UNIVERSIDAD CENTROCCIDENTAL
"LISANDRO ALVARADO"

DESARROLLO DE UNA HERRAMIENTA PARA LA GESTIÓN DE DOCUMENTOS y CONTENIDOS EN EL PROCESO ENSEÑANZA APRENDIZAJE DENTRO DEL DECANATO DE CIENCIAS Y TECNOLOGIA DE LA UCLA

RAMON ANTONIO VALERA ARANGUREN

Barquisimeto, 2005

UNIVERSIDAD CENTROCCIDENTAL “LISANDRO ALVARADO”
DECANATO DE CIENCIAS Y TECNOLOGÌA
POSTGRADO EN CIENCIAS DE LA COMPUTACIÓN

DESARROLLO DE UNA HERRAMIENTA PARA LA GESTIÓN DE DOCUMENTOS y CONTENIDOS EN EL PROCESO ENSEÑANZA APRENDIZAJE DENTRO DEL DECANATO DE CIENCIAS Y TECNOLOGIA DE LA UCLA

Trabajo presentado para optar al grado de
Magíster Scientiarum

Por: RAMON ANTONIO VALERA ARANGUREN

Barquisimeto, 2005

DESARROLLO DE UNA HERRAMIENTA PARA LA GESTIÓN DE DOCUMENTOS y CONTENIDOS EN EL PROCESO ENSEÑANZA APRENDIZAJE DENTRO DEL DECANATO DE CIENCIAS Y TECNOLOGIA DE LA UCLA

Por: RAMON ANTONIO VALERA ARANGUREN

Trabajo de grado aprobado

_____	_____
(Jurado 1)	**(Jurado 2)**
Tutor	

_____

(Jurado 3)

Barquisimeto, ___ de _____ de 2005

INDICE GENERAL

Página

INDICE DE TABLAS

Página

INDICE DE FIGURAS

Página

UNIVERSIDAD CENTROCCIDENTAL “LISANDRO ALVARADO”
DECANATO DE CIENCIAS Y TECNOLOGIA
POSTGRADO EN CIENCIAS DE LA COMPUTACIÓN

DESARROLLO DE UNA HERRAMIENTA PARA LA GESTIÓN DE DOCUMENTOS y CONTENIDOS EN EL PROCESO ENSEÑANZA APRENDIZAJE DENTRO DEL DECANATO DE CIENCIAS Y TECNOLOGIA DE LA UCLA

Autor(a): Ramón Antonio Valera Aranguren

Tutor(a): Maritza Bracho

RESUMEN

En la sociedad del conocimiento, las organizaciones avanzan gracias a la difusión, asimilación, aplicación y sistematización de conocimientos creados y obtenidos localmente, o accesados desde el exterior. Ellas necesitan de procesos que incentiven la creación, uso y difusión del conocimiento. En este contexto, es fundamental el papel que desempeñan las tecnologías de la información, dada su habilidad para potenciar la comunicación, colaboración y búsqueda de conocimiento a lo largo de la organización. Las tecnologías de información y la ingeniería de software han contribuido a crear sistemas de gestión de conocimiento, que permiten formar un extenso y homogéneo repositorio de documentos estructurados o semiestructurados, en el que el conocimiento organizacional queda explícito, organizado y representado de acuerdo a un único sistema de significados. En este sentido surgen alternativas como los sistemas de gestión de contenidos y documentos basados en Web, que facilitan a los miembros de una organización el proceso de creación, intercambio, publicación y búsqueda de documentos, además, establece los niveles de seguridad necesarios, para que el conocimiento explicito plasmado en los documentos solo este disponible a personas autorizadas. La presente investigación se centra en aplicar la metodología de desarrollo ágil ‘eXtreme Programming’ (XP) y técnicas de ingeniería de software para el desarrollo de una herramienta de gestión de contenidos basada en tecnología Web, que se pueda usar para mejorar la comunicación y el intercambio de conocimiento entre profesores y alumnos, dentro del Decanato de Ciencias y Tecnología.

Palabras Claves: Gestión de Conocimiento, Sistema de Gestión de Contenido, Gestión de Documentos, Distribución de Contenido WEB.

INTRODUCCION

La nueva Sociedad del Conocimiento es una sociedad dinámica, en donde las comunidades, empresas y organizaciones que la conforman avanzan gracias a la difusión, asimilación, aplicación y sistematización de conocimientos creados u obtenidos localmente o desde el exterior, de allí la necesidad de fomentar estrategias que permitan gestionar de manera eficiente el flujo de conocimiento dentro de cada una de ellas, es así como surge el concepto de Gestión de Conocimiento.

La gestión de conocimiento involucra procedimientos y políticas para ayudar a generar, organizar y apoyar el conocimiento en contextos sociales. Estas estrategias están influenciadas ampliamente por el uso de las tecnologías de información y comunicaciones; estas han propiciado la creación de variadas herramientas de software a las que se les ha llamado Sistemas de Gestión de Conocimiento.

Los Sistemas de Gestión del Conocimiento deben realizar funciones específicas para apoyar la generación del conocimiento por parte de las personas, así como su almacenamiento, recuperación y representación. En este sentido surgen alternativas como los sistemas de gestión de contenidos y documentos basados en Web, que facilitan a los miembros de una organización el proceso de creación, intercambio, publicación y búsqueda de documentos, además, establece los niveles de seguridad necesarios, para que el conocimiento explicito plasmados en los documentos solo este disponible a personas autorizadas.

La presente investigación hace uso de una metodología de desarrollo ágil, para producir un sistema de gestión de contenidos y documentos, que sea eficiente y mantenible, empleando para ello técnicas y métodos de Ingeniería de Software. El producto final podrá ser usado en el proceso enseñanza-aprendizaje en el Decanato de Ciencias y Tecnología con la intención de mejorar el flujo de conocimientos entre

profesores y alumnos. De esta manera el estudio se encuentra dividido en capítulos, los cuales se detallan a continuación:

Capitulo I El Problema. Se inicia con el planteamiento de la situación que genera el trabajo investigativo, así mismo se describen los objetivos del mismo, la justificación e importancia, alcances y limitaciones representando estos aspectos las bases que conducen la investigación.

Capitulo II Marco Teórico. Presenta los antecedentes y las bases teóricas y filosóficas que soportan el conocimiento del tema en estudio.

Capitulo III Marco Metodológico. Describe los aspectos metodológicos que conducirán a la realización del estudio diagnóstico, se presenta el análisis de la información recopilada a través de entrevistas. Seguidamente, se presentará un estudio factible de la alternativa de solución que permita la gestión de contenidos y documentos en el proceso enseñanza-aprendizaje en el Decanato de Ciencias y Tecnología.

Capitulo IV Propuesta del Estudio. Dado el estudio diagnostico y de factibilidad del capitulo anterior, se presentan todas las fases de la metodología XP, que fueron necesarias para producir el software de gestión de contenido y documentos OBELISCO, en este capitulo se presentan los principales artefactos UML que describen las diversas vistas del sistema en estudio.

Finalmente el Capitulo V Conclusiones y Recomendaciones. Ofrece un conjunto de conclusiones obtenidas del estudio y algunas recomendaciones enfocadas al software de gestión de contenido y documentos OBELISCO.

CAPITULO I

EL PROBLEMA

Planteamiento del Problema

Las organizaciones de hoy en día se ven en la imperiosa necesidad de adecuarse y adaptarse al contexto cambiante y dinámico donde se desenvuelven, como consecuencia de una sociedad globalizada donde el conocimiento es parte íntegra de las actividades cotidianas y donde las tecnologías de información y comunicaciones son un catalizador para el desarrollo eficiente de cualquier actividad económica y social.

Drucker (1993), en Rojas (2000), hace referencia a esta nueva etapa de la sociedad y la caracteriza diciendo, que el recurso económico básico en las organizaciones de hoy es el conocimiento, hay un cambio de paradigma en lo que se refiere a las actividades de creación de riquezas. Estas ya no están únicamente centradas por la colocación de capital para su uso productivo y de trabajo, si no que van más allá y centran sus actividades en la productividad e innovación. En esta nueva etapa se destacan el grupo de los trabajadores del conocimiento, ejecutivos del conocimiento, profesionales del conocimiento y emprendedores del conocimiento quienes tienen la comprensión y la visión de la sociedad para dar uso productivo al conocimiento. El desafío económico de la nueva sociedad será en consecuencia la productividad del trabajo del conocimiento y del trabajador del conocimiento.

En este nuevo contexto, se hace evidente la necesidad de gestionar bien los procesos que incentiven la creación, uso y difusión del conocimiento. Estos procesos se convierten en la tarea primordial de cualquier organización inmersa en un mundo en constante modificación. Surge así el concepto de la Gestión del Conocimiento entendida como el proceso que continuamente asegura el desarrollo y aplicación de todo tipo de conocimientos pertinentes en una organización, con objeto de mejorar su capacidad de resolución de problemas y así contribuir a la sostenibilidad de sus ventajas competitivas. En otras palabras, como el arte de transformar la información y los activos intangibles en un valor constante para sus clientes y para su personal.

La gestión de conocimiento involucra personas, procesos, actividades, tecnología y un vasto ambiente que permite la identificación, creación, comunicación y uso del conocimiento individual y organizacional. Se relaciona a los procesos que dirigen la creación, distribución y utilización de conocimiento para lograr los objetivos organizacionales. El poder de la gestión de conocimiento está en permitir a las organizaciones activar y soportar explícitamente estas actividades para apalancar el valor de los grupos y de los individuos. Esto requiere una mezcla de conocimiento del negocio, actitudes creativas y prácticas, sistemas, herramientas, políticas y procedimientos, diseñados para liberar el poder de la información y de las ideas.

En este contexto, la importancia de las tecnologías de información y comunicaciones radica en su habilidad para potenciar la comunicación, la colaboración y la búsqueda, la generación de información y conocimiento. No se debe olvidar, sin embargo, que el objetivo más importante –cuando se hace referencia a gestión del conocimiento – es el de conseguir un entorno de trabajo que sea colaborativo y que esté dotado de una constante vocación de aprendizaje. Se trata de lograr un entorno laboral en el que los trabajadores puedan realizar todo tipo de actividades de aprendizaje y compartir el conocimiento adquirido con sus compañeros, clientes y socios.

En organizaciones generadoras de conocimiento como las universidades, la gestión de conocimiento se ha convertido en el motor para la innovación y la supervivencia. Estas deben adaptarse a este nuevo entorno dinámico e inestable donde se mueven. La Universidad Centroccidental "Lisandro Alvarado" - UCLA - no escapa a esta realidad y una manera de adecuar la institución a esas necesidades y cambios que demanda el ambiente es promover y estimular la gestión eficiente del conocimiento, que reside fundamentalmente en los miembros de la organización.

En lo que respecta a la producción y distribución de los conocimientos generados dentro de la UCLA en las áreas de investigación, extensión y docencia, merece particular atención el proceso de enseñanza aprendizaje por el volumen de conocimientos producidos. En el logro de la excelencia, el docente debe garantizar que sus conocimientos se encuentren disponibles el mayor tiempo posible a sus alumnos.

La importancia que ha adquirido la gestión del conocimiento, ha originado la aparición de nuevas herramientas de software. Estas herramientas proporcionan los medios para la estructuración del conocimiento individual de los usuarios hacia el conocimiento colectivo de la comunidad, facilitando su distribución y acceso.

Entre estas herramientas podemos nombrar a los sistemas de gestión de conocimientos basados en documentos o sistemas de gestión de contenidos, estos sistemas crean un largo repositorio de documentos organizados en base a un sistema de significados, admitiendo relaciones entre ellos y brindando mecanismos de búsquedas poderosos y sencillos.

En este sentido Zhan y Chen (1997) hablan de las potencialidades de la tecnología Web y se refieren a ella como el medio más idóneo para soportar la naturaleza de la gestión de conocimiento, destacan así mismo que los elementos que han permitido esto son:

- Dependencia del protocolo TCP/IP, un protocolo muy maduro y ampliamente difundido en la industria.
- La información puede ser recolectada, recibida y compartida a través de un Web Browser, que pueden correr en una gran cantidad de plataformas de sistemas operativos.

En la actualidad el desarrollo de las aplicaciones de gestión de contenidos y documentos, se realizan en el contexto de la Ingeniería de Software (IS), disciplina que trata los aspectos concernientes al desarrollo de sistemas de software complejos y de calidad.

Coltell y Arregui (2004), conceptualiza la Ingeniería del Software, como una disciplina que trata los aspectos concernientes al desarrollo de sistemas de software, que requiere para su construcción de: normas, estándares, procesos rigurosos, sistemáticos y controlables mediante modelos y métodos.

La IS armoniza el trabajo en equipo dentro del grupo de desarrolladores, y direccióna su trabajo a través de la confección de metodologías de desarrollo, Kenneth y Laudon (1996), afirman que una metodología de desarrollo del software se define de la siguiente manera: "Colección de métodos, uno o más para cada actividad dentro de cada fase del desarrollo del proyecto". Así mismo, Pressman (2001), describe los 3 elementos de la Ingeniería del Software como:

- Métodos que indican como se debe construir el software.

- Herramientas que dan soporte al desarrollo del software.

