tareas de ingeniería haciendo un énfasis detallado de cada una de ellas, además de un resumen de las pruebas unitarias y de aceptación que fueron realizadas al sistema.

CAPÍTULO I: FUNDAMENTACIÓN TEÓRICA

1.1 Introducción

Este capítulo abordará conceptos fundamentales relacionados con la gestión documental, los sistemas de gestión documental, sincronización de contenidos entre otros. Además se describirá la importancia, características y ventajas principales de dichos términos. Se realizará un estudio de los principales sistemas que utilizan este tipo de alternativa. Además de realizarse un estudio de las metodologías, tecnologías y herramientas para el desarrollo de la herramienta.

1.2 Gestión documental

La gestión documental ha existido desde hace miles de años, debido a la necesidad de "documentar" o fijar actos administrativos y transacciones legales y comerciales por escrito para dar fe de los hechos. Este tipo de documentos se plasmaron sucesivamente en tablillas de arcilla, hojas de papiro, pergaminos y papel, cuya gestión se fue haciendo cada vez más compleja a medida que crecía el tamaño de los fondos documentales, esto se evidencia en la bibliotecas antiguas que organizaban listados de documentos por carpetas y categorías.

Con el desarrollo del sistema decimal se utilizaron la fichas /catálogos para clasificar la información haciendo más fácil su búsqueda. El surgimiento de las nuevas tecnologías de información y comunicación marcaron un precedente en esta área pues comenzaron a utilizarse en la administración pública y privada, sus comienzos tuvieron lugar con la aparición de las computadoras que propiciaron que los documentos comenzaran a organizarse en diferentes discos donde se etiquetaban manualmente y se guardaban en cajas tipo archivador para poder encontrar el disco apropiado según el documento que se necesitaba. Más adelante la interfaz gráfica de los sistemas operativos permitió desarrollar el concepto de las carpetas, organizando los documentos en los propios sistemas, almacenados en discos duros y posteriormente en servidores.

El inicio de las bases de datos, procesadores de textos, aplicaciones ofimáticas y sobre todo la llegada del correo electrónico, disparó la necesidad de conservar también documentos que nacen, viven y mueren en formato electrónico. Conseguir esto representó un nuevo salto en la complejidad y exigencias a los sistemas

informatizados, surgiendo los Sistemas de Gestión Documental los cuales trajeron cambios en la forma en que se gestionaba la información.

En la actualidad, coexisten en el mundo los más diversos sistemas de gestión documental: desde el simple registro manual de la correspondencia que entra y sale, hasta los más sofisticados sistemas informáticos. Estos sistemas manejan no sólo la documentación administrativa, ya sea en papel o en formato electrónico, sino que también controlan los flujos de trabajo del proceso de tramitación de cualquier información, capturan información desde bases de datos de producción, contabilidad y otros. Además se enlazan con el contenido de archivos, bibliotecas, centros de documentación y permiten realizar búsquedas.

1.3 Conceptos fundamentales

Para comprender mejor lo referente a la gestión documental es necesario que primero se conozcan algunos términos, por tanto a continuación se describen algunos de los conceptos más importantes.

1.3.1 Gestión Documental

Concepto que engloba todos los procesos y herramientas necesarios para administrar el ciclo completo de vida de los documentos de una organización, incluyendo su creación, digitalización, archivo, edición, consulta, distribución, aprobación, etc., mediante soluciones software (1).El término abarca todo lo referente a la captura, almacenamiento y recuperación de documentos. Determina que estén guardados el tiempo estrictamente necesario fijado por la entidad o el que resulte imprescindible para realizar sobre ellos las operaciones oportunas. Esta gestión se realiza a través de un conjunto de normas, técnicas y prácticas usadas para administrar el flujo de documentos de todo tipo en una organización, asegurándose la conservación indefinida de los más valiosos, aplicando principios de racionalización y economía.

1.3.2 Sistemas de Gestión Documental

Los sistemas de gestión documental son programas informáticos dedicados a crear una base de datos con una amplia tecnología. Estos datos pueden ser científicos, culturales, administrativos, técnicos, etc. (2). Permiten el manejo, gestión, conservación, publicación y trabajo sobre documentos electrónicos (ya sean documentos escaneados o que se haya creado originalmente en formato digital).

1.3.3 Trabajo Colaborativo

El trabajo colaborativo se halla donde los individuos trabajan juntos, debido a la naturaleza de sus tareas. La tarea del grupo debe ser colaborativa en su naturaleza. Las personas involucradas comparten las mismas metas, parte de las cuales es el cumplimiento de su tarea compartida. Por esto el trabajo colaborativo es claramente no competitivo. Se desarrolla en un espacio normalmente informal y usualmente se ejecuta en grupos pequeños, generalmente proyectos grupales. Los miembros del grupo hacen uso extensivo de la comunicación horizontal. Esta puede tomar lugar tanto en formas de interacción indirectas como directas y distribuidas o no distribuidas. Los límites del trabajo colaborativo no son siempre congruentes con los límites de la organización formal; en realidad, un proceso de trabajo colaborativo involucraría a personas en sitios distintos, y se caracteriza por ser relativamente autónomo. Influencias externas sobre las tareas, es decir, planificación y control externo, reducen la naturaleza colaborativa del trabajo. Sin embargo, esto no significa que no es planificado o más bien programado. (3)

1.3.4 Sincronización de contenidos

Se refiere a la sincronización de archivos o contenidos cuando dos dispositivos se actualizan de forma que contengan los mismos datos y cuando varios procesos se ejecutan a la vez con el propósito de completar una tarea.

1.3.5 Notificación y Alerta

En las herramientas informáticas estos términos se utilizan para avisar al usuario de cualquier cambio que tenga lugar en el entorno donde se encuentra, avisándole de cualquier error, confirmación o simplemente notas que sean de su interés.

1.4 Sistemas de Gestión Documental

El gran volumen, la complejidad y diversidad de información estructurada y no estructurada, ha provocado que sea algo imprescindible apoyarse en aplicaciones de gestión documental en cualquier tipo de organización para evitar el caos en los sistemas de información. Estos sistemas permiten almacenar y recuperar la información, son vitales para la supervivencia de cualquier organización, sin importar su tamaño. Los documentos son un recurso y un activo organizacional. Como recurso, proveen información, y como activo, proveen documentación. Si se utilizan sistemas automatizados para archivar la información, éstos ayudan a localizar el documento en

una forma más rápida y desde cualquier lugar, sea en la organización o fuera de ésta evitando la pérdida de dichos documentos.

En este tipo de sistema se tiene acceso a la información y los documentos de manera segura y estructurada, al poder definir diferentes permisos de acceso a los datos y documentos dentro del sistema de gestión documental. Se tiene un control de cambios de los documentos, de revisiones y accesos. Existe una amplia colaboración entre usuarios para la creación, modificación y gestión de la misma documentación. La captura y extracción de la información de los documentos se puede realizar en papel y la integración en el sistema de gestión documental como metadatos. Éstos se indexan para su fácil recuperación mediante búsquedas personalizadas.