- Procesos y metodologías, que corresponde a la unión entre los métodos y herramientas que definen la secuencia en que se aplican los métodos, las

entregas que se requieren, los controles necesarios para asegurar la calidad y la coordinación de los cambios.

La presenta investigación se centra en usar una metodología de desarrollo de software para producir una herramienta de gestión de contenido, que pueda ser usada para organizar el conocimiento de una comunidad de usuarios. En este proceso de desarrollo se recurre a distintas técnicas y métodos de IS, como Arquitecturas de Software y Lenguajes de Patrones, que le permitan al desarrollador ahorrar esfuerzos y direccionar sus actividades, para lograr un software de calidad.

La metodología de desarrollo que dirige las actividades de construcción de software es 'eXtreme Programming' (XP), ella sugiere un desarrollo incremental, a través de la planificación de distintas iteraciones, cada iteración finaliza con un período de prueba y validación, que garantiza al cliente, la exactitud en las funcionalidades y la satisfacción de sus necesidades. Cada iteración además de producir componentes de software, produce documentos basados en el lenguaje de modelado UML, que denotan la estructura de la aplicación y la iteracción entre sus componentes.

El sistema se desarrolló pensando en las posibilidades de mostrar los documentos en distintos formatos, dada la cantidad de dispositivos que en la actualidad están conectados y que en un futuro pudieran necesitar trabajar con el repositorio de documentos, en este sentido es importante desaclopar el contenido del documento a su presentación. Para dar solución a este problema se recurrió a un patrón arquitectural Modelo-Vista-Controlador MVC, que permitió dividir la aplicación en tres elementos funcionales, y con ello desacoplar la presentación, del resto del sistema.

Dada la necesidad de usar un patrón arquitectural MVC en el desarrollo del Sistema, es imprescindible buscar APIs o Frameworks que implementen el citado

patrón, y de esa manera ayuden a acelerar el tiempo de entrega del software. Así mismo es importante ubicar herramientas, probadas, abiertas y documentadas que se usen en el desarrollo de los procesos de búsquedas del sistema.

Se ha observado como a pesar de contar con una plataforma tecnológica, que se encarga de conectar a todos los nodos y dependencias que forman parte de la UCLA, hay una ausencia de políticas que permitan integrar de manera coherente los trabajos de docencia que se desarrollan en los distintos decanatos y núcleos de la institución, así mismo no existe un mecanismo de búsquedas único que facilite ubicar información a lo largo de los distintos repositorios de documentos de la universidad.

Para dar solución a la problemática planteada, se aplicara el sistema de gestión de contenidos descrito anteriormente, con la intención de mejorar la distribución de documentos y contenido relacionados al proceso enseñanza aprendizaje usando la tecnología de red disponible dentro de la universidad, y minimizando las barreras tecnológicas para sus usuarios finales, permitiendo la búsqueda y acceso eficiente a documentos e información.

Objetivos

Objetivo General

Usar la metodología de desarrollo ágil 'eXtreme Programming' XP, en conjunto con técnicas y métodos de Ingeniería de Software, para la construcción de una herramienta de gestión de contenido y documentos, que pueda ser aplicada al proceso enseñanza-aprendizaje en el Decanato de Ciencias y Tecnología de la UCLA.

Objetivos Específicos

1. Realizar un diagnostico de las necesidades y requerimientos relacionados a una herramienta de distribución de contenido y documento, que pueda ser usada en el proceso enseñanza-aprendizaje dirigido por los profesores del departamento de sistemas en el Decanato de Ciencias y Tecnología.
2. Determinar la arquitectura del sistema que permitirá la gestión de documentos y contenido.
3. Desarrollar la herramienta de gestión de contenido, que pueda introducirse en el proceso enseñanza aprendizaje dentro del Decanato de Ciencias y Tecnología.

Justificación e Importancia

En la nueva sociedad del conocimiento la gestión del conocimiento es una actividad crítica y estratégica que facilita la supervivencia en un mundo dinámico y cambiante. Esta gestión del conocimiento debe ser incorporada dentro de todos los procesos de la UCLA, particularmente en aquellos como el proceso enseñanza aprendizaje donde el manejo y generación intensiva de contenido y documentos es frecuente. Los distintos profesores de la Universidad deben lograr que sus conocimientos e innovaciones - expresados de manera explicita en documentos y contenido - se den a conocer a la mayor cantidad de personas posibles que forman parte de la sociedad donde esta inmersa, y recibir de ellos una retroalimentación que garantice que se va en el rumbo adecuado.

El desarrollo de un sistema de gestión de documentos y contenido para el Decanato de Ciencias y Tecnología, cuya construcción se haga bajo los parámetros de una metodología, en el ánimo de lograr un producto de calidad y eficiencia, sería un aporte interesante para los miembros de la comunidad universitaria, puesto que facilitará la transferencia de experiencias y conocimientos. Esto por un lado abre las puertas para que cada miembro de la comunidad universitaria conozca las actividades desarrolladas por otros miembros de la comunidad, incrementando los debates y discusiones sobre determinados temas; por otro lado con una estructuración adecuada del conocimiento miembros de la sociedad y de la misma comunidad universitaria pueden buscar y contrastar conocimientos de expertos, en distintas áreas de trabajo, de una manera automatizada, eficiente y expedita.

Esto último es confirmado por Lehaney (2003) quien habla de las distintas formas en las que puede ayudar a la universidad una estrategia de gestión de conocimiento, al respecto explica que permite:

- Dar soporte a la innovación, a la generación de nuevas ideas, y a la explotación del poder de pensamiento de la organización.
- Capturar la visión y experiencias de los miembros de la organización, para que se encuentren disponibles, en cualquier momento y lugar por quien lo requiera.
- Hacer fácil el proceso de encontrar y reusar las fuentes conocimiento.
- Mejorar la calidad del proceso de toma de decisiones y otras tareas de inteligencia.
- Entender el valor y las contribuciones del capital intelectual e incrementar su valor, efectividad y explotación.

La UCLA tiene una plataforma tecnológica que es importante aprovechar y a la que se le debe dar valor agregado, un modelo como el desarrollado incrementará la retroalimentación entre la comunidad universitaria y el conjunto de organizaciones y personas que tienen contacto directo con la universidad, así mismo explotará grandemente los recursos tecnológicos con los que se dispone.

Alcances y Limitaciones

Para desarrollar el sistema de gestión de contenidos y documentos, se aplico la metodología de desarrollo XP. Esta delimita la secuencia de actividades necesarias para construir un producto de software de manera incremental e iterativa.

El desarrollo comienza con una fase exploratoria, en la que se recogen los requerimientos funcionales y no-funcionales del sistema, estos son usados posteriormente en la fase de planificación para la construcción de una lista de tareas, que bajo una asignación de prioridades sirven para la planificación de iteraciones o de entrega de versiones.

En cada iteración de construye componentes de software, que deben satisfacer la lista de tareas en base a la cual se planificó. En este sentido existe una iteración muy particular y es la llamada iteración 0, es aquí donde se definirá la arquitectura de sistema, y con el uso de lenguaje de patrones se expondrá el comportamiento y diseño de cada uno de sus componentes.

Para este trabajo de investigación se ejecutaron tres iteraciones, que sirven para medir el comportamiento y desenvolvimiento de la metodología XP, y para emplear técnicas de IS que conlleven a lograr un software eficiente y de calidad.

Este proyecto se circunscribe solo al Decanato de Ciencias y Tecnología de la UCLA por ser el Decanato donde la gente esta más familiarizada con la importancia y ventajas de las tecnologías de información.

CAPITULO II

MARCO TEORICO

ANTECEDENTES

La sección de antecedentes de la presente investigación comienza describiendo los componentes y principales características de la llamada sociedad del conocimiento, usando la opinión e investigaciones de distintos autores. En el marco de la sociedad del conocimiento, se hará hincapié en la importancia de una estrategia de gestión de conocimiento, como elemento para la supervivencia de las organizaciones en el entorno dinámico y cambiante que caracteriza a esta nueva sociedad.

En una estrategia de gestión del conocimiento, el factor tecnológico es preponderante, la tecnología ha propiciado el origen de nuevas herramientas que contribuyen a una mejor distribución y socialización del conocimiento, en este grupo aparecen los Sistemas de Manejo de Contenidos.

Para dar inicio a estos antecedentes, se expondrá el concepto que con respecto a "Sociedad del Conocimiento" tienen algunos investigadores como Nonaka y Takeuchi (1995), Choo (1998). Ellos indican que la nueva Sociedad del Conocimiento es una sociedad con capacidad para generar, apropiar, y utilizar el conocimiento para atender las necesidades de su desarrollo y así construir su propio

futuro, convirtiendo la creación y transferencia del conocimiento en herramienta de la sociedad para su propio beneficio.

En esta nueva Sociedad del Conocimiento destacan dos características:

- La conversión del Conocimiento es factor crítico para el desarrollo productivo y social.
- El fortalecimiento de los procesos de Aprendizaje Social asegura la apropiación social del conocimiento y su transformación en resultados útiles.

En este nuevo contexto, se hace evidente la necesidad de gestionar bien los procesos que incentiven la creación, uso y difusión del conocimiento. Ésto se convierte en una tarea primordial para cualquier organización inmersa en un mundo en constante modificación. Surge así el concepto de la Gestión del Conocimiento, Andreu y Sieber (1999) citados por Barnes (2002), lo entienden como el proceso que continuamente asegura el desarrollo y aplicación de todo tipo de conocimientos pertinentes en una organización, con objeto de mejorar su capacidad de resolución de problemas y así contribuir a la sostenibilidad de sus ventajas competitivas.

La Gestión de Conocimiento involucra personas, procesos, actividades, tecnología y un vasto ambiente que permite la identificación, creación, comunicación y uso del conocimiento individual y organizacional. Se relaciona a los procesos que dirigen la creación, distribución y utilización de conocimiento para lograr los objetivos organizacionales. El poder de la gestión de conocimiento esta en permitir a las organizaciones activar y soportar explícitamente estas actividades para apalancar el valor de los grupos y de los individuos. Esto requiere una mezcla de conocimiento del negocio, actitudes creativas y prácticas, sistemas, herramientas, políticas y procedimientos, diseñados para liberar el poder de la información y de las ideas.

Por ello, todo modelo de gestión del conocimiento estará típicamente basado en la codificación del conocimiento explícito y en la difusión y socialización del

conocimiento tácito, Nonaka y Takeuchi, (1995). La codificación se fundamenta en almacenar conocimiento explícito, de manera que éste pueda ser utilizado con posterioridad. Por su parte, la difusión y socialización del conocimiento tácito consiste en fomentar la comunicación entre los individuos que componen la organización a fin de que se vuelva colectivo su conocimiento individual.

En este contexto la tecnología, proporciona un medio eficiente y económico para que fluya el conocimiento a los distintos niveles de la Organización. Convirtiéndose en un elemento de importancia para compartir el conocimiento, ofreciendo nuevas oportunidades de negocio y estrategias para mejorar su desempeño.

Es así como muchas organizaciones se esfuerzan en construir sus propias soluciones informáticas, para que el conocimiento fluya entre los distintos niveles de la organización, esto implica el uso de distintas tecnologías, herramientas y metodologías. Al respecto un estudio realizado por Daventport y Prusak (1999) destaca que a pesar de la heterogeneidad entre las soluciones desarrolladas, ellas comparten algunas características como:

- Instalación de Intranets corporativas para asegurar el acceso a la información desde potencialmente cualquier parte.
- Creación de repositorios homogéneos y ubicuos accesibles desde distintos dispositivos que contienen el conocimiento organizacional, con el objetivo de formar un punto central de acceso en forma de portal.
- Creación de comunidades virtuales que representan los lugares de donde fluye el conocimiento, en donde sus miembros intercambian experiencias, habilidades y destrezas.
- Aparición de un nuevo rol, El Administrador de Conocimiento, que se encarga de facilitar la interacción entre las distintas comunidades.
- Diseño de un lenguaje corporativo, que representa en una forma estándar y común el conocimiento de la empresa.

- Diseño de procesos de contribución de conocimiento, por medio del cual los miembros de una organización usando el lenguaje corporativo codifican su conocimiento tácito.

Es importante destacar la tendencia actual de los sistemas de gestión de conocimiento, consistente en formar un extenso y homogéneo repositorio de documentos estructurados o semiestructurados ricos en recursos multimedia, en el que el conocimiento organizacional quede explícito, organizado y representado de acuerdo a un único sistema de significados. Este sistema de significados frecuentemente se llama mapa de conocimiento, ontología, categorización o sistema de clasificación y sirve para representar los conceptos y sus relaciones dentro de la memoria organizacional, permitiendo la comunicación y el intercambio de conocimiento de una manera estándar a lo largo de la organización.

Tal como lo señalan Davenport, De Long y Beers (1998), un factor que garantiza el éxito de un proyecto de gestión de conocimiento, es el uso de una infraestructura tecnológica que permita la compatibilidad y la interoperabilidad entre los distintos sistemas. En este sentido Zhan y Chen (1997) ven a la tecnología Web como altamente interoperable y lo justifican indicando algunas de sus características más resaltantes:

- La tecnología de la Web usa un protocolo estándar TCP/IP, el cual es muy maduro y soportado por la mayoría de los vendedores.
- La información se puede recoger, recibir y compartir a través de los más populares Web Browsers como Netscape Navigator e Internet Explorer.
- Una página Web se puede desarrollar, distribuir y compartir rápidamente.
- Existen lenguajes especialmente desarrollados para la Web, como Java.
- Con la tecnología Web, una organización disfruta de la independencia de la plataforma. No se necesita rescribir el código que se usa en el lado del cliente.