Los sistemas de gestión documental pueden realizar la recuperación de los documentos en formato electrónico. Garantizan el mínimo de costos en la organización basado en la reducción de tiempo de los empleados al automatizar y facilitar en gran medida las tareas de gestionar la documentación y los procesos de la empresa, el ahorro de dinero al no tener que utilizar tanto papel en la empresa, y el aprovechamiento máximo del espacio al no tener que destinar departamentos para guardar archivos en formato de papel. A continuación se muestran ejemplos de sistemas de gestión documental:

Nuxeo: Gestor documental de código abierto, su punto fuerte es la gestión centralizada de la documentación, basada en Java, multiplataforma, por lo tanto, se puede instalar tanto en Windows como en Mac o Linux. Dispone de algunas de las opciones que facilitan la tarea a la hora de gestionar los documentos en la empresa como son los siguientes:

- **Servidor de documentación:** principal propósito de este programa. Centraliza toda la documentación y la deja accesible en la red, ya sea a nivel interno o a través de Internet. A la vez permite la gestión de usuarios y permisos para la seguridad documental de la empresa.
- **Espacio de usuario y de grupo:** permite gestionar cuotas ya sean para un usuario concreto o para un grupo.
- **Versiones:** posibilita conocer quién cambió qué y en qué momento.
- **Flujos de trabajo:** lleva un registro del documento en cada momento y su estado si se tiene que realizar alguna acción con el mismo, por ejemplo una

revisión, o ver por quién ha pasado, y el estado del documento en cada uno de esos procesos (aprobado, obsoleto, borrado, etc.…).

- **Notificaciones:** brinda la posibilidad de suscribirse a actualizaciones de carpetas mediante RSS. Además guarda en cada documento un hilo temporal con los comentarios que cada usuario ha realizado en cada momento. (4)

Nuxeo brinda una serie de servicios para integrarse a él entre ellos se encuentran:

SOAP. (Simple Object Access Protocol).

AMF\Flash. (Action Message Format).

REST. (Transferencia de Estado Representacional).

En SOAP las respuestas son demasiado complejas y difíciles de interpretar. Una vez implementado, si se desea cambiar algo en el servidor impacta de forma negativa en los clientes. Requiere de librerías para su interpretación de parte del cliente y de parte del servidor.

AMF es un protocolo de servicios web de alto rendimiento que ha sido desarrollado y diseñado especialmente para aplicaciones Flash de Adobe. AMF es generalmente considerado un protocolo RPC (es decir, una alternativa optimizada para SOAP) (5). Este servicio no va a ser utilizado ya que define una forma de invocar métodos remotos desde un cliente basado en Flash y no coincide con las características del sistema.

Para comunicar la herramienta con el gestor documental se utilizará el servicio REST ya que es muy ligero y sus respuestas contienen exactamente la información que se necesita. Para los humanos es muy fácil y simple de interpretar. Es sencillo de desarrollar y no se necesita mucho código extra. Es flexible en cuanto al tipo de respuesta que se necesita. Además cuenta con una librería desarrollada en java, con el objetivo de facilitar el trabajo con dicho servicio.

1.5 Software de trabajo colaborativo

Los Software de trabajo colaborativo o groupware son programas informáticos que integran el trabajo en un sólo proyecto con muchos usuarios concurrentes que se

encuentran en diversas estaciones de trabajo conectadas a través de una red. Entre las principales ventajas de estos software se encuentran las siguientes: la comunicación fluye más rápido y de manera más precisa, permite hacer un mejor uso de los recursos humanos al permitir la colaboración entre personas ubicadas en sitios remotos y con diferentes horarios de trabajo y mejora la productividad. A pesar de contar con innumerables beneficios, cuenta con una serie de inconvenientes que radican fundamentalmente en problemas de comunicación entre los participantes ya que es deficiente comparada con la comunicación directa.

1.5.1 Ejemplos de Software de trabajo colaborativo

Redmine: Es una herramienta para la gestión de proyectos que incluye un sistema de seguimiento de incidentes con seguimiento de errores. Es software libre y de código abierto. Entre sus principales características se encuentran: Soporte para múltiples proyectos, sistema de seguimientos de errores y roles flexibles basados en el control de acceso.

Teambox: Es un proyecto de software colaborativo en línea, que combina las mejores prácticas en productividad con herramientas propias de redes sociales. Este proyecto trata de cubrir la necesidad de una herramienta de administración de proyectos con capacidades de colaboración y fácil de usar. Posee integración con Google Docs., Dropbox e integra fácilmente archivos, calendarios y email.

1.6 Software que utilizan la sincronización de contenidos

Un software sincronizador de contenidos es un programa que hace copias de seguridad del contenido de dos o varios ordenadores interconectados (también llamados sistemas o terminales) y actualiza los cambios realizados en cualquiera de ellos de manera automática. Estos programas no reescriben el documento nuevo encima del viejo, sino solamente los cambios. Esta opción está pensada para no perder tiempo ni malgastar recursos del sistema. No sólo permite usar indistintamente ambos ordenadores para el mismo trabajo, sino que ofrece una copia de seguridad actualizada.

1.6.1 Ejemplos de software que utilizan la sincronización de contenidos

Git: Es un sistema de control de versiones libre y de código abierto, diseñado por Linus Torvalds para manejar todo, desde pequeños a grandes proyectos con rapidez y

eficiencia. Trabaja pensando en la confiabilidad del mantenimiento de versiones de aplicaciones cuando estas tienen un gran número de archivos de código fuente. Controla todas las modificaciones dentro del código y sabe exactamente los cambios y modificaciones que se hicieron y alerta a los demás desarrolladores para no tener que reescribir el código o romper modificaciones de otros desarrolladores. (6)

Subversion: Es un sistema para el control de versiones, fue creado para centralizar y compartir información, gestiona los cambios de archivos y directorios a través del tiempo y permite recrear un proyecto desde cualquier momento de su historia.

1.7 Mecanismos de envío de notificaciones y alertas

En los sistemas informáticos existen disímiles formas de notificar a los usuarios. El aumento del uso de los mensajes de texto, el correo electrónico y la mensajería instantánea han posibilitado que el envío de notificaciones y alertas mediante ellos sea cada vez mayor.

Uno de los métodos más comunes para mostrar notificaciones en un sistema informático es el uso de los cuadros de diálogos o ventanas. Luego de analizarse los mecanismos de notificaciones anteriormente mencionados, se determinó utilizar los cuadros de diálogos para mostrar las notificaciones en la herramienta a implementar. Las notificaciones a través de estos cuadros de diálogos son sencillas de implementar y permiten una comunicación simple entre el cliente y la herramienta informática. No conllevan a un coste elevado para su envío como es el caso del envío de notificaciones mediante mensajes de texto. La herramienta a desarrollar mostrará las notificaciones cuando ocurra un cambio en los archivos ya sea una modificación, actualización, etc. Además posibilitará configurar el tiempo de duración de dichas alertas.