La web facilita la publicación de los documentos gracias a su protocolo abierto HTTP, Zhan y Chen (1997). Sin embargo la producción de documentos profesionales conlleva a elevados costos de creación en cuanto a tiempo y esfuerzo. En este sentido y como alternativa a la reducción de costos, surgen los sistemas de manejo de contenidos basados en la Web, estos se convierten en una opción económica para la generación de documentos profesionales y para mantener la coherencia y exactitud de un sitio Web, en donde la organización pueda de una manera eficiente obtener los documentos, aplicando procesos de búsqueda.

BASES TEORICAS

En esta sección se presentara un conjunto de conceptos y definiciones, con el fin de establecer un marco teórico que sustentara el desarrollo de la actual investigación. Se comenzará por definir datos, información y conocimiento, como elementos críticos y dinamizadores en la llamada Sociedad del Conocimiento.

En relación a esta nueva Sociedad del Conocimiento es importante conocer sus elementos y rasgos más importantes, y entender el significado e importancia de las estrategias de gestión de conocimiento, como recurso para la supervivencia de las organizaciones en esta Sociedad dinámica.

La Gestión de conocimiento, involucra procesos, tecnologías y personas; surgen así herramientas informáticas, llamadas sistemas de gestión de conocimiento, que tienden a mejorar el flujo del conocimiento a lo largo de la organización, en este sentido es compresible la importancia de definirlos, caracterizarlos y conocer su arquitectura.

Como un tipo especial de sistema de gestión de conocimiento, surgen los sistemas de gestión de contenidos, caracterizados por aprovechar la interoperabilidad de sistemas brindada por la tecnología de Internet.

Los sistemas de gestión de contenido, son herramientas que se centran en definir un proceso sencillo, amigable y escalable para que los miembros de la organización, codifiquen conocimiento y los distribuyan a otros usando generalmente herramientas de la Web.

Teniendo en cuenta que el objetivo de este estudio es el uso de la metodología de desarrollo ágil XP, se describirá lo que es una metodología ágil y en específico lo que significa eXtreme Programming.

A lo largo de la metodología XP, es necesario expresar características del sistema en términos de Arquitectura de Software, Diagramas UML, Patrones Arquitecturales y Patrones de Diseño, es por ello necesario conocer todos estos términos y de que manera se usan dentro del proceso de desarrollo.

En este apartado se procederá a definir la arquitectura J2EE y el Frameworks para el desarrollo de aplicaciones Web, STRUTS, que forman pilares fundamentales en la implementación de la solución de Software.

Datos, Información y Conocimiento

Según Davenport y Prusak (1999), un Dato es un conjunto discreto, de factores objetivos sobre un hecho real. Dentro de un contexto organizacional, el concepto de dato es definido como un registro de transacciones. Un dato no dice nada sobre el por que de las cosas, y por sí mismo tiene poca o ninguna relevancia o propósito. Por su

parte Cornella (2000), lo define como el resultado aplicar medidas a hechos concretos, o de valores aplicados unívocamente a acontecimientos u objetos. Un dato cumple con las siguientes características:

- Son perfectamente identificables, sin posibilidad de confusión, por un conjunto de símbolos (básicamente letras y números).
- Son contrastables, es decir se puede determinar, con precisión, si el dato es cierto o no.
- Gozan de un elevado nivel de estructura, lo que hace posible, que sea fácil de: capturar, almacenar y transmitir mediante tecnologías de información.

Davenport y Prusak (1999), definen a la información como un mensaje, normalmente bajo la forma de un documento o algún tipo de comunicación audible o visible:

- Tiene un emisor y un receptor.
- La información es capaz de cambiar la forma en que el receptor percibe algo, es capaz de impactar sobre sus juicios de valor y comportamientos.
- La información se mueve entorno a las organizaciones a través de redes formales e informales.
- A diferencia de los datos, la información tiene significado, relevancia y propósito.
- No sólo puede formar potencialmente al que la recibe, sino que esta organizada para algún propósito.
- Los datos se convierten en información cuando su creador les añade significado.

Para Davenport y Prusak (1999) el conocimiento es una mezcla de experiencia, valores, información y "saber hacer" que sirve como marco para la incorporación de nuevas experiencias e información y es útil para la acción. Se origina y aplica en la mente de los conocedores. En las organizaciones con frecuencia no sólo se encuentra

dentro de documentos o almacenes de datos, sino que también esta en rutinas organizativas, procesos, prácticas y normas.

Nonaka y Takeuchi (1995) sugieren una clasificación para el conocimiento, éste puede ser tácito o explicito. El conocimiento tácito hace referencia a las creencias y valores que son difíciles de expresar, pero se deduce del comportamiento de los miembros de la organización. El conocimiento explícito se puede expresar con facilidad, por ejemplo la formalización de una rutina o de un proceso organizativo a través de un diagrama de flujo. El conocimiento organizacional e individual se crea a través de un dialogo continuo entre el conocimiento tácito y explicito de los individuos. Nonaka y Takeuchi (1995) reconocen cuatro procesos de transformación del conocimiento: socialización (de tácito a tácito), interiorización (de explícito a tácito), externalización (de tácito a explícito) y combinación (de explícito a explícito).

Sociedad del Conocimiento

Una sociedad de conocimiento como lo dice Arráez (1999), es una sociedad con capacidad para generar, apropiar, y utilizar el conocimiento para atender las necesidades de su desarrollo y así construir su propio futuro, convirtiendo la creación y transferencia del conocimiento en herramienta para su propio beneficio.

En la sociedad del conocimiento, las comunidades, y organizaciones avanzan gracias a la difusión, asimilación, aplicación y sistematización de conocimientos creados y obtenidos localmente, o accedidos del exterior. El proceso de aprendizaje se potencia en común, a través de redes, gremios, comunicaciones inter e intrainstitucional, entre comunidades y países. Una sociedad del conocimiento representa una nación y unos agentes económicos más competitivos e innovadores.

De acuerdo a Arráez (1999), una Sociedad del Conocimiento tiene dos características principales:

- Conversión del Conocimiento en factor critico para el desarrollo productivo y social.
- Fortalecimiento de los procesos de aprendizaje social del conocimiento y su transformación en resultados útiles, en donde la Educación juega un papel central.

Gestión de Conocimiento

Carrillo (1999) en Rojas (2000) indica que la gestión de conocimiento (GC) es la aplicación sistemática del entendimiento científico acerca del conocimiento, como una estrategia deliberada de los individuos, las organizaciones o las sociedades para optimizar la producción de valor.

En este mismo sentido la UTEXAS (1999) referenciado por Rojas (2000), a través de su sitio en el World Wide Web define a la GC como un proceso sistemático de encontrar, seleccionar, organizar, destilar y presentar información de forma que mejora la comprensión del empleado en un área específica de interés. La GC permite a la organización ganar profundidad y comprensión de su propia experiencia. Las actividades específicas de la GC ayudan a enfocar a la organización en la adquisición, almacenamiento y utilización del conocimiento para tareas tales como resolución de problemas, aprendizaje dinámico, planeación estratégica y toma de decisiones.

En función a las definiciones presentadas anteriormente podemos inferir que la GC se basa en conceptos y teorías que ayudan a generar, organizar y apoyar el conocimiento en contextos sociales. No existe una sola definición de GC pero, en

general, la idea está relacionada con la captación y el uso del conocimiento de los individuos para que esté disponible como un recurso organizativo independiente.

Sistemas de Gestión de Conocimiento

Ruggles (1997) citado por Rivero (2000) define a los Sistemas de Gestión de Conocimiento como herramientas tecnológicas que aumentan y permiten la generación, codificación y transferencia del conocimiento. Los Sistemas de Gestión del Conocimiento deben realizar funciones específicas para apoyar la generación del conocimiento por parte de las personas, así como su almacenamiento, recuperación y presentación.

Sistema de Manejo de Contenido

En Howard (2003) se define un sistema de manejo de contenido Web, afirmando que este consiste típicamente de dos elementos, una Aplicación que Administra el Contenido (AAC), y una Aplicación que Distribuye el Contenido (ADC). El AAC permite a un autor, sin necesidad de conocer HTML, crear, modificar y remover contenido de un sitio Web. El ADC usa y compila la información que se deposita en el Sitio Web para distribuirla a todos los miembros de una organización. Las características de un sistema SMC varían, pero la mayoría incluyen publicación, manejo de formatos de presentación, control de versiones, indexación, búsqueda y recepción de información.

Otra visión de SMC muy orientada al proceso esta plasmada en Howard (2003), alli expresa la necesidad de un sistema para crear el contenido, describirlo, cambiarlo y actualizarlo, permitiendo a múltiples personas participar en su edición, permitiendo

que las personas correctas puedan hacer las operaciones correctas, evitando que personas no autorizadas lo manipulen, manteniendo una traza de cómo éste es cambiado, mostrándolo en un forma controlada, coherente, y estandarizada.

Un SMC es capaz de:

- Definir niveles de acceso, para la publicación y consulta de documentos.
- Almacenar y ordenar todos los tipos de documentos, incluyendo gráficos, audio y video en un repositorio central.
- Crear un flujo de trabajo entre las personas que crean documentos y los responsables de su publicación.
- Cambiar la apariencia general del formato del sitio, a través de la definición y reuso de plantillas.
- Agregar nuevas categorías de contenido y metadata a los documentos antes y después de que son almacenados en la base de datos de documentos.

Existen trabajos como los de Robertson (2002) que indican que el Conocimiento organizacional no se encuentra por sí solo en el contenido de un SMC, por el contrario este esta inmerso en los procesos y las prácticas más comunes relacionadas al trabajo de un SMC.

Arquitectura de Software

Una definición reconocida sobre arquitectura de software es la de Clements (1996), donde afirma que una arquitectura es una vista del sistema que incluye los componentes principales del mismo, la conducta de esos componentes según se la percibe desde el resto del sistema y las formas en que lo componentes interactúan y coordinan para alcanzar la misión del sistema. La vista arquitectónica es una vista

abstracta, aportando el más alto nivel de comprensión y la supresión o diferimiento del detalle inherente a la mayor parte de las abstracciones.

Patrón

En el sentido más general, un patrón es un modelo a seguir que describe una solución exitosa de un problema particular en un contexto dado.

Zambrano y Acosta (2001) se refieren a un patrón desde el punto de vista informático, como una descripción de una solución computacional exitosa a un problema recurrente en un contexto particular; esta solución expresa la experiencia y el conocimiento de expertos y, en este sentido, sirve como medio de comunicación para difundir este conocimiento.

Un patrón soluciona un problema recurrente, ya que si el problema es poco o nada frecuente, entonces pudiera no tener sentido construir un patrón de su solución, el cual no se utilizará más. De allí que, una de las ventajas de los patrones es fomentar y facilitar el reuso, en este caso, de soluciones.

Patrón de Arquitectura

Según Welicki (2005) los patrones de arquitectura expresan el esquema fundamental de organización para sistemas de software. Proveen un conjunto de subsistemas predefinidos; especifican sus responsabilidades e incluyen reglas y guías para organizar las relaciones entre ellos. Los patrones de arquitectura representan el nivel más alto en el sistema de patrones propuesto en Buschmann (1996) y ayudan a especificar la estructura fundamental de una aplicación.

Patrón de Arquitectura Modelo-Vista-Controlador (MVC)

El Modelo-Vista-Controlador (MVC) fue introducido inicialmente en la comunidad de desarrolladores de Smalltalk-80. MVC divide una aplicación interactiva en 3 áreas: procesamiento, salida y entrada. Para esto, utiliza las siguientes abstracciones:

- Modelo (Model): Encapsula los datos y las funcionalidades. El modelo es independiente de cualquier representación de salida y/o comportamiento de entrada.
- Vista (View): Muestra la información al usuario. Obtiene los datos del modelo. Pueden existir múltiples vistas del modelo. Cada vista tiene asociado un componente controlador.
- Controlador (Controller): Reciben las entradas, usualmente como eventos que codifican los movimientos o pulsación de botones del ratón, pulsaciones de teclas, etc. Los eventos son traducidos a solicitudes de servicio ("service requests" en el texto original) para el modelo o la vista. El usuario interactúa con el sistema a través de los controladores.

Las Vistas y los Controladores conforman la interfaz de usuario. Un mecanismo de propagación de cambios asegura la consistencia entre la interfaz y el modelo. La separación del modelo de los componentes vista y del controlador permite tener múltiples vistas del mismo modelo. Si el usuario cambia el modelo a través del controlador de una vista, todas las otras vistas dependientes deben reflejar los cambios. Por lo tanto, el modelo notifica a todas las vistas siempre que sus datos cambien. Las vistas, en cambio, recuperan los nuevos datos del modelo y actualizan la información que muestran al usuario. La Figura 1 muestra la estructura del patrón MVC.