1.8 Mecanismos de sincronización

Subversion identifica las diferencias de un archivo usando un algoritmo de diferenciación binario, que funciona exactamente igual en archivos de texto (legibles) y archivos binarios (ilegibles por los humanos). Ambos tipos de archivos se almacenan comprimidos en el repositorio y las diferencias se transmiten en ambas direcciones a través de la red. (7)

El proceso de actualización de los archivos contenidos en un repositorio y la posterior generación de una nueva versión utilizando **Git** es el siguiente: cada

desarrollador debe generar una copia del repositorio original en su computadora, creando un repositorio local. Este repositorio local contiene toda la información del historial de cambios y los archivos del directorio de trabajo. En esta etapa el desarrollador puede empezar a trabajar en el repositorio Git local, creando y modificando los archivos de acuerdo a sus requerimientos. (7)

La herramienta realiza la sincronización de manera que a la hora de descargar el archivo del servidor se le realiza un hash al archivo con el algoritmo MD5 y se guarda en la carpeta de trabajo. Para saber si ha cambiado el estado del archivo se le realiza un hash al archivo local y se compara con el hash guardado en la carpeta de trabajo, en caso de ser diferentes, entonces ha ocurrido un cambio y el estado pasa a configurado local. Para saber si ha ocurrido algún cambio en el servidor se le realiza un hash al archivo del servidor y se compara con el hash guardado en la carpeta de trabajo, en caso de ser diferentes, el estado del archivo cambia a modificado remoto. Por tanto si es modificado localmente y modificado remotamente entonces se establece un conflicto.

1.9 Metodología, Herramientas y Tecnologías a utilizar

Metodología

Una metodología de software se puede definir como una colección de procedimientos, técnicas, herramientas y documentos auxiliares que ayudan a los desarrolladores de software en sus esfuerzos por implementar nuevos sistemas de información. Una metodología está formada por fases, cada una de las cuales se puede dividir en sub-fases, que guiarán a los desarrolladores de sistemas a elegir las técnicas más apropiadas en cada momento del proyecto y también a planificarlo, gestionarlo, controlarlo y evaluarlo. (8)

Las metodologías de desarrollo de software se caracterizan en dos corrientes las Metodologías Ágiles y las Metodologías Tradicionales. Entre estas metodologías existen grandes diferencias que se muestran en la siguiente tabla:

Tabla 1 Comparación entre Metodologías Ágiles y Tradicionales (9)

Metodologías Ágiles	Metodologías Tradicionales
Pocos Artefactos. El modelado es	Más Artefactos. El modelado es esencial,

prescindible, modelos desechables**.**	mantenimiento de modelos.
Pocos Roles, más genéricos y flexibles**.**	Más Roles, más específicos.
No existe un contrato tradicional, debe ser bastante flexible**.**	Existe un contrato prefijado.
Cliente es parte del equipo de desarrollo**.**	El cliente interactúa con el equipo de desarrollo mediante reuniones.
Orientada a proyectos pequeños. Corta duración (o entregas frecuentes), equipos pequeños (< 10 integrantes) y trabajando en el mismo sitio**.**	Aplicables a proyectos de cualquier tamaño, pero suelen ser especialmente efectivas/usadas en proyectos grandes y con equipos posiblemente dispersos.
La arquitectura se va definiendo y mejorando a lo largo del proyecto**.**	Se promueve que la arquitectura se defina tempranamente en el proyecto.
Énfasis en los aspectos humanos: el individuo y el trabajo en equipo**.**	Énfasis en la definición del proceso: roles, actividades y artefactos.
Basadas en heurísticas provenientes de prácticas de producción de código**.**	Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo.
Se esperan cambios durante el proyecto**.**	Se espera que no ocurran cambios de gran impacto durante el proyecto.

Luego del análisis realizado se decidió utilizar una metodología ágil para desarrollar la herramienta. Entre las más conocidas que se encuentran en esta categoría están: Scrum, Programación Extrema (XP), Crystal, entre otras. Para determinar cuál de ellas era la indicada para desarrollar la herramienta se analizaron las principales características del proyecto a realizar, que se muestran a continuación.

- ✓ Posee un equipo de desarrollo pequeño.
- ✓ Se necesita estar en constante comunicación con el cliente.
- ✓ Se cuenta con poco tiempo para su implementación.

- ✓ Posee pocos artefactos y el modelado es prescindible.
- ✓ Se esperan cambios durante el proyecto.

Por estas razones se elige entre las metodologías ágiles a Programación Extrema (XP) ya que está centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP recomienda que las tares de desarrollo se realicen en parejas lo que se adapta a las características principales de la herramienta a implementar.

Esta metodología también se basa en la retroalimentación continua entre el cliente y el equipo de desarrollo, la comunicación fluida entre todos los participantes, la simplicidad en las soluciones implementadas y es fácil para gestionar los cambios. XP se define como especialmente adecuada para proyectos con requisitos imprecisos muy cambiantes donde existe un alto riesgo técnico. (10)

Lenguajes

Java: Es un lenguaje de programación de propósito general, concurrente, basado en clases, y orientado a objetos, que fue diseñado específicamente para tener tan pocas dependencias de implementación como fuera posible. Es utilizado dicho lenguaje ya que es multiplataforma permitiendo desarrollar en disímiles entornos, posee multitud de librerías de código abierto que pueden utilizarse en caso de que los contenidos que se necesiten no se encuentren en las librerías nativas de Java. Otra de las grandes ventajas es que existe gran cantidad de documentación.

UML: Es un lenguaje gráfico para visualizar, especificar y documentar cada una de las partes que comprende el desarrollo de software. UML entrega una forma de modelar elementos conceptuales como lo son procesos de negocio y funciones de sistema, además de elementos concretos como escribir clases en un lenguaje determinado, esquemas de base de datos y componentes de software reusables. Su objetivo principal es entregar un material de apoyo que le permita al lector poder definir diagramas propios como también poder entender el modelamiento de diagramas ya existentes.

Herramienta de modelado

Visual Paradigm: Visual Paradigm para UML es una herramienta UML profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. (11)Es una suite completa de herramientas de Ingeniería de Software Asistida por Computadora (CASE).

Herramienta de desarrollo

NetBeans: Es un entorno de desarrollo integrado, una herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas. Permite rápida y fácilmente desarrollar aplicaciones Java de escritorio, móviles y aplicaciones web, mientras que también proporciona una gran herramienta para desarrolladores de PHP y C / C + +. Es gratuito, de código abierto con una gran comunidad de usuarios y desarrolladores de todo el mundo. (12)

Tecnologías

Framework Spring: Ofrece una programación completa y un modelado de configuración para aplicaciones basadas en Java, en cualquier tipo de plataforma de despliegue. Un elemento clave de este Framework es el apoyo de infraestructura a nivel de aplicación. Spring incluye inyección de dependencias, es flexible con estilos de configuración XML y anotaciones basadas en soporte avanzado para la programación orientada a aspectos. (13)

Una vez concluida la aplicación se desarrollarán las pruebas pertinentes para la validación de su funcionamiento. Para las pruebas unitarias, que se realizan sobre el código Java, se utilizará la librería JUnit 4.0. Este tipo de herramienta hace que el esfuerzo y el trabajo en la fase de pruebas se reduzcan, permitiendo que el desarrollador se centre en la verificación de resultados correctos y no escribiendo código extenso para realizar sus pruebas. Además proporciona clases de las cuales se pueden heredar para formar las nuevas clases que serán las que realicen las pruebas unitarias a cada una de las clases que conforman la aplicación. Una prueba puede estar conformada por una serie de datos, utilización y resultados, este último se

compara con los datos que en realidad debería mostrar el software, para tener conocimiento de si la aplicación está cumpliendo con la realización de lo solicitado. (14)

1.9 Conclusiones parciales

En este capítulo se analizaron los principales conceptos relacionados con la gestión documental, el trabajo colaborativo y la sincronización de contenidos. También, se seleccionó la metodología ágil XP como metodología a usar, los lenguajes UML y Java, como herramientas de desarrollo se seleccionó el IDE Netbeans dado a que son las herramientas y tecnologías que cumplieron con las características necesarias para dar solución a la propuesta.