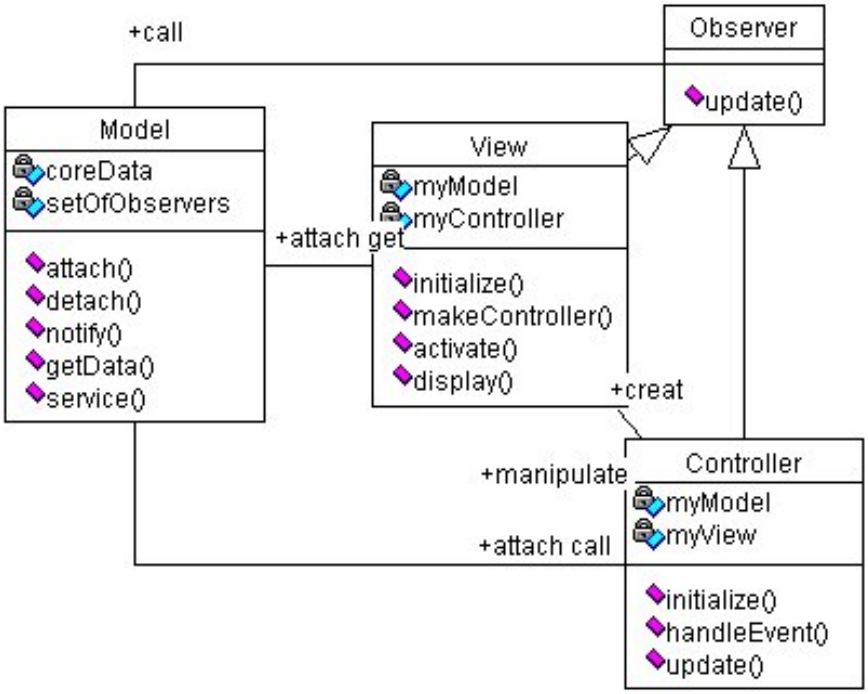

Figura 1. Diagrama de Clases del Patrón Arquitectónico MVC.
Fuente: Pattern Oriented Software Architecture, Volume 1.
Autor: Buschmann, Frank.

Patrón de Diseño

Según Welicki (2005), un patrón de diseño describe una estructura recurrente de componentes que se comunican para resolver un problema general de diseño en un contexto particular.

Lenguaje de Patrones

Zambrano y Acosta (2001) señalan que un lenguaje de patrones es una colección estructurada de patrones de diseño que guía al diseñador desde los problemas de gran escala a los problemas de pequeña escala. Los patrones se identifican a diferentes niveles. Un patrón a un nivel puede implicar un número de patrones a un nivel más bajo para completarlo (ejemplo, un patrón de un cuarto puede implicar patrones para ventanas, puertas, etc.), esos patrones enlazados proveen una gramática informal para el diseño: un lenguaje de patrones.

Plataforma J2EE

Palo (2003) describe a la plataforma J2EE como una arquitectura para el desarrollo y despliegue de aplicaciones basadas en objetos distribuidos transacionales o software de componentes del lado del servidor. Estos componentes del lado del servidor, se llama Enterprise Java Beans (EJB), y son objetos distribuidos que están localizados en contenedores de EJB y proporcionan servicios remotos para clientes distribuidos a lo largo de la red.

El contenedor EJB controla cada aspecto del componente EJB en tiempo de ejecución incluyendo accesos remotos al bean, seguridad, persistencia, transacciones, concurrencia, y accesos a un almacén de recursos.

El contenedor aísla al EJB de accesos directos por parte de aplicaciones cliente. Cuando una aplicación cliente invoca un método remoto de un EJB, el contenedor primero intercepta la llamada para asegurar que la persistencia, las transacciones, y la seguridad son aplicadas apropiadamente a cada operación que el cliente realiza en el EJB. El contenedor maneja estos aspectos de forma automática, por eso el

desarrollador no tiene que escribir este tipo de lógica dentro del propio código del EJB. El desarrollador de EJB puede enfocarse en encapsular la reglas del negocio, mientras el contenedor se ocupa de todo lo demás.

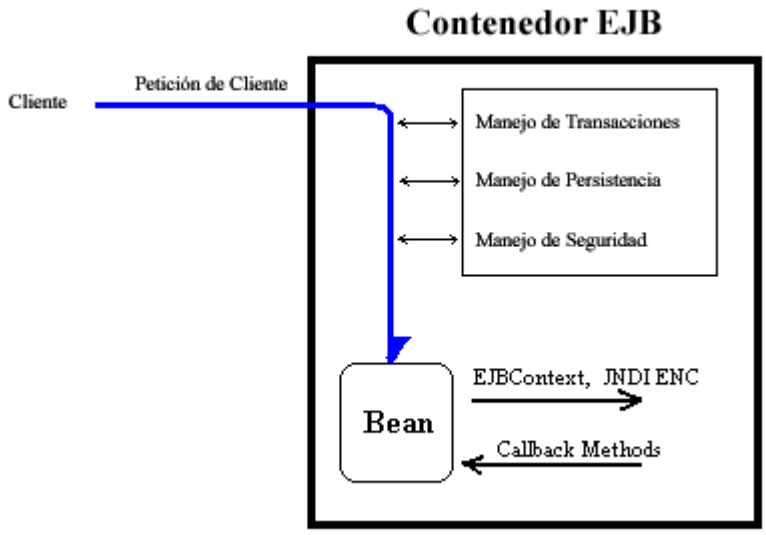

Figura 2. Contenedor de Enterprise Java Beans.
Fuente: http://www.programacion.net.

Los contenedores manejan muchos Beans simultáneamente y para reducir el consumo de memoria y de proceso, los contenedores almacenan los recursos y manejan los ciclos de vida de todos los beans de forma muy cuidadosa. Cuando un Bean no está siendo utilizado, un contenedor lo situará en un almacén para ser reutilizado por otros clientes, o posiblemente lo sacará de la memoria y sólo lo traerá de vuelta cuando sea necesario. Como las aplicaciones cliente no tienen acceso directo a los beans --el contenedor trata con el cliente y el bean-- la aplicación cliente se despreocupa completamente de las actividades de control de recursos del contenedor. Por ejemplo, un bean que no está en uso, podría ser eliminado de la memoria del servidor, mientras que su referencia remota en el cliente permanece intacta. Cuando un cliente invoca a un método de la referencia remota, el contenedor

simplemente re-activa el bean para servir la petición. La aplicación cliente se despreocupa de todo el proceso.

El Framework Struts

Cavaness (2002), presenta al proyecto Struts como un framework lanzado en Mayo del 2000 por Craig R. McClanahan, para proporcionar un marco de trabajo MVC estándar a la comunidad Java.

Las aplicaciones Struts tiene tres componentes principales: un servlet controlador, que está proporcionado por el propio Struts, páginas JSP (Vista), y la lógica de negocio de la aplicación (Modelo).

El servlet controlador Struts une y enruta solicitudes HTTP a otros objetos del marco de trabajo, incluyendo JavaServer Pages y subclases org.apache.struts.action.Action proporcionadas por el desarrollador.

Metodologías de desarrollos Ágiles

En Canós, Letelier y Penadé (2002) se explica de donde nace el término ágil aplicado a la ingeniería de software, ellos relatan que esto ocurre en febrero de 2001, tras una reunión celebrada en Utah-EEUU. En esta reunión participan un grupo de 17 expertos de la industria del software, incluyendo algunos de los creadores o impulsores de metodologías de software. Su objetivo fue bosquejar los valores y principios que deberían permitir a los ingenieros desarrollar software rápidamente y respondiendo a los cambios que puedan surgir a lo largo del proyecto. Se pretendía ofrecer una alternativa a los procesos de desarrollo de software tradicionales,

caracterizados por ser rígidos y dirigidos por la documentación que se genera en cada una de las actividades desarrolladas.

Tras esta reunión se creó The Agile Alliance3, una organización, sin ánimo de lucro, dedicada a promover los conceptos relacionados con el desarrollo ágil de software y ayudar a las organizaciones para que adopten dichos conceptos. El punto de partida fue el Manifiesto Ágil, un documento que resume la filosofía ágil.

El Manifiesto Ágil

En Canós, Letelier y Penadé. (2002) se describe el manifiesto ágil, este muestra como se valora:

- Al individuo y las interacciones del equipo de desarrollo sobre el proceso y las herramientas. La gente es el principal factor de éxito de un proyecto software. Es más importante construir un buen equipo que construir el entorno. Muchas veces se comete el error de construir primero el entorno y esperar que el equipo se adapte automáticamente. Es mejor crear el equipo y que éste configure su propio entorno de desarrollo en base a sus necesidades.
- Desarrollar software que funciona más que conseguir una buena documentación. La regla a seguir es no producir documentos a menos que sean necesarios de forma inmediata para tomar una decisión importante. Estos documentos deben ser cortos y centrarse en lo fundamental.
- La colaboración con el cliente más que la negociación de un contrato. Se propone que exista una interacción constante entre el cliente y el equipo de desarrollo. Esta colaboración entre ambos será la que marque la marcha del proyecto y asegure su éxito.
- Responder a los cambios más que seguir estrictamente un plan. La habilidad de responder a los cambios que puedan surgir a los largo del proyecto

(cambios en los requisitos, en la tecnología, en el equipo, etc.) determina también el éxito o fracaso del mismo. Por lo tanto, la planificación no debe ser estricta sino flexible y abierta.

Los valores anteriores inspiran los doce principios del manifiesto. Son características que diferencian un proceso ágil de uno tradicional. Los dos primeros principios son generales y resumen gran parte del espíritu ágil. El resto tienen que ver con el proceso seguir y con el equipo de desarrollo, en cuanto metas a seguir y organización del mismo. Los principios son:

- La prioridad es satisfacer al cliente mediante tempranas y continuas entregas de software que le aporte un valor.
- Dar la bienvenida a los cambios. Se capturan los cambios para que el cliente tenga una ventaja competitiva.
- Entregar frecuentemente software que funcione desde un par de semanas a un par de meses, con el menor intervalo de tiempo posible entre entregas.
- La gente del negocio y los desarrolladores deben trabajar juntos a lo largo del proyecto.
- Construir el proyecto en torno a individuos motivados. Darles el entorno y el apoyo que necesitan y confiar en ellos para conseguir finalizar el trabajo.
- El diálogo cara a cara es el método más eficiente y efectivo para comunicar información dentro de un equipo de desarrollo.
- El software que funciona es la medida principal de progreso.
- Los procesos ágiles promueven un desarrollo sostenible. Los promotores, desarrolladores y usuarios deberían ser capaces de mantener una paz constante.
- La atención continua a la calidad técnica y al buen diseño mejora la agilidad.
- La simplicidad es esencial.
- Las mejores arquitecturas, requisitos y diseños surgen de los equipos organizados por sí mismos.

- En intervalos regulares, el equipo reflexiona respecto a cómo llegar a ser más efectivo, y según esto ajusta su comportamiento.

Programación Extrema o eXtreme Programing (XP)

eXtreme Programing (XP) nace como nueva disciplina de desarrollo de software hace aproximadamente unos seis años, y ha causado un gran revuelo entre el colectivo de programadores del mundo. Beck (2000), su autor, es un programador que ha trabajado en múltiples empresas y que actualmente lo hace como programador en la conocida empresa automovilística DaimlerChrysler. Con sus teorías ha conseguido el respaldo de gran parte de la industria del software y el rechazo de otra parte.

Los objetivos de XP son muy simples: la satisfacción del cliente. Esta metodología trata de dar al cliente el software que él necesita y cuando lo necesita.

El segundo objetivo es potenciar al máximo el trabajo en grupo. Tanto los jefes de proyecto, los clientes y desarrolladores, son parte del equipo y están involucrados en el desarrollo del software.

XP está diseñada para el desarrollo de aplicaciones que requieran un grupo de programadores pequeño, dónde la comunicación sea más factible que en grupos de desarrollo grandes. La comunicación es un punto importante y debe realizarse entre los programadores, los jefes de proyecto y los clientes.

A continuación se presentara las características esenciales de XP, Beck (2000), organizadas en los tres apartados siguientes: historias de usuario, roles, proceso y prácticas.

Las Historias de Usuario

Las historias de usuario son la técnica utilizada en XP para especificar los requisitos del software. Se trata de tarjetas de papel en las 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, en cualquier momento las historias de usuario pueden romperse, reemplazarse por otras más específicas o generales, añadirse nuevas o ser modificadas. Cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas

Roles de la Programación Extrema

Beck (2000) propone los siguientes roles para el desarrollo cualquier proyecto:

- Programador: El programador escribe las pruebas unitarias y produce el código del sistema. Debe existir una comunicación y coordinación adecuada entre los programadores y otros miembros del equipo.
- Cliente: El cliente escribe las historias de usuario y las pruebas funcionales para validar su implementación. Además, asigna la prioridad a las historias de usuario y decide cuáles se implementan en cada iteración centrándose en aportar mayor valor al negocio. El cliente es sólo uno dentro del proyecto pero puede corresponder a un interlocutor que está representando a varias personas que se verán afectadas por el sistema.