CAPÍTULO II: PROPUESTA DE SOLUCIÓN, EXPLORACIÓN Y PLANIFICACIÓN

2.1 Introducción

En este capítulo se hará una propuesta de solución del sistema a desarrollar, se describirán los requisitos funcionales y no funcionales de la herramienta. Además se profundizará en las fases de exploración y planificación propias de la metodología de desarrollo de software utilizada, donde se confeccionan las HU. Se creará el plan de iteraciones así como el plan de duración de las mismas para obtener el plan de entrega de la herramienta.

2.2 Propuesta de solución

Se propone la implementación de una aplicación de escritorio que se comunique con el servidor Nuxeo a través de un servicio web REST, la cual permita acceder a los archivos guardados en el servidor Nuxeo, sin necesidad de depender de un navegador, ni de interactuar con una interfaz web, agregando funcionalidades como:

- ✓ Actualizaciones automáticas.
- ✓ Notificaciones en el escritorio.

2.3 Levantamiento de requisitos

Un requerimiento es una característica que el sistema debe tener o restricción que el sistema debe satisfacer para ser aceptada por el cliente. El levantamiento de requisitos es la especificación del sistema en términos que el cliente entienda. (15)

2.3.1 Requisitos funcionales

Los requisitos funcionales describen la interacción entre el sistema y su ambiente independientemente de su implementación. El ambiente incluye al usuario y cualquier otro sistema externo que interactúa con el sistema.

A continuación se muestran los requisitos funcionales

1. Configurar carpeta de trabajo
2. Configurar servidor
3. Configurar puerto

4. Configurar tiempo de encuesta
5. Configurar datos personales del usuario por sesión
6. Configurar tiempo de visualización de alertas
7. Visualizar estado de la conexión
8. Probar conexión con el servidor Nuxeo
9. Crear archivos
10. Eliminar archivos
11. Sincronizar archivos
12. Compartir carpeta
13. Descompartir carpeta
14. Mostrar carpetas compartidas
15. Mostrar historial de archivos
16. Visualizar alertas

2.3.2 Requisitos no funcionales

Los requisitos no funcionales describen aspectos del sistema que son visibles por el usuario que no incluyen una relación directa con el comportamiento funcional del sistema. Además incluyen restricciones como el tiempo de respuesta, la precisión, recursos consumidos, seguridad, etc.

Usabilidad: Se necesita una preparación previa, aunque no extensa para operar con el sistema. Se requiere un nivel medio de conocimientos de computación, aunque el manejo de la aplicación es sencillo y permite la fácil comprensión por el usuario.

Seguridad: Los usuarios se autenticarán con el servidor Nuxeo para poder acceder a sus archivos.

Software: Se requiere la instalación de la máquina virtual de java 1.6 o superior.

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

Interfaz de usuario: La aplicación propuesta poseerá una interfaz amigable y tendrá diseño sencillo.

2.4 Exploración

En esta fase, los clientes plantean a grandes rasgos las historias de usuario que son de interés para la primera entrega del producto. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto. Se prueba la tecnología y se exploran las posibilidades de la arquitectura del sistema construyendo un prototipo. (16)

2.4.1 Historias de usuario

Las historias de usuario corresponden a la técnica utilizada para especificar los requisitos del software. Se trata de formatos en los cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales. El tratamiento de las historias de usuario es muy dinámico y flexible. Cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas. (17)

Tabla 2 Historia de Usuario No.1. Configurar carpeta de trabajo

Historia de Usuario	
Número: 1	**Nombre de Historia de Usuario:** Configurar carpeta de trabajo
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Sistema	**Iteración Asignada:**1
Prioridad en negocio: Media	**Puntos estimados**: 1
Riesgo en Desarrollo: Medio	**Puntos Reales:** 1
Descripción: Configurar la carpeta de trabajo, la cual usará el sistema para almacenar los archivos pertenecientes al usuario.	

Observaciones:
Prototipo de interfaz:

Tabla 3. Historia de Usuario No.9. Sincronizar archivos.

Historia de Usuario	
Número: 9	**Nombre de Historia de Usuario:** Sincronizar archivos
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Sistema	**Iteración Asignada:**2
Prioridad en negocio: Alta	**Puntos estimados**: 3
Riesgo en Desarrollo: Medio	**Puntos Reales:** 3
Descripción: Permite la sincronización de un archivo en el área de trabajo con respecto al servidor.	
Observaciones:	
Prototipo de interfaz:	

Tabla 4. Historia de Usuario No.18. Visualizar alertas

Historia de Usuario	
Número: 16	**Nombre de Historia de Usuario:** Visualizar alertas
Modificación de Historia de Usuario Número: Ninguna	
Usuario: Sistema	**Iteración Asignada:**3
Prioridad en negocio: Alta	**Puntos estimados**: 3
Riesgo en Desarrollo: Medio	**Puntos Reales:** 3
Descripción: Visualizará una alerta cuando ocurra una actualización en el área de trabajo, con los detalles siguientes: • Nombre del fichero modificado.	

• URL local del fichero modificado. • Tipo de modificación realizada (adición, actualización, eliminación, asignación/des-asignación, otros) con un color diferente en cada caso. • Fecha de creación. • Fecha de modificación. • Usuario que creó el fichero. • Usuario que realizó la última modificación. • Tipo de documento (Word, PDF, otros).
Observaciones:
Prototipo de interfaz:

2.5 Planificación

En esta fase el cliente establece la prioridad de cada historia de usuario, y correspondientemente, los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. Se toman acuerdos sobre el contenido de la primera entrega y se determina un cronograma en conjunto con el cliente.

2.5.1 Estimación de esfuerzo por Historias de Usuario

Las estimaciones de esfuerzo asociado a la implementación de las historias la establecen los programadores utilizando como medida el punto. Un punto, equivale a una semana ideal de programación. Las historias generalmente valen de 1 a 3 puntos. Por otra parte, el equipo de desarrollo mantiene un registro de la "velocidad" de desarrollo, establecida en puntos por iteración, basándose principalmente en la suma de puntos correspondientes a las historias de usuario que fueron terminadas en la última iteración. (18)

Tabla 5. Estimación de esfuerzo por Historia de Usuario

No	Historia de Usuario	Puntos de estimación
1	Configurar carpeta de trabajo	1
2	Configurar servidor	1

3	Configurar puerto	1
4	Configurar tiempo de encuesta	1
5	Configurar datos personales del usuario por sesión	1
6	Configurar tiempo de visualización de alertas	3
7	Probar conexión con el servidor Nuxeo	3
8	Visualizar estado de la conexión	3
9	Crear archivos	3
10	Eliminar archivos	3
11	Sincronizar archivos	3
12	Compartir carpeta	3
13	Descompartir carpeta	3
14	Mostrar carpetas compartidas	3
15	Mostrar historial de archivos	3
16	Visualizar alertas	3

2.6 Iteraciones

Esta fase incluye varias iteraciones sobre el sistema antes de ser entregado. El Plan de Entrega está compuesto por iteraciones de no más de tres semanas. En la primera iteración se puede intentar establecer una arquitectura del sistema que pueda ser utilizada durante el resto del proyecto. Los elementos que deben tomarse en cuenta durante la elaboración del Plan de la Iteración son: historias de usuario no abordadas, velocidad del proyecto, pruebas de aceptación no superadas en la iteración anterior y tareas no terminadas en la iteración anterior.