- Encargado de Pruebas: El encargado de pruebas ayuda al cliente a escribir las pruebas funcionales. Ejecuta las pruebas regularmente, difunde los resultados en el equipo y es responsable de las herramientas de soporte para pruebas.
- Encargado de Seguimiento: El encargado de seguimiento proporciona realimentación al equipo en el proceso XP. Su responsabilidad es verificar el grado de acierto entre las estimaciones realizadas y el tiempo real dedicado, comunicando los resultados para mejorar futuras estimaciones. También realiza el seguimiento del progreso de cada iteración y evalúa si los objetivos son alcanzables con las restricciones de tiempo y recursos presentes. Determina cuándo es necesario realizar algún cambio para lograr los objetivos de cada iteración.
- Entrenador: Es responsable del proceso global. Es necesario que conozca a fondo el proceso XP para proveer guías a los miembros del equipo de forma que se apliquen las prácticas XP y se siga el proceso correctamente.
- Consultor: Es un miembro externo del equipo con un conocimiento específico en algún tema necesario para el proyecto. Guía al equipo para resolver un problema específico.
- Gestor: Es el vínculo entre clientes y programadores, ayuda a que el equipo trabaje efectivamente creando las condiciones adecuadas. Su labor esencial es de coordinación.

El Proceso XP

Un proyecto XP tiene éxito cuando el cliente selecciona el valor de negocio a implementar basado en la habilidad del equipo para medir la funcionalidad que puede entregar a través del tiempo. El ciclo de desarrollo consiste (a grandes rasgos) en los siguientes pasos:

1. El cliente define el valor de negocio a implementar.
2. El programador estima el esfuerzo necesario para su implementación.

3. El cliente selecciona qué construir, de acuerdo con sus prioridades y las restricciones de tiempo.
4. El programador construye ese valor de negocio.
5. Vuelve al paso 1.

En todas las iteraciones de este ciclo tanto el cliente como el programador aprenden. No se debe presionar al programador a realizar más trabajo que el estimado, ya que se perderá calidad en el software o no se cumplirán los plazos. De la misma forma el cliente tiene la obligación de manejar el ámbito de entrega del producto, para asegurarse que el sistema tenga el mayor valor de negocio posible con cada iteración.

Fases

El ciclo de vida ideal de XP, Beck (2000), consiste de seis fases: Exploración, Planificación de la Entrega de una Versión, Iteraciones, Producción, Mantenimiento y Muerte del Proyecto.

Fase I: 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. La fase de exploración toma de pocas semanas a

pocos meses, dependiendo del tamaño y familiaridad que tengan los programadores con la tecnología.

Fase II: Planificación de la Entrega

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. Una entrega debería obtenerse en no más de tres meses. Esta fase dura unos pocos días.

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.

La planificación se puede realizar basándose en el tiempo o el alcance. La velocidad del proyecto es utilizada para establecer cuántas historias se pueden implementar antes de una fecha determinada o cuánto tiempo tomará implementar un conjunto de historias. Al planificar por tiempo, se multiplica el número de iteraciones por la velocidad del proyecto, determinándose cuántos puntos se pueden completar. Al planificar según alcance del sistema, se divide la suma de puntos de las historias de usuario seleccionadas entre la velocidad del proyecto, obteniendo el número de iteraciones necesarias para su implementación.

Fase III: 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. Esto se logra escogiendo las historias que fuercen la creación de esta arquitectura, sin embargo, esto no siempre es posible ya que es el cliente quien decide qué historias se implementarán en cada iteración (para maximizar el valor de negocio). Al final de la última iteración el sistema estará listo para entrar en producción.

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. Todo el trabajo de la iteración es expresado en tareas de programación, cada una de ellas es asignada a un programador como responsable, pero llevadas a cabo por parejas de programadores.

Fase IV: Producción

La fase de producción requiere de pruebas adicionales y revisiones de rendimiento antes de que el sistema sea trasladado al entorno del cliente. Al mismo tiempo, se deben tomar decisiones sobre la inclusión de nuevas características a la versión actual, debido a cambios durante esta fase.

Es posible que se rebaje el tiempo que toma cada iteración, de tres a una semana. Las ideas que han sido propuestas y las sugerencias son documentadas para su posterior implementación.

Fase V: Mantenimiento

Mientras la primera versión se encuentra en producción, el proyecto XP debe mantener el sistema en funcionamiento al mismo tiempo que desarrolla nuevas iteraciones. Para realizar esto se requiere de tareas de soporte para el cliente. De esta forma, la velocidad de desarrollo puede bajar después de la puesta del sistema en producción. La fase de mantenimiento puede requerir nuevo personal dentro del equipo y cambios en su estructura.

Fase VI: Muerte del Proyecto

Es cuando el cliente no tiene más historias para ser incluidas en el sistema. Esto requiere que se satisfagan las necesidades del cliente en otros aspectos como rendimiento y confiabilidad del sistema. Se genera la documentación final del sistema y no se realizan más cambios en la arquitectura. La muerte del proyecto también ocurre cuando el sistema no genera los beneficios esperados por el cliente o cuando no hay presupuesto para mantenerlo.

Prácticas XP

La principal suposición que se realiza en XP es la posibilidad de disminuir la mítica curva exponencial del costo del cambio a lo largo del proyecto, lo suficiente para que el diseño evolutivo funcione. Esto se consigue gracias a las tecnologías disponibles para ayudar en el desarrollo de software y a la aplicación disciplinada de las siguientes prácticas.

1. El juego de la planificación. Hay una comunicación frecuente el cliente y los programadores. El equipo técnico realiza una estimación del esfuerzo requerido para la implementación de las historias de usuario y los clientes deciden sobre el ámbito y tiempo de las entregas y de cada iteración.
2. Entregas pequeñas. Producir rápidamente versiones del sistema que sean operativas, aunque no cuenten con toda la funcionalidad del sistema. Esta versión ya constituye un resultado de valor para el negocio. Una entrega no debería tardar más 3 meses.
3. Metáfora. El sistema es definido mediante una metáfora o un conjunto de metáforas compartidas por el cliente y el equipo de desarrollo. Una metáfora es una historia compartida que describe cómo debería funcionar el sistema (conjunto de nombres que actúen como vocabulario para hablar sobre el dominio del problema, ayudando a la nomenclatura de clases y métodos del sistema).
4. Diseño simple. Se debe diseñar la solución más simple que pueda funcionar y ser implementada en un momento determinado del proyecto.
5. Pruebas. La producción de código está dirigida por las pruebas unitarias. Éstas son establecidas por el cliente antes de escribirse el código y son ejecutadas constantemente ante cada modificación del sistema.
6. Refactorización (Refactoring). Es una actividad constante de reestructuración del código con el objetivo de remover duplicación de código, mejorar su legibilidad, simplificarlo y hacerlo más flexible para facilitar los posteriores

cambios. Se mejora la estructura interna del código sin alterar su comportamiento externo.

7. Programación en parejas. Toda la producción de código debe realizarse con trabajo en parejas de programadores. Esto conlleva ventajas implícitas (menor tasa de errores, mejor diseño, mayor satisfacción de los programadores).
8. Propiedad colectiva del código. Cualquier programador puede cambiar cualquier parte del código en cualquier momento.
9. Integración continua. Cada pieza de código es integrada en el sistema una vez que esté lista. Así, el sistema puede llegar a ser integrado y construido varias veces en un mismo día.
10. 40 horas por semana. Se debe trabajar un máximo de 40 horas por semana. No se trabajan horas extras en dos semanas seguidas. Si esto ocurre, probablemente está ocurriendo un problema que debe corregirse. El trabajo extra desmotiva al equipo.
11. Cliente in-situ. El cliente tiene que estar presente y disponible todo el tiempo para el equipo. Éste es uno de los principales factores de éxito del proyecto XP. El cliente conduce constantemente el trabajo hacia lo que aportará mayor valor de negocio y los programadores pueden resolver de manera inmediata cualquier duda asociada. La comunicación oral es más efectiva que la escrita.
12. Estándares de programación. XP enfatiza que la comunicación de los programadores es a través del código, con lo cual es indispensable que se sigan ciertos estándares de programación para mantener el código legible.

El mayor beneficio de las prácticas se consigue con su aplicación conjunta y equilibrada puesto que se apoyan unas en otras. Esto se ilustra en la Figura 1, donde una línea entre dos prácticas significa que las dos prácticas se refuerzan entre sí. La mayoría de las prácticas propuestas por XP no son novedosas sino que en alguna forma ya habían sido propuestas en ingeniería del software e incluso demostrado su valor en la práctica. El mérito de XP es integrarlas de una forma efectiva y

complementarlas con otras ideas desde la perspectiva del negocio, los valores humanos y el trabajo en equipo.

Lenguaje de Modelado Unificado (UML)

UML es un lenguaje gráfico para visualizar, especificar, construir y documentar los componentes de un sistema software, Booch, Rumbaugh y Jacobson (1999). UML permite tanto la especificación conceptual de un sistema como la especificación de elementos concretos, como pueden ser las clases o un diseño de base de datos.

Según su definición, los objetivos de UML son los siguientes:

- Visualizar: UML permite representar mediante su simbología el contenido y la estructura de un sistema software. La notación UML permite definir modelos que serán claramente comprensibles por otros desarrolladores facilitando así el mantenimiento del sistema que describe.
- Especificar: UML permite especificar los procesos de análisis, diseño y codificación de un sistema software. También permite determinar modelos precisos, sin ambigüedades, detallando las partes esenciales de los mismos.
- Construir: Las anteriores características permiten que UML pueda generar código en distintos lenguajes de programación y tablas en una base de datos a partir de modelos UML. Además permite simular el comportamiento de sistemas software.
- Documentar: Como ya se comentó antes, UML permite especificar los procesos de análisis, diseño y codificación y también permite documentar los mismos, dejando clara la arquitectura del sistema.

El Lenguaje Unificado de Modelado preescribe un conjunto de notaciones y diagramas estándar para modelar sistemas orientados a objetos, y describe la

semántica esencial de lo que estos diagramas y símbolos significan. Mientras que ha habido muchas notaciones y métodos usados para el diseño orientado a objetos, ahora los modeladores sólo tienen que aprender una única notación.

UML se puede usar para modelar distintos tipos de sistemas: sistemas de software, sistemas de hardware, y organizaciones del mundo real. UML ofrece nueve diagramas en los cuales modelar sistemas.

- Diagramas de Casos de Uso para modelar los procesos.
- Diagramas de Secuencia para modelar el paso de mensajes entre objetos.
- Diagramas de Colaboración para modelar interacciones entre objetos.
- Diagramas de Estado para modelar el comportamiento de los objetos en el sistema.
- Diagramas de Actividad para modelar el comportamiento de los Casos de Uso, objetos u operaciones.
- Diagramas de Clases para modelar la estructura estática de las clases en el sistema.
- Diagramas de Objetos para modelar la estructura estática de los objetos en el sistema.
- Diagramas de Componentes para modelar componentes.
- Diagramas de Implementación para modelar la distribución del sistema.

UML es una consolidación de muchas de las notaciones y conceptos más usados orientados a objetos. Empezó como una consolidación del trabajo de Booch, Rumbaugh y Jacobson (1999), creadores de tres de las metodologías orientadas a objetos más populares.

Para finalizar este capitulo se describe a continuación las variables del estudio y la operacionalización de las mismas.

Operacionalización de las Variables

Variables Conceptuales

- Documento y Contenido: Es una agrupación de elementos texto, imagen, audio y video que resume el conocimiento y experticia que una persona desea comunicar.
- Gestión de Documentos y Contenido: Es el proceso de crear, publicar y distribuir los documentos y contenido generados por una persona.

Variables Operacionales

Para efectos de la operacionalización de las variables conceptuales se procedió de la siguiente manera:

La variable -Documento- se operacionalizó considerando dos dimensiones, la dimensión Estructura del Documento y la dimensión Proceso enseñanza-aprendizaje.

La dimensión estructura está constituida por los elementos que conforman un documento. La dimensión proceso enseñanza-aprendizaje está constituida por los factores que influyen en la distribución de documentos dentro del proceso enseñanza aprendizaje, facilidad de acceso, rapidez y disponibilidad.

La variable –Gestión de Documentos- se operacionalizó considerando dos dimensiones, la dimensión Importancia del Servicio y Funcionalidades del Sistema.

La dimensión Importancia del Servicio esta constituida por la apreciación acerca de cómo debe funcionar el servicio y su importancia. La dimensión Funcionalidades del Sistema, recoge la plataforma tecnológica y los requisitos funcionales, como factores determinantes en las características de un sistema de gestión documentos y contenido.

A continuación se presenta la tabla N° 1 que resume la operacionalización de las variables en estudio.

Tabla 1: Operacionalización de las variables estudiadas.

Propósito

Determinar las funcionalidades asociadas a un sistema de gestión de gestión de contenidos y documentos basado en tecnología Web para proceso de enseñanza-aprendizaje.

Definición

Un sistema de gestión de documentos y contenidos es un software que permite automatizar el proceso de creación, distribución y búsqueda de documentos entre una comunidad de usuarios.

Variable	Dimensiones	Indicadores	Item Cuestionario N° 1	Item Cuestionario N° 2
Documento	Estructura	Elementos de un Documento	1,2	
	Proceso Enseñanza-Aprendizaje	Facilidad de Acceso	3,4	
		Rapidez	3	
		Disponibilidad	3,5	
Gestión de Documentos	Importancia del Servicio	Apreciación del Servicio	6	
		Beneficios del Servicio	8	
	Funcionalidades del Sistema	Plataforma Tecnológica.	9	1,2,3,4
		Requisitos Funcionales.	7	

Fuente: El autor de la investigación.

CAPITULO III

MARCO METODOLOGICO

En toda investigación científica, se hace necesario, que los hechos estudiados, así como las relaciones que se establecen entre estos, los resultados obtenidos y las evidencias significativas encontradas en relación con el problema investigado, además de los nuevos conocimientos aportados, reúnan las condiciones de fiabilidad, objetividad y validez interna; para lo cual, se requiere delimitar los procedimientos de orden metodológico, a través de los cuales se intenta dar respuestas a las interrogantes objeto de investigación.

En consecuencia, el marco metodológico de la presente investigación que desarrolla un sistema que permita la gestión de contenidos relacionados al proceso enseñanza aprendizaje en el Decanato de Ciencias y Tecnología de la UCLA; es la instancia que alude al momento tecno-operacional presente en todo proceso de investigación; donde se sitúa en detalle, el conjunto de métodos, técnicas y protocolos instrumentales que se emplearon en el proceso de recolección de los datos requeridos en la investigación propuesta.

En este sentido, este capítulo desarrolla aspectos metodológicos relativos al tipo de estudio y su diseño de investigación, incorporados en relación a los objetivos establecidos, las técnicas e instrumentos empleados en la recolección de los datos incluyendo sus características; las formas de codificación, presentación de los datos; y el análisis e interpretación de los resultados que permitieron las conclusiones y

desarrollo de un sistema de gestión de contenidos para el Decanato de Ciencias y Tecnología de la UCLA.

Naturaleza del Estudio

El estudio se apoyó en una investigación bajo la modalidad proyectos tal como lo especifica UCLA (2002), ya que al crear una herramienta de software para el manejo de documentos y contenido dentro del proceso enseñanza-aprendizaje en el Decanato de Ciencias y Tecnología se descubrieron valores reales que representaron el diseño de un modelo factible para la solución de un problema de tipo práctico.

Además, este trabajo es una investigación de campo de tipo descriptiva. Para el caso de estudio, la información está registrada en la propia institución, estos registros son producto de esfuerzos sistemáticos de organización de los datos e información sobre el tema objeto de estudio y como tal se considera de alta confiabilidad. Es de campo, porque se siguen las estrategias basadas en métodos que permiten conocer los datos en forma directa de la realidad. Constituye un proceso sistemático riguroso y racional de recolección, tratamiento, análisis y presentación de los datos, basados en una estrategia de recolección. Igualmente, es descriptiva, ya que comprende además de la descripción, el registro, análisis e interpretación de la naturaleza actual del problema del fenómeno estudiado, Ballestrini (2002).

En síntesis, esta investigación analizo una situación particular como lo es la actividad de generación, distribución y búsqueda de documentos relacionados al proceso enseñanza-aprendizaje en el Decanato de Ciencias y Tecnología; recogiendo su importancia, y desarrollando un sistema de gestión de contenidos que permite mejorarlo y optimizarlo.

La metodología aplicada permitió evaluar varios aspectos relacionados al proceso de generación, distribución y búsqueda de documentos, entre estos:

- Desarrollo actual del proceso, identificando las actividades criticas y personas involucradas.
- Identificar los problemas asociados a las actividades del proceso.
- Identificar oportunidades de mejora relacionadas a las actividades del proceso.
- Establecer el impacto de la mejora asociada al proceso.

Para recopilar información en esta investigación se utilizaron dos métodos. Según Lee (1992), varios investigadores dedicados a hacer estudios tanto cuantitativos como cualitativos, sugieren combinar mas de un método en una misma investigación.

Para está investigación, se aplicaron los siguientes métodos de recolección de datos:

- Revisión Bibliográfica, que permite conocer conceptos, características y funcionamiento de los actuales sistemas de gestión de documentos y contenido, definir su papel en una estrategia integral de gestión de conocimiento, y descubrir además, las nuevas tendencias de gestión de contenido.
- Entrevista Semi-estructurada, con el método de recolección de datos, se realizaron entrevistas con preguntas abiertas, a personal docente que interviene directamente en el proceso enseñanza-aprendizaje y a personal administrativo del centro de computación, responsable de la infraestructura de computación en el Decanato de Ciencias y Tecnología. Estas permitieron conocer por un lado las necesidades relativas a un sistema de distribución de documentos de dominio público y por otro analizar las capacidades técnicas con que cuenta el decanato para la implementación de un software de gestión de contenidos.

Fases del Estudio

En atención a esta modalidad de investigación, se ejecutaron tres fases en el estudio, a fin de cumplir con los requisitos involucrados en un Proyecto Factible. En la primera de ellas, se desarrollo el conocimiento de la situación existente en la realidad objeto de estudio, a fin de describir la plataforma tecnológica y las funcionalidades requeridas por el sistema de gestión de documentos y contenidos. En la segunda fase de la investigación y atendiendo a los resultados de la fase diagnóstica, se formuló un modelo arquitectónico, para dar respuestas a los problemas planteados inicialmente en la investigación. En la tercera fase se desarrolló un software, alineado al modelo operativo propuesto. En el desarrollo de la solución de software se usaran los principios y técnicas recomendados por la metodología ágil de desarrollo "eXtreme Programming" (XP).

Fase Diagnostica

En esta fase se desarrollaron dos entrevistas semi-estructuradas; una a profesores activos del Departamento de Sistemas, para determinar sus necesidades con respecto a la distribución y publicación de material instruccional, y otra a personal del centro de computación para medir los recursos técnicos y administrativos con que cuenta el Decanato de Ciencias y Tecnología para la implementación de un sistema de gestión de documentos.

Procedimiento

En el siguiente aparte se listan los pasos de la presente investigación:

1. Elaborar los instrumentos.
2. Aplicar los instrumentos.
3. Analizar los datos recabados por los instrumentos.
4. Presentar las conclusiones del diagnóstico.
5. Presentar las recomendaciones del diagnóstico.
6. Efectuar el análisis de factibilidad.
7. Elaborar la propuesta.

Técnicas e Instrumentos de Recolección de Datos

El plan de recolección de datos se llevó a cabo en las siguientes etapas:

Primera Etapa: Consistió en el **diseño de los instrumentos de recolección de datos.**

Se diseñaron dos instrumentos**:**

El primer instrumento una entrevista semi-estructurada, para profesores del Departamento de Sistemas, con la cual se buscaba conocer su opinión respecto a la necesidad de un sistema de gestión de documentos orientado al proceso de enseñanza-aprendizaje.

El segundo instrumento es una entrevista semi-estructurada, para personal administrativo del centro de computación, para conocer los recursos técnicos y humanos, disponibles en el Decanato de Ciencias y Tecnología, para la implantación y puesta en marcha de un sistema de gestión de contenidos.

Segunda Parte: Aplicación de los instrumentos. En la aplicación de los instrumentos se realizaron las siguientes actividades: 1) Se contactaron a profesores y personal administrativo. 2) Se efectuaron las entrevistas semi-estructuradas.

Resultados

A través de la entrevista semi-estructurada aplicada a profesores, se identificaron ciertos problemas originados por la ineficiente distribución de documentos, que afectan al proceso enseñanza-aprendizaje y en consecuencia al cuerpo docente y a la población estudiantil. Entre estos se citan:

- Desinformación con respecto al horario de permanencia de los profesores del decanato, que influye en los procesos de consulta y asesoría que requieren los alumnos para el cumplimiento de los objetivos en las distintas asignaturas.
- Ausencia de información, con respecto a convocatoria de reuniones de coordinación y departamento, que retrasa la generación de estrategias tendientes a atacar problemas presentes en el proceso enseñanza-aprendizaje.
- Desconocimiento sobre los trabajos y líneas de investigación que desarrolla el cuerpo docente del Decanato de Ciencias y Tecnología.
- Retraso en el acceso al material instruccional y de apoyo, construido por los distintos docentes y que son requeridos por los alumnos del decanato.
- Falta de información en cuanto a cambios de la planificación, avisos importantes y resultados de evaluación, en la población estudiantil.

Se pudo conocer que los docentes del Decanato utilizan diversos mecanismos para distribuir su material instruccional, algunos emplean el correo electrónico y sitios web fuera de las instalaciones de la Universidad, otros convierten el material en guías impresas y lo proporcionan para que sus alumnos la reproduzcan. En todo caso es imposible, bajo estas circunstancias, ubicar de manera rápida el material producido por el personal docente del Decanato de Ciencias y Tecnología.

Las circunstancias antes descritas determinaron la necesidad de construir un sistema, que permita agilizar el proceso de distribución de los documentos relacionados al proceso enseñanza-aprendizaje.

En base a la consulta bibliográfica realizada en Internet, en la que fue posible determinar las funcionalidades de distintos sistemas de gestión de contenidos, se caracterizo el sistema requerido en los siguientes términos:

- El manejo y la construcción de documentos debe ser completamente digital y colocarse en una plataforma que permita su acceso y distribución ininterrumpidamente.
- La plataforma de implementación debe ser económica y adaptadas a las tendencias tecnológicas.
- El formato en que será construido un documento es indiferente para el sistema.
- Un documento podrá estar relacionado con otros presentes en el sistema.
- Cada documento estará organizado por áreas de interés o departamento.
- La forma en que se manejan los documentos en un área de interés será independiente de otra.
- Los documentos deben tener datos descriptivos que se puedan usar para procesos de búsqueda.
- Debe ser portable, para adaptarse a distintas plataformas de sistema operativo y hardware disponibles en la Universidad.

- Los dueños de los documentos deben tener la libertad de modificarlos y distribuirlos en el momento en que lo consideren conveniente.
- Debe contar con mecanismos que permitan controlar el acceso a las distintas áreas de conocimiento y asignar un responsable en cada una de ellas, que se encarguen de supervisar y controlar los documentos que se generan dentro de el.

Las entrevistas realizadas al personal administrativo del centro de computación, permitieron comprobar la disponibilidad de personal capacitado y equipos, en el Centro de Computación del Decanato de Ciencias y Tecnología, que pueden contribuir a la implantación y puesta en marcha de un sistema de gestión de documentos.

Igualmente se evidenció los niveles de interconexión que existen entre las distintas áreas físicas que componen el decanato, esto a través de una red institucional de alta velocidad, que facilitaría el proceso de creación y distribución de documentos.

Fase de Factibilidad

Social

La factibilidad social se basó en indagar las necesidades de los usuarios a fin de describir las funcionalidades asociadas con un servicio de gestión de contenidos y documentos, basado en tecnología Web.

En función de lo anterior, los miembros del Decanato de Ciencias y Tecnología valoran y aprecian el desarrollo de un sistema de gestión de contenidos que permita la

creación, distribución y búsqueda de documentos de intención académica, razón por la cual esta es una alternativa que permitirá satisfacer las exigencias de un cuerpo social en la búsqueda métodos alternativos de comunicación entre profesores y alumnos.

Operativa

En relación a la factibilidad operativa desde la perspectiva del cliente, se tiene que esta queda garantizada en virtud de que el manejo de aplicaciones de Internet es conocido por los miembros del Decanato de Ciencias y Tecnología.

Por otra parte, desde la perspectiva del servidor, un servicio de gestión de documentos y contenidos necesita de un administrador. Este profesional debe contar con los conocimientos y habilidades que le permitan manejar los aspectos inherentes a una plataforma de distribución y manejo de contenido, incluyéndose tareas de: implantación, administración, seguridad, y otros. La operatividad en este sentido está garantizada por la estructura organizativa asociada al Centro de Computación del Decanato de Ciencias y Tecnología.

Institucional

En relación al aspecto institucional el desarrollo de una herramienta de gestión de contenidos y documento, fortalecerá la comunicación entre alumnos y profesores, esto garantizaría a la institución mejorar la efectividad del proceso de enseñanza-aprendizaje, que se lleva a cabo en aula. En este sentido es importante destacar que el Decanato de Ciencias y Tecnología cuenta con personal que esta en condiciones de administrar un software como el planteado en el presente trabajo de investigación.

Tecnológica

Desde el punto de vista tecnológico, el desarrollo de una herramienta de software, requiere de recursos de software y hardware.

Dada la iniciativa del gobierno nacional en promover el uso del software libre en los organismos gubernamentales, se dará prioridad a la utilización de herramientas de código abierto para la implementación del sistema de gestión de documentos y contenido. En este sentido es importante acotar, que dado los alcances y limitaciones de esta investigación, existe en el mundo del software libre una amplia gama de componentes que permitieron el desarrollo del sistema y su puesta en marcha. Entre los componentes necesarios están:

- Sistema Operativo.
- Herramienta para el Modelado.
- Lenguaje de programación.
- Ambiente de Desarrollo Visual.
- Ambiente de Diseño para la Interfase y componentes Web.
- Manejador de Base de Datos.
- Servidor Web.
- Servidor de Aplicaciones Web.
- Librerías de Desarrollo Complementarias.

En lo que se refiere al hardware necesario, se realizó un estudio de los equipos necesarios determinándose que el Decanato de Ciencias cuenta con un Servidor, sobre el que se instalará la aplicación Web de gestión de contenido y documentos.