2.6.1 Plan de iteraciones

Iteración 1

En la iteración 1 se llevará a cabo el desarrollo de las HU del número 1 hasta el número 6.

Iteración 2

En la iteración 2 se llevará a cabo el desarrollo de la HU número 7 a la número 11.

Iteración 3

En la iteración 3 se llevará a cabo el desarrollo de la HU número 12 al número 16, obteniéndose una versión inicial del producto final y a partir de aquí el sistema se pondrá en función para ser evaluado.

2.6.5 Plan de duración de las iteraciones

El plan de duración de las iteraciones es el encargado de mostrar las HU que serán implementadas en cada una de las iteraciones, así como la duración estimada y el orden de implementación de cada una de ellas. Este plan se crea para lograr una mayor organización del trabajo. A continuación se muestra cómo quedarán distribuidas las HU según el orden en que serán abordadas en cada iteración.

Tabla 6. Plan de duración de las iteraciones

No. De iteración	Orden de Historia de Usuario a implementar	Duración
1	Configurar carpeta de trabajo Configurar servidor Configurar puerto Configurar tiempo de encuesta Configurar datos personales del usuario por sesión	1 semana
2	Configurar tiempo de visualización de alertas Visualizar estado de la conexión	3 semanas

	Probar conexión con el servidor Nuxeo Crear archivos Eliminar archivos Sincronizar archivos	
3	Compartir carpeta Descompartir carpeta Mostrar carpetas compartidas Mostrar historial de archivos Visualizar alertas	3 semanas

2.6.6 Plan de entregas

El Plan de Entrega está compuesto por iteraciones de no más de tres semanas. En este epígrafe se presenta el plan de entregas estimado para la fase de implementación, como producto del presente plan se realizarán versiones entregables del sistema al finalizar cada iteración.

Tabla 7. Plan de entregas

Herramienta	**Fin Iteración 1** **(8 de abril de 2013)**	**Fin Iteración 2** **(29 de abril de 2013)**	**Fin Iteración 3** **(22 de mayo de 2013)**
Nuxeo-Desktop	Funcionalidades 1.0	Funcionalidades 1.2	Funcionalidades 1.3

2.7 Conclusiones parciales

En este capítulo se hizo una propuesta de solución de la herramienta a desarrollar, se definieron los requisitos funcionales y no funcionales que el sistema debe cumplir, además se determinaron las HU haciendo una breve descripción de cada una y se estimó el tiempo que requerirá su implementación, también se confeccionó el plan de

iteraciones especificando la duración en semanas que se necesitará para concluirlas y por último se elaboró el plan de entregas.

CAPÍTULO III: DISEÑO DEL SISTEMA

3.1 Introducción

En este capítulo se describe la fase de diseño propio de la metodología de desarrollo XP. Se identifican y organizan las clases relevantes para las funcionalidades del sistema mediante las tarjetas CRC (Clase-Responsabilidad-Colaborador), así como el patrón arquitectónico y los patrones de diseño utilizados en el desarrollo de la herramienta.

3.2 Patrones de diseño

Los patrones de diseño son soluciones simples y elegantes a problemas específicos y comunes del diseño orientado a objetos. (19) Brindan una solución ya probada y documentada a problemas de desarrollo de software que están sujetos a contextos similares. Los patrones de diseño expresan esquemas para definir estructuras de diseño (o sus relaciones) con las que construir sistemas de software.

3.2.1 Patrones GoF

Los patrones de diseño GoF (Gang of Four) o Grupo de los Cuatro se clasifican en 3 grandes categorías basadas en su propósito: creacionales, estructurales y de comportamiento. (20) Estos patrones definen el comportamiento entre las clases y los objetos. Los utilizados en la implementación de la herramienta son:

Decorador: El marco de trabajo Spring implementa este patrón, el cual es el encargado de asignarle responsabilidades a objetos y añadir funcionalidades de manera dinámica.

Instancia Única: Garantiza la existencia de una única instancia para una clase y la creación de un mecanismo de acceso global a dicha instancia. El marco de trabajo Spring lo utiliza a través de la inyección de dependencia que permite tener acceso a dicha instancia desde cualquier clase.

Fachada: Se aplica para utilizar una interfaz común para un conjunto de interfaces del módulo, haciendo que el mismo sea más fácil de usar.

Fábrica: Proporciona una interfaz para la creación de objetos interdependientes o interrelacionados, sin especificar sus clases concretas.

3.2.2 Patrones GRASP

Los patrones GRASP (**G**eneral **R**esponsibility **A**ssignment **S**oftware **P**atterns) describen los principios fundamentales de la asignación de responsabilidades a objetos, expresados en formas de patrones. En el desarrollo de la herramienta se utilizaron los siguientes:

Experto: Se aplica para definir a qué clase se le debe asignar una responsabilidad, teniendo en cuenta que la misma contenga la información necesaria para cumplir con dicha responsabilidad. El uso de este patrón permitirá a los objetos valerse de su propia información para hacer lo que se les pide favoreciendo la existencia de mínimas relaciones entre las clases.

Creador: Este patrón se aplica para garantizar que una instancia de un objeto sólo pueda ser creada por la clase que contiene la información necesaria para ello.

Bajo Acoplamiento: La utilización de este patrón trajo consigo la existencia de pocas dependencias entre las clases. Esto aumenta la reutilización y logra que se tenga la mínima repercusión posible en el resto de clases, en caso de producirse una modificación en alguna de ellas.

Alta Cohesión: Este patrón se aplica para garantizar que los datos y responsabilidades de una clase sean coherentes y estén fuertemente ligados a la misma, en un sentido lógico.

Controlador: El uso de este patrón permite asignar la responsabilidad de controlar el flujo de eventos del sistema, a clases específicas. En este caso la clase controladorallamadaPrincipal.java, recibe los datos del usuario y los envía a las distintas clases según el método llamado.

3.3 Arquitectura

El término arquitectura se refiere a un grupo de abstracciones y patrones que nos brindan un esquema de referencia útil para guiarnos en el desarrollo de software dentro de un sistema informático21. Entre los diferentes patrones arquitectónicos que existen encontramos al n capas, este patrón tiene como objetivo principal separar los diferentes aspectos del desarrollo, tales como las cuestiones de presentación, lógica

de negocio, mecanismos de almacenamiento, etc. Las principales ventajas de este estilo son: se logran aplicaciones robustas debido al encapsulamiento, su mantenimiento es más sencillo ya que permite arreglar directamente el componente en cuestión. Además presenta una separación adecuada de funciones y mayor flexibilidad permitiendo añadir nuevos módulos para dotar al sistema de nuevas funcionalidades.

En el desarrollo de la herramienta se hizo uso de este patrón y quedó estructurado de la siguiente manera:

Capa Presentación: Esta capa está compuesta por las clases encargadas de las interfaces de usuarios. Como ejemplos de estas clases se muestran las siguientes: ListaArchivos.java, Principal.java, Archivo.java y ModeloListaArchivo.java.