Económica

La factibilidad económica esta garantizada ya que el decanato no necesita hacer inversiones adicionales para la puesta en marcha del software de gestión de contenidos y documentos.

CAPITULO IV

PROPUESTA DEL ESTUDIO

Introducción

Los modelos de desarrollo de software organizan el conjunto de actividades necesarias para obtener productos de software de calidad y consistencia, se centran en definir un proceso de desarrollo racional y controlable, pero de ninguna manera existe un modelo de desarrollo universal y único, existen múltiples variantes, y una u otra son mas o menos efectivas en el contexto donde se empleen. Los modelos de desarrollo representan una guía de cómo y en que orden deben realizarse cada una de estas actividades, y establecen el marco del ciclo de vida del software.

El producto final de la presente investigación, será un software de gestión de documentos y contenidos que podrá ser utilizado en el proceso enseñanza-aprendizaje que se desarrolla en el Decanato de Ciencias y Tecnología de la UCLA, para ello es necesario establecer un modelo de desarrollo que estructure las actividades y especifique de que manera se ejecutaran cada una de ellas. En este sentido se ha planteado utilizar el modelo de desarrollo “eXtreme Programming”.

XP es una metodología ágil para equipos de desarrollo de software pequeños y medianos, diseñada para responder rápidamente a los cambios y lograr entregas de software, continuas, rápidas y de valor. Esta metodología establece un ciclo de vida que se utilizara para dirigir el planteamiento y el desarrollo de la propuesta.

Un proyecto de XP ideal consiste de seis fases: Exploración, Planificación de la Entrega de una Versión, Iteraciones, Producción, Mantenimiento y Muerte del Proyecto. En esta investigación se realizaron las fases de exploración, planificación de entregas y las iteraciones necesarias para lograr el producto de software que se quiere, no se implementaron las fases de producción y mantenimiento.

La fase de exploración en este proyecto, implicó establecer los alcances y requerimientos funcionales y no-funcionales del sistema. En este sentido se elaboró un modelo de requerimientos basado en las características de distintos sistemas de gestión de contenido y documentos presentes en el mercado y en historias de usuario que se recogieron mediante la aplicación de una entrevista. También fue necesario conocer la tecnología usada durante el desarrollo del proyecto, al igual que explorar posibles arquitecturas de software, que soporten eficientemente los alcances del proyecto.

Las historias de usuarios recogidas en la fase de exploración, sirvieron para construir la planificación del proyecto, que consiste en proponer un cronograma de entregas, en donde se especifique las historias de usuarios a ser implementadas en cada versión del software. La metodología sugiere un desarrollo incremental, que se logra a través de iteraciones cortas y sencillas, cada iteración tiene un ciclo de vida que se inicia con una planificación, que consiste en tomar las historias de usuario que no fueron cubiertas en la iteración anterior y las planificadas para la presente, para comenzar su desarrollo. Cada iteración concluye con un test de aceptación, que recoge la satisfacción del cliente con respecto a las historias programadas.

En el caso del presente trabajo de investigación se desarrollaron 4 iteraciones. La primera iteración es la mas importantes porque es allí donde se establece la arquitectura del sistema y la tecnología utilizada para el desarrollo del producto de software, el resto de las iteraciones como se explicará más adelante cubre una lista de historia de usuarios y requerimientos que son establecidos en el plan de entregas.

Exploración

Para esta primera etapa se procedió a recoger las historias de usuario y se realizó una investigación documental sobre productos de gestión de contenidos disponibles en el mercado, con el objeto de describir el sistema, en términos de requerimientos y tareas, también se llevaron a cabo actividades de investigación con respecto a la arquitectura e infraestructura necesaria para el desarrollo del sistema.

Descripción del Sistema

El sistema de gestión de contenidos y documentos, el cual se denomino "OBELISCO", es un software que permitirá automatizar la publicación de información en la Web, a través de un proceso sencillo, concreto y bien documentado.

Debe estar desarrollado con tecnología que le permita su portabilidad a distintas plataformas de sistemas operativos, con el objeto de que no se limite al usuario a usar una sola plataforma.

El acceso al sistema se basa en una cuenta de usuario, sobre la que se establecen privilegios, que limitan las actividades que puede realizar. Este es un mecanismo para garantizar la confidencialidad de los documentos de los usuarios, en el caso en que fuera necesario.

Debe estar en capacidad de separar el contenido de su presentación al usuario, con el objeto de garantizar que el mismo pueda ser desplegado de distintas maneras a múltiples usuarios, sin necesidad de alterar los procesos de negocio modelados dentro del sistema.

Una de las prioridades del sistema es involucrar a la mayor cantidad de personas de una organización, para ello OBELISCO debe proporcionar mecanismos simples y amigables que permita a los distintos usuarios, publicar, administrar y distribuir el contenido generados por ellos.

OBELISCO debe tener capacidad para:

- Controlar quien publica documentos.
- Almacenar y ordenar todos los tipos de documentos, incluyendo gráficos, audio y video en un repositorio central.
- Crear un flujo de trabajo entre las personas que crean documentos y los responsables de su publicación.
- Cambiar la apariencia general del formato del sitio, a través de la definición y reuso de plantillas.
- Agregar nuevas categorías de contenido y metadata a los documentos antes y después de que son almacenados en la base de datos de documentos.

La unidad básica de trabajo en OBELISCO se llama documento. Un documento normalmente se crea para expresar un problema o situación particular, es una constante en el ser humano tratar día a día con problemas o situaciones.

Un Documento posee atributos que lo describen (Leming, 2003):

- Titulo.
- Identificador.
- Autor.
- Directorio.
- Resumen.
- Palabras Claves.
- Fecha de Creación.

- Fuentes o Contribuciones.
- Tipo de documento (Puede representar el propósito del documento, Noticias, Especificaciones, Reportaje).
- Relaciones con otros Documentos.
- Tipo de Audiencia.
- Versión.

Una adecuada taxonomía de los documentos facilita la estructuración de la información, para ello OBELISCO utiliza una estructura llamada directorio. Un directorio tiene atributos que definen el tratamiento que se les dará a los documentos depositados. Entre los atributos que caracterizan a un directorio tenemos:

- Identificador.
- Titulo.
- Tiempo de Vida de los Documentos.
- Publicación automática de documentos.

Los directorios de OBELISCO se dispondrán de manera jerárquica y los documentos dentro de él se discriminan como documentos principales y secundarios.

El ciclo de vida del contenido (Roger y Kirriemuir, 2003) de OBELISCO se desarrolla en tres pasos:

1. Creación de Contenido: En este paso múltiples usuarios autorizados crean documentos, con contenido multimedia y archivos binarios, usando una interfase amigable y sencilla.
2. Aprobación del Contenido: La aprobación del contenido es crucial en el ciclo de vida de los documentos del sistema. Amerita un proceso estricto de aprobación que garantice la validez del contenido. El flujo de trabajo comienza cuando un usuario llamado 'Publicador' envía su trabajo a un directorio determinado para su aprobación. Después de que el trabajo es enviado al repositorio de documentos, un usuario llamado 'Moderador' se

encarga de aprobar o rechazar el documento enviado. El moderador es el usuario que establece quien puede producir información en un directorio y quien puede consumirla o consultarla. Cada directorio del sistema es administrado por un usuario moderador.

3. Distribución del Contenido: El sistema se encarga de desplegar los documentos a los que tiene acceso el usuario actual.

El flujo de trabajo definido para la creación, publicación y distribución del contenido, a través del cual se controla su pertinencia, coherencia y validez, se logra definiendo un moderador por cada directorio creado en el sistema. El moderador se encarga de:

- Aprobar o Rechazar el contenido generado por otros usuarios en su área de competencia.
- Definir el tiempo de vencimiento de los documentos creados.
- Establecer los niveles de acceso de los usuarios del sistema en el repositorio baso su responsabilidad.
- Crear áreas de conocimiento más especializadas dentro de su área de competencia y asignar sus respectivos responsables.

Un usuario podrá crear documentos en un determinado directorio siempre y cuando su acceso es autorizado por su moderador. El Moderador puede discriminar actividades de lectura y escritura dentro del directorio bajo su responsabilidad.

En OBELISCO existirá un administrador cuya responsabilidad es: 1) Actualizar la base de datos de seguridad del sistema. 2) Especificar la estructura de directorios que soportara los documentos. 3) Fijar los documentos principales o de entrada al sistema.

Grupo de usuarios

El sistema está diseñado para que cualquier persona pueda publicar contenido de interés colectivo, sin ser limitativo el conocimiento de la tecnología Web. Su uso en el Decanato de Ciencias y Tecnología debe facilitar a los profesores de distintas áreas de conocimiento, la publicación de material instruccional y su acceso y consulta por el resto de la comunidad universitaria.

Lista de Tareas

A partir de la descripción del sistema podremos extraer una lista de tareas, que permitirán establecer los alcances del sistema, estas son:

- Administrar Documentos de Usuario.
    - Crear Documento.
    - Editar Documento.
    - Eliminar Documento.
- Consultar Contenido.
    - Consultar Documento
    - Consultar Directorio.
- Crear Estructura de Directorio.
- Moderar Directorio
    - Aprobar/Rechazar Documentos.
    - Activar/Desactivar Documentos.
    - Administrar Documentos Principales y Secundarios.
- Administrar Sistema
    - Actualizar Usuarios.
    - Actualizar Parámetros del Sitio.

- Actualizar Preferencias de Usuario
    - Directorios Favoritos.
    - Datos Personales

Diagrama de Casos de Uso

En función a la lista de tareas definidas, se creó un diagrama de casos de uso, Figura 3, que presenta de manera gráfica y usando la notación UML las funcionalidades del sistema.

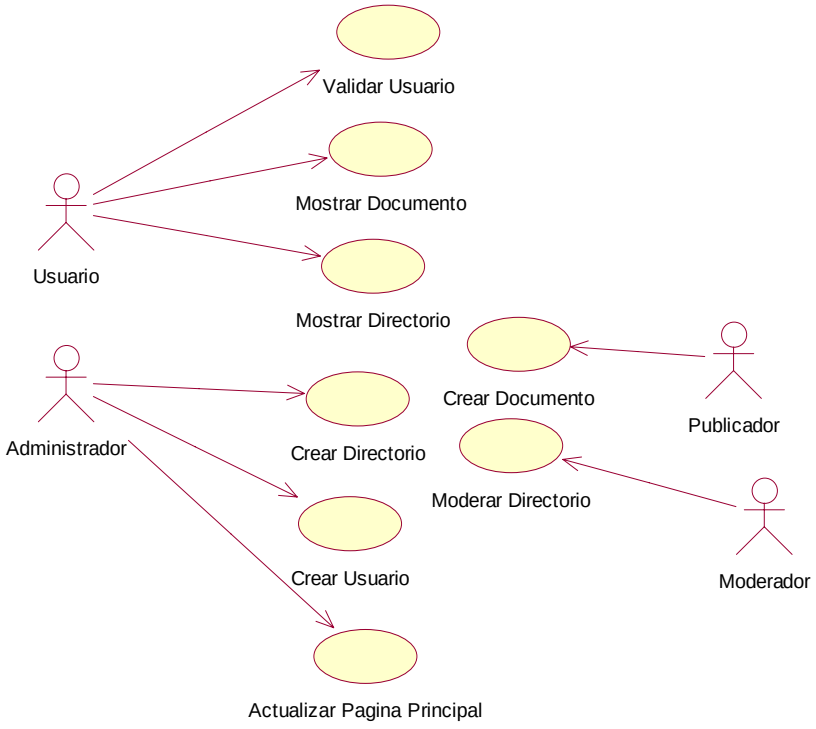

Figura 3. Diagrama de Casos de Uso de OBELISCO.
Autor: El autor de la investigación.

Diccionario de Actores

Dentro del flujo de trabajo que ocurre en OBELISCO, existen 4 actores fundamentales con diferentes roles y funciones, estos son:

- Usuario: Cualquier persona que usa el sistema, con la intención de consultar los documentos almacenados en los distintos directorios.
- Administrador: Es un tipo de usuario especializado en actualizar la base de datos de seguridad del sistema, en especificar la estructura de directorios que soportara los documentos y en determinar los documentos principales o de entrada al sistema.
- Moderador: Usuario al que se le asigna la responsabilidad directa de la administración de un directorio para crear las listas de usuarios de lectura y escritura, para aprobar o rechazar la publicación de los documentos creados por los distintos usuario, para establecer los documentos principales y secundarios, los documentos principales son documentos candidatos a ser documentos de entrada a OBELISCO.
- Publicador: Usuario con la potestad de crear documentos en un directorio en particular.

La Figura 4 presenta la relación que existe entre los distintos actores que intervienen en el flujo de trabajo de OBELISCO. En ella se observa que el Administrador, Publicador y Moderador heredan características del actor Usuario.

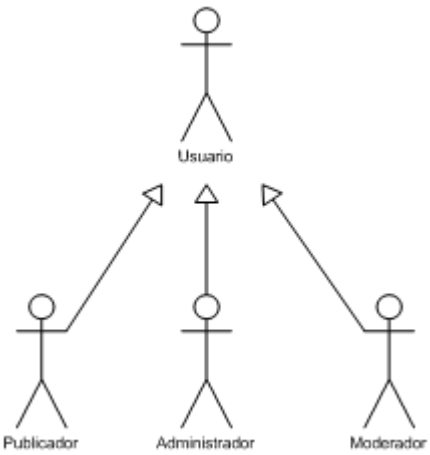

Figura 4. Actores de OBELISCO.
Fuente: El autor de la investigación.