Capa Servicios: Esta capa encierra toda la lógica del negocio de la aplicación. Entre las clases que se encuentran dentro de esta capa están: ServicioConfiguración.java, ServicioOperaciones.java y ServicioNotificaciones.java.

3.4 Tarjetas CRC

En la fase de diseño, para organizar las clases más relevantes del sistema, la metodología de desarrollo XP propone la creación de las tarjetas CRC.

La característica más sobresaliente de estas tarjetas es su simpleza y ductilidad, ya que no son más que fichas de papel o cartón que representan una entidad del sistema, a las cuales se les asigna responsabilidades y colaboraciones. (21)

Tabla 8. Tarjeta CRC Clase ServicioConfiguracion

Clase ServicioConfiguracion	
Descripción: clase que se encarga de gestionar los parámetros de configuración	
Responsabilidad	**Colaborador**
Encriptar contraseña	StringEncrypter
Guardar los parámetros de configuración	
Chequear la configuración	
Leer los parámetros de configuración	

Tabla 9. Tarjeta CRC Clase OperacionesArchivos

Clase OperacionesArchivos	
Descripción: Clase encargada de realizar toda la gestión de los archivos y carpetas	
Responsabilidad	**Colaborador**
Obtener la lista de subcarpeta dado una carpeta	HttpAutomationClient
Sincronizar archivos o carpetas con el servidor	HttpAutomationClient
Ejecutar un archivo de la copia local	
Compartir y descompartir carpeta con otro usuario	HttpAutomationClient
Crear nuevos archivos y carpetas	HttpAutomationClient
Eliminar archivos y carpetas	HttpAutomationClient
Realizar la comprobación de estado de archivo	HttpAutomationClient

Tabla 10. Tarjeta CRC Clase ServicioNotificaciones

Clase ServicioNotificaciones	
Descripción: Clase que se encarga de gestionar las notificaciones	
Responsabilidad	**Colaborador**
Mostrar estado de la conexión en la barra de tareas	
Mostrar las notificaciones de cambio de estado	Telegraph

3.5 Conclusiones parciales

En el presente capítulo fueron abordados los contenidos de la fase de diseño, siendo generados sus artefactos, se explicaron las Tarjetas CRC y se definieron los patrones de diseño y de arquitectura a utilizar.

CAPÍTULO IV: IMPLEMENTACIÓN Y PRUEBAS

4.1 Introducción

En el presente capítulo se detallan las iteraciones que serán llevadas a cabo durante la implementación del sistema y se describen las tareas de ingeniería generadas por cada una de las Historias de Usuario. Por último, y no menos importante se realizarán las pruebas al software.

4.2 Implementación

En esta fase se realiza la implementación de las historias de usuarios planteadas anteriormente y se determinan las tareas de ingeniería que se realizan para completar su desarrollo. Estas tareas se describen en forma de ficha que contendrá el número identificador de la tarea, el identificador de la historia de usuario con la que está relacionada, el nombre de la tarea, el tipo (nuevo desarrollo, corrección, mejora, etc. la fecha de inicio, su fecha de fin, el programador responsable y la descripción.

4.2.1 Iteración 1

A continuación se muestran las HU implementadas en la iteración 1.

Tabla 11. HU implementadas en la iteración 1

Historias de Usuarios	**Tiempo Estimado (días)**	**Tiempo Real (días)**
Configurar carpeta de trabajo	2	2
Configurar servidor	1	1
Configurar puerto	1	1
Configurar tiempo de encuesta	1	1
Configurar datos personales del usuario por sesión	2	2
Total	7	**7 (1 semana)**

Ejemplos de las tareas de ingenierías generadas en esta iteración.

Tabla 12. Tarea de ingeniería No. 1. Implementar funcionalidad configurar carpeta de trabajo

Tarea de Ingeniería	
Número Tarea: 1	**Número Historia de Usuario:** 1
Nombre Tarea: Implementar funcionalidad Configurar Carpeta de trabajo	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 2/7
Fecha Inicio: 1-04-2013	**Fecha Fin:** 2-04-2013
Programador Responsable: Roberto Pérez Santos	
Descripción: Se implementa la funcionalidad Configurar Carpeta de trabajo, permitiendo que el usuario pueda configurar su espacio de trabajo.	

Tabla 13. Tarea de ingeniería No. 5.Implementar funcionalidad Configurar datos personales del usuario por sesión

Tarea de Ingeniería	
Número Tarea: 5	**Número Historia de Usuario:** 5
Nombre Tarea: Implementar funcionalidad Configurar datos personales del usuario por sesión	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 2/7
Fecha Inicio: 5-04-2013	**Fecha Fin:** 8-04-2013
Programador Responsable: Roberto Pérez Santos y Andy Díaz montesino	
Descripción: Se implementa la funcionalidad Configurar datos personales del usuario por sesión permitiendo que cada usuario pueda tener una configuración diferente en su sesión.	

4.2.2 Iteración 2

A continuación se muestran las HU implementadas en la iteración 2.

Tabla 14. HU implementadas en la iteración 2

Historias de Usuarios	Tiempo Estimado (días)	Tiempo Real (días)
Configurar tiempo de visualización de alertas	2	2
Visualizar estado de la conexión	2	2
Probar conexión con el servidor Nuxeo	3	3
Crear archivos	3	3
Eliminar archivos	4	4
Sincronizar archivos	7	7
Total	**21**	**21 (3 semanas)**

Ejemplos de las tareas de ingenierías generadas en esta iteración.

Tabla 15. Tarea de ingeniería No. 9.Implementar funcionalidad Crear archivos

Tarea de Ingeniería	
Número Tarea: 9	**Número Historia de Usuario:** 9
Nombre Tarea: Implementar funcionalidad Crear archivos	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 3/21
Fecha Inicio: 16-04-2013	**Fecha Fin:** 19-04-2013
Programador Responsable: Roberto Pérez Santos	
Descripción: Se implementa la funcionalidad Crear archivos permitiendo crear nuevos archivos.	

Tabla 16. Tarea de ingeniería No. 9.Implementar funcionalidad Crear archivos

Tarea de Ingeniería	
Número Tarea: 11	**Número Historia de Usuario:** 11

Nombre Tarea: Implementar funcionalidad Sincronizar Archivos	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 3/21
Fecha Inicio: 21-04-2013	**Fecha Fin:** 29-04-2013
Programador Responsable: Roberto Pérez Santos	
Descripción: Se implementa la funcionalidad Sincronizar Archivos que permitirá que los usuarios actualicen los archivos ya sea para bajarlos al servidor o subirlos.	

4.2.3 Iteración 3

A continuación se muestran las HU implementadas en la iteración 3.

Tabla 17. HU implementadas en la iteración 3

Historias de Usuarios	Tiempo Estimado (días)	Tiempo Real (días)
Compartir carpeta	5	5
Descompartir carpeta	5	5
Mostrar carpetas compartidos	2	2
Mostrar historial de archivos	2	2
Visualizar alertas	7	7
Total	**21**	**21 (3 semanas)**

Ejemplos de las tareas de ingenierías generadas en esta iteración.

Tabla 18. Tarea de ingeniería No. 13.Implementar funcionalidad Descompartir carpeta

Tarea de Ingeniería	
Número Tarea: 13	**Número Historia de Usuario:** 13
Nombre Tarea: Implementar funcionalidad Descompartir carpeta	

Tipo de Tarea: Desarrollo	**Puntos Estimados:** 5/21
Fecha Inicio: 4-05-2013	**Fecha Fin:** 9-05-2013
Programador Responsable: Roberto Pérez Santos	
Descripción: Se implementa la funcionalidad Descompartir carpeta dando la posibilidad al usuario de Descompartir el directorio deseado.	

Tabla 19. Tarea de ingeniería No. 16. Implementar funcionalidad Visualizar alertas

Tarea de Ingeniería	
Número Tarea: 16	**Número Historia de Usuario:** 16
Nombre Tarea: Implementar funcionalidad Visualizar alertas	
Tipo de Tarea: Desarrollo	**Puntos Estimados:** 7/21
Fecha Inicio: 13-05-2013	**Fecha Fin:** 22-05-2013
Programador Responsable: Roberto Pérez Santos	
Descripción: Se implementa la funcionalidad Visualizar Alertas que permite notificar a los usuarios cuando ocurran cambios en el servidor.	

4.3 Pruebas

La metodología XP anima a probar tanto como sea posible. Esto permite aumentar la calidad de los sistemas reduciendo el número de errores no detectados y disminuyendo el tiempo transcurrido entre la aparición de un error y su detección. También permite aumentar la seguridad de evitar efectos colaterales no deseados a la hora de realizar modificaciones. XP divide las pruebas del sistema en dos grupos: pruebas unitarias, encargadas de verificar el código y diseñada por los programadores, y pruebas de aceptación o pruebas funcionales destinadas a evaluar si al final de una iteración se consiguieron las funcionalidades requeridas diseñadas por el cliente final. (22)

Pruebas Unitarias: Las pruebas unitarias son las encargadas de verificar el código y son diseñadas por los programadores. Cada uno de los desarrolladores tiene que ir probando constantemente lo que va obteniendo en el transcurso de la implementación

de un sistema, para garantizar que las funcionalidades exigidas por el cliente estén siendo implementadas correctamente (23). Las pruebas unitarias se fueron desarrollando a medida que se terminaba de implementar alguna funcionalidad, probándola directamente en el entorno real, para lo cual se utilizó la librería JUnit.

.**Pruebas de aceptación:** Por su parte las pruebas de aceptación son especificadas por el cliente y se enfocan en las características generales y las funcionalidades del sistema, están destinadas a evaluar si al final de una iteración se obtuvo la funcionalidad requerida; así como comprobar que dicha funcionalidad sea la esperada, es decir: la descrita por el cliente en las HU que se han implementado. (24)

Tabla 20. Prueba de aceptación No. 1

Prueba de Aceptación	
Código de prueba: P1.	**Número Historia de Usuario:** 1.
Nombre: Configurar carpeta de trabajo.	
Descripción: Prueba para verificar la funcionalidad configurar la carpeta de trabajo	
Condiciones de Ejecución: Debe estar iniciada la aplicación.	
Entrada/ Pasos de Ejecución: ➢ Una vez que el usuario entre en el sistema se dirige al menú configuración. ➢ El sistema muestra la interfaz correspondiente a dicho menú. ➢ El usuario selecciona la opción configurar carpeta de trabajo y selecciona la carpeta de trabajo en la cual se trabajará.	
Resultado Esperado: El sistema muestra la carpeta de trabajo y permite descargar en ella todos los archivos que serán bajados del servidor.	
Evaluación de la prueba: Satisfactoria	

Tabla 21. Prueba de aceptación No. 9

Prueba de Aceptación	
Código de prueba: P9.	**Número Historia de Usuario:** 11.
Nombre: Sincronizar archivo.	
Descripción: Prueba para verificar la funcionalidad sincronizar archivo.	
Condiciones de ejecución: Debe estar iniciada la aplicación y tener establecida la	

conexión con el servidor.
Entrada/ Pasos de Ejecución: ➢ El usuario entra en el sistema. ➢ El sistema muestra en el estado del archivo si ha sido modificado. ➢ El usuario puede subir el archivo modificado en caso de ser modificado localmente o descargarlo si fue modificado en el servidor.
Resultado Esperado: El sistema cambia el estado del archivo de modificado a sincronizado.
Evaluación de la prueba: Satisfactoria

Tabla 22. Prueba de aceptación No. 14

Prueba de Aceptación	
Código de prueba: P14.	**Número Historia de Usuario:** 16.
Nombre: Visualizar alertas.	
Descripción: Prueba para verificar la funcionalidad visualizar alertas.	
Condiciones de Ejecución: Debe estar iniciada la aplicación y tener establecida la conexión con el servidor.	
Entrada/ Pasos de Ejecución: ➢ El usuario al ejecutar la aplicación entra en su carpeta de trabajo.	
Resultado Esperado: El sistema muestra las notificaciones una vez que se produzcan cambios en el servidor.	
Evaluación de la prueba: Satisfactoria	

A continuación se realiza una descripción de las no conformidades detectadas durante la ejecución de las pruebas de aceptación:

En la primera iteración se encontraron un total de 12 no conformidades (NC):

- Faltas ortográficas 2 NC.
- Errores de validación 10 NC.

En la segunda iteración se encontraron un total de 6 no conformidades (NC):

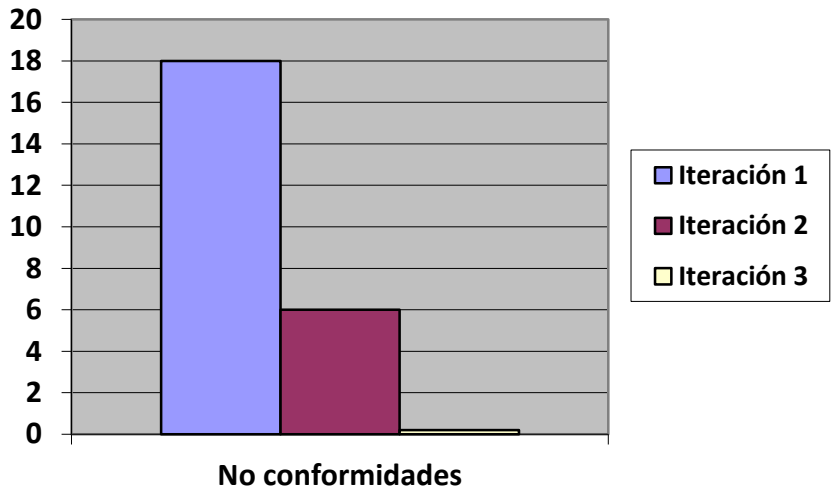
- Faltas ortográficas 2 NC.
- Errores de validación 4 NC.

En la tercera iteración no se encontraron no conformidades (NC):

- Faltas ortográficas 0 NC.
- Errores de validación 0 NC.

La gráfica que se muestra a continuación representa el resultado de estas pruebas según las NC detectadas durante su ejecución:

Fig.1. Resultados de las pruebas de aceptación



4.4 Conclusiones

En este capítulo se generaron los artefactos propuestos por la metodología XP para las fases de Implementación y Prueba. Se realizaron las tareas de ingeniería que dieron solución a todas las HU logrando una mayor organización y rapidez en el desarrollo del sistema. Se mostraron los resultados de las pruebas de aceptación realizadas para comprobar que se cumplieron todos los objetivos trazados.

CONCLUSIONES GENERALES

En el presente trabajo se arribaron a las siguientes conclusiones:

- Mediante la definición de los diferentes mecanismos de conexión con el servidor Nuxeo, de envío de notificaciones y los métodos de sincronización de archivos se obtuvieron las características necesarias para la selección de los mecanismos utilizados en la implementación de la herramienta.
- Con la selección de la metodología, herramientas y tecnologías a utilizar se contribuyó al desarrollo de la propuesta de solución.
- La correcta definición de los requisitos funcionales y no funcionales propició una correcta definición de las características de la herramienta.
- Con el diseño e implementación de la herramienta se resolvieron las limitantes planteadas en el problema.
- Se comprobó la completa y correcta realización de las funcionalidades requeridas durante el proceso de pruebas de aceptación cumpliendo así con los requisitos y necesidades del cliente.
- Con la realización de la herramienta se resolvieron los inconvenientes planteados anteriormente ya que contribuye a la sincronización automática de contenidos y a la visualización de alertas de cambios existentes mediante el trabajo colaborativo en la gestión documental con Nuxeo.

RECOMENDACIONES

- Agregar las funcionalidades avanzadas de Nuxeo.
- Implementar la herramienta utilizando el servicio SOAP.

REFERENCIAS BIBLIOGRÁFICAS

1. [Online] [Cited: 10 20, 2012.]
http://www.kyoceradocumentsolutions.es/index/about_us/glosario.html.
2. [Online] [Cited: noviembre 10, 2012.]
http://www.tecnologiapyme.com/software/nuxeo-gestor-documental-de-codigo-abierto.
3. **Campos, M., Cerda, C. and Rivera, R.** *Los Proyectos Colaborativos Interescolares en la Red.*
4. [Online] [Cited: noviembre 10, 2010.]
http://www.tecnologiapyme.com/software/nuxeo-gestor-documental-de-codigo-abierto.
5. [Online] http://docs.codehaus.org/display/ENUNCIATE/AMF+and+REST.
6. Git. [Online] [Cited: 1 10, 2013.] http://git-scm.com.
7. No. 1, 2012, Vol. Vol. 3.
8. **Carvajal Riola, José Carlos.** *Metodologías Ágiles: Herramientas y Modelo de desarrollo para aplicaciones.* España,Barcelona : s.n., 2008.
9. Cyta. [Online] [Cited: 2 20, 2013.] www.cyta.com.ar.
10. Ingeniería de Software. [Online] [Cited: 1 14, 2013.]
www.ingenieriadesoftware.mex.tl/52753_XP---Extreme-Programing.html.
11. FreeManager. [Online] [Cited: 1 14, 2013.]
www.freedownloadmanager.org/es/downloads/Paradigma_Visual
_para_UML_%28M%C3%8D%29_14720_p.
12. NetBeansIDE. [Online] [Cited: 1 20, 2013.] www.netbeans.org.
13. Spring. [Online] [Cited: 1 20, 2013.] www.spring.org.
14. **Ramos, Rafael Aldo.** compujuy. [Online] 2013. [Cited: Abril 10, 2013.]
http://www.compujuy.com.ar/.
15. **Bernd Bruegge, Allen H.Dutoit.** *Object Oriented Software Engineering.* 2000.
16. Cyta. [Online] [Cited: 3 1, 2013.] www.cyta.com.ar/ta0502/b_v5n2a1.htm.
17. **Gimson, Loraine.** Metodologías ágiles y desarrollo basado en conocimiento. 2012.
18. Cyta. [Online] [Cited: 3 2, 2013.] www.cyta.com.ar/ta0502/b_v5n2a1.htm.
19. El ingeniero de software. [Online] [Cited: 4 2, 2013.]
http://www.ingenierosoftware.com/analisisydiseno/patrones-diseno.php.
20. [Online] 4 2, 2013. geektheplanet.net/5462/patrones-gof.xhtml.
21. **Sandra I. Casas, héctor H. Reinaga.** *Un enfoque basado en las tarjetas CRC.* Argentina : s.n., 2009.
22. **Lisa Crispin, Tip House.** *Testing Extreme Programing.* s.l. : Addison Wesley, 2002.

23. **Pérez, Isaías Carrillo.** METODOLOGIA DE DESARROLLO DE SOFTWAR. 2005.

BIBLIOGRAFÍA

1. Brito Acuña, Karenny. eumed.net. *BIBLIOTECA VIRTUAL de Derecho, Economía y Ciencias Sociales.* [Online] 2012. [Cited: Diciembre 10, 2012.] http://www.eumed.net/libros-gratis/2009c/584/Metodologias%20tradicionales%20y%20metodologias%20agiles.htm.
2. Canós, José H., Letelier, Patricio and Penadés, María del Carmen. *Metodologías Agiles de Desarrollo de Software.* Valencia : s.n.
3. Joskowicz, Ing. Jose.*Reglas y Prácticas en eXtreme Programming.* España : s.n., 2008.
4. Hernández Orallo, Enrique. *El Lenguaje Unificado de Modelado (UML).* España : s.n., 2013.
5. Visual Paradigm for UML. Visual Paradigm for UML. *sitio web de Visual Paradigm for UML.* [Online] 2012. [Cited: Noviembre 25, 2012.] http://www.visual-paradigm.com/product/vpuml/.
6. Java.net. Java.net. *sitio web de The Source for Java Technology Collaboration.* [Online] 2013. [Cited: Junio 6, 2013.] https://javahelp.java.net/.
7. Redes Zone. Redes Zone. [Online] 2013. [Cited: Enero 19, 2013.] http://www.redeszone.net/2012/07/27/netbeans-7-2-nueva-version-de-este-popular-ide/.
8. Las Metodologías de Desarrollo Ágil como una Oportunidad para la Ingeniería del Software Educativo. Orjuela Duarte, Ailin and Rojas C., Mauricio. 2, Colombia : ISSN, 2008, Vol. 5
9. Reynoso, Carlos and Kiccillof, Nicolás.Estilos y Patrones en la Estrategia de Arquitectura de Microsft. Buenos Aires : s.n., 2004.
10. Buenas Tareas. [En línea] [Citado el: 15 de Mayo de 2013.] http://www.buenastareas.com/temas/pruebas-de-aceptaci%C3%B3n/20.
11. Gestión de Calidad y Pruebas de Software. [En línea] [Citado el: 9 de Mayo de 2013.] Disponible en: http://pruebasdesoftware.com/pruebadeaceptacion.htm
12. Grau Abalo, Ricardo, Correa Valdés, Cecilia y Rojas Betancourt, Mauricio. METODOLOGÍA DE LA INVESTIGACIÓN. [Documento] Ibagué : s.n.
13. Gutiérrez, J. J., y otros, y otros. PRUEBAS DEL SISTEMA EN PROGRAMACIÓN EXTREMA. [Documento] s.l. : Universidad de Sevilla.
24. Wigahluk. [En línea] [Citado el: 18 de Mayo de 2013.] http://wigahluk.wordpress.com/category/software-development/pruebas-unitarias.