Planificación de entregas

La metodología de desarrollo XP sugiere establecer prioridades sobre las tareas que ejecutará el sistema, con el fin de organizarlas para construirlas en cada iteración del proceso de desarrollo.

Una vez definidas las prioridades sobre las tareas y habiéndolas organizado en iteraciones, el grupo de desarrollo calcula el esfuerzo necesario para completarlas, tomando en cuenta no solo el tiempo para su desarrollo, sino también el invertido en la planificación y la validación del producto de software con los usuarios finales.

La Tabla 2 muestra la planificación de las iteraciones, mostrando el esfuerzo necesario en semanas para completar cada tarea del sistema.

Tabla 2: Planificación de Iteraciones.

Iter.	Tareas a desarrollar	Semanas
0	• Describir la Arquitectura de Software. • Definir Herramientas de Desarrollo. • Establecer los patrones de Diseño necesarios para el desarrollo de los componentes del sistema.	2
1	Desarrollar las tareas: • Consultar Contenido.	3
2	Desarrollar las tareas: • Administrar Documentos de Usuario.	3
3	Desarrollar las tareas: • Moderar Directorio.	3
4	Desarrollar las tareas: • Administrar Sistema. • Actualizar Preferencias del Usuario.	3

Fuente: El autor de la investigación.

Iteración N° 0

Esta iteración se centra en definir la arquitectura que soportara el desarrollo de software. Con ella se puede establecer las herramientas tecnológicas más idóneas para su construcción y de esta manera seleccionar los patrones de diseños mas apropiados para la confección de cada uno de los componentes del sistema.

Arquitectura de Software

Un sistema de manejo de contenidos como OBELISCO, requiere para su funcionamiento, de la participación e iteración de todos los usuarios de la organización, en este sentido hay que acotar que no todos poseen los mismos niveles de conocimiento con respecto a informática y computación, de allí radica la necesidad de contar con interfaces visuales rápidas, precisas y fáciles de usar.

Otro elemento a tomar en cuenta, es la necesidad de contar con mecanismos de distribución, que permitan exportar el contenido en múltiples formatos y tengan flexibilidad para poder modificar su presentación de manera rápida y sencilla.

Por lo antes expuesto se necesita de una arquitectura de software, que ofrezca flexibilidad para separar el contenido de las interfaces de usuario. En este sentido se recurrirá al patrón arquitectural MVC.

La Arquitectura de OBELISCO se muestra en la Figura 5.

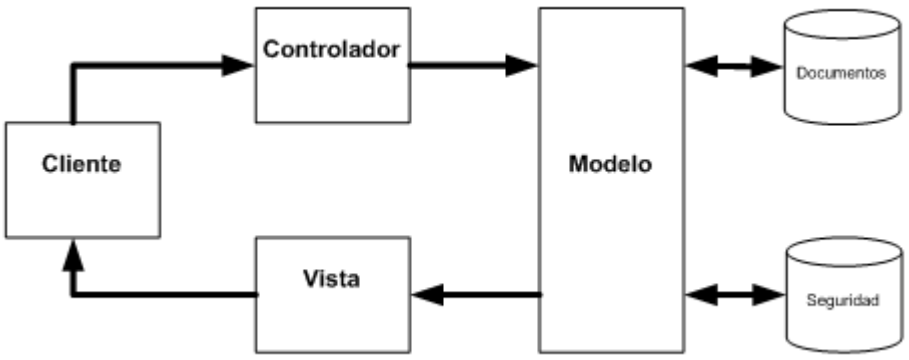

Figura 5. Arquitectura de OBELISCO.
Fuente: El autor de la investigación.

El patrón arquitectural permite dividir la aplicación en 3 componentes fundamentales; un primer componente que se llama ‘Modelo’ que contiene las reglas de negocio y la representación de los datos, un segundo llamado ‘Vista’ que presenta los datos del ‘Modelo’ y permite capturar información de los usuarios, y un ultimo componente llamado ‘Controlador’ que se encarga de controlar el flujo de trabajo entre los elementos del ‘Modelo’ y los elementos de presentación. Usando MVC se desacoplan las reglas de negocio y las interfaces de usuario, con lo que el sistema se hace más escalable y mantenible, permitiendo separar las tareas y roles de desarrollo. Separar las reglas del negocio de la presentación facilita la personalización de las interfaces, sin temor a afectar la lógica de negocio.

El ‘Modelo’ de OBELISCO estará conformado por una serie componentes que acceden el repositorio de documentos y de seguridad que están alojados en un Manejador de Base de Datos.

El componente ‘Vista’ se centra en el desarrollo de una serie de elementos, que permitirán mostrar el contenido depositado en el repositorio de documentos, usando para ello múltiples formatos y configuraciones de presentación.

El componente ‘Controlador’ dentro de la arquitectura es quien se encarga de captar las entradas de usuario, y junto con el componente ‘Vista’, sirven para desarrollar las interfaces de usuario, que permitirán la interacción con el sistema.

La Figura 6 muestra el Diagrama de Despliegue de OBELISCO.

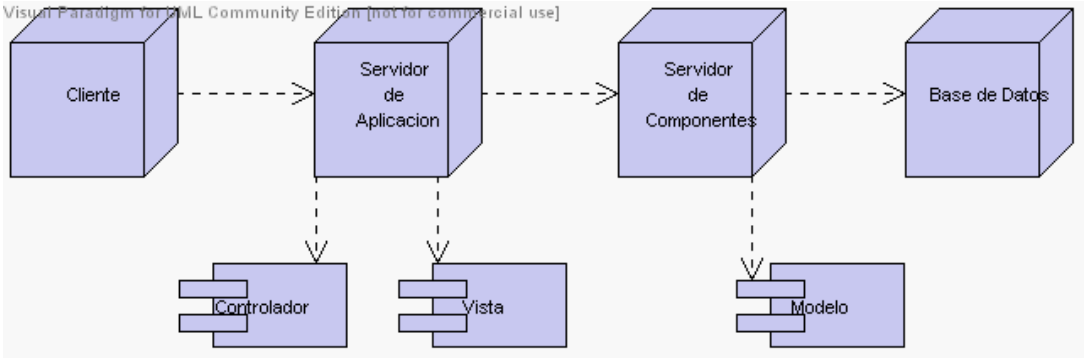

Figura 6. Diagrama de Despliegue de OBELISCO.
Fuente: El autor de la investigación.

La arquitectura permite desarrollar una aplicación distribuida, con la cual se pueden ubicar los distintos componentes de la aplicación en diferentes servidores. Es posible que la base de datos que aloja el repositorio de documentos se encuentre en un servidor independiente al del componente 'Modelo', así mismo el componente 'Modelo' cuenta con los mecanismos necesarios para que el componente 'Controlador' y 'Vista' puedan acceder al repositorio controlado y administrado por él.

Con el enfoque distribuido de la aplicación es posible identificar los cuellos de botellas en su ejecución y de esta manera es posible asignar los recursos de hardware necesarios al componente de software que más lo necesite

Usando este esquema de trabajo para el desarrollo y despliegue de la aplicación, se logran características importantes de mantenibilidad. Al dividir la aplicación en componentes funcionales bien diferenciados, es más fácil identificar el fragmento de código que es necesario cambiar para lograr la funcionalidad deseada.

Infraestructura y herramientas Tecnológicas

Dada la necesidad de que el sistema funcione en distintas plataformas de sistemas operativos, se selecciona como lenguaje de desarrollo el lenguaje Java, además es fácil encontrar APIs y Frameworks que pueden facilitar el desarrollo.

Por ser una aplicación distribuida se aprovecharon las tecnologías disponibles en la especificación J2EE. Dicha plataforma cuenta con elementos para el desarrollo de Aplicaciones Web y recursos para crear componentes distribuidos.

Para el desarrollo de OBELISCO que esta dirigido por un patrón arquitectural MVC, se utilizo STRUTS, framework de código abierto que implementa el patrón y tiene la ventaja de contar con el respaldo de una amplia base de desarrolladores. Adicionalmente posee herramientas muy poderosas para la validación de entradas de datos y para ayudar en la construcción del componente 'Vista'.

STRUTS en específico se usó para la construcción del componente 'Vista' y 'Controlador'. Ambos componentes se ejecutan en un componente de software llamado Tomcat que es un JSP-Servlet Container de fuente abierta.

El componente 'Modelo' de la aplicación se construyo usando la tecnología Enterprise Java Bean (EJB), que permite del desarrollo de componentes de negocio distribuidos, a los que podrá tener acceso el componente 'Controlador' de la aplicación usando el protocolo RMI. Los EJB que modelan el funcionamiento del sistema de gestión de contenidos estarán corriendo en el EJB Container de fuente abierta JBoss, que cumple con las especificaciones del Java Comunity Process (JCP) para la tecnología Enterprise Java Bean. En la Figura 7 se muestra la Arquitectura OBELISCO en la plataforma J2EE.

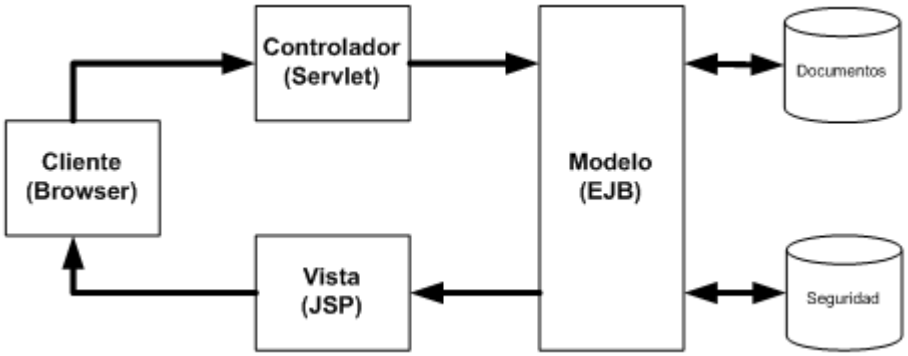

Figura 7. Arquitectura de OBELISCO en la plataforma J2EE.
Fuente: El autor de la Investigación

El repositorio de documentos estará alojado en el manejador de base de datos libre PostgreSQL, a él se conectara el componente 'Modelo' usando drivers de conexión JDBC.

La interacción entre el usuario y el sistema se realiza a través de un navegador Web, que usa el protocolo HTTP para enviar datos al componente 'Controlador' y recibir el contenido del componente 'Vista'.

XP se caracteriza por hacer hincapié en la refactorización del código producido y en la propiedad colectiva del mismo, en este sentido es importante contar con herramientas basadas en el lenguaje Java que permitan la prueba y verificación de los componentes desarrollados. Para esta actividad se selecciona el Framework JUNIT.

Lenguaje de Patrones

En esta sección se describen los patrones de diseño que serán incorporados en los distintos componentes del sistema, con el fin de proporcionarle la mantenibilidad y la eficiencia necesaria.

En el componente 'Modelo' se usan los siguientes patrones de diseño:

- Session Facade. Permite aislar la lógica del negocio del resto de la aplicación. De esta manera las aplicaciones necesitan conectarse a la red y llamar a una transacción dentro del EJB Container, para poder usar la lógica de negocio implementada.
- General Attribute Access Properties. Proporciona una interfaz independiente del dominio de aplicación para acceder a los atributos de un EJB de Entidad, ofreciendo mejoras en el mantenimiento y el rendimiento de los componentes desarrollados.
- Data Transfer Object. Establece un mecanismo para el intercambio de datos a través de la red, entre los componentes de software.
- JDBC Reading. Se refiere a una estrategia para mejorar los tiempos de respuesta de una aplicación que esta usando EJB de Entidad para consultar datos. Se basa principalmente en la incorporación de JDBC para esta actividad.
- Data Access Command Bean. Representa una manera de desacoplar un EJB de Entidad de la lógica y detalles de persistencia, haciendo más fácil su escritura y mantenimiento.
- Stored Procedure or Autogenerated Key. Describe en que momentos usar los servicios de un manejador de base datos para generar claves y como pueden usarse de manera portable en un EJB de Entidad.

En el componente ‘Controlador’ se incorporaron los siguientes patrones de diseño:

- EJBHomeFactory. Proporciona una forma para que la aplicación pueda interactuar con la capa EJB de la aplicación, definiendo para ello un “Singleton Factory” que contiene todas las los objetos Home y encapsula además la complejidad de instanciarlos y manejar los errores correspondientes.
- Business Delegate. Se usa para desacoplar la capa cliente de la capa donde esta implementado el patrón Session Facade, ocultando las complejidades que se encuentran dentro de la capa EJB.

El patrón EJBHomeFactory, es usado para realizar de una mejor manera la iteracción entre el componente ‘Controlador’ con el componente ‘Modelo’. El intercambio de datos entre ambos será dirigido por el patrón Data Transfer Object.

El componente ‘Controlador’ accede la lógica del negocio establecida a través del patrón Session Facade en el componente ‘Modelo’, usando para ello el patrón Business Delegate.

Organización del Código Fuente

Uno de los principios de la metodología XP es su orientación al código, en este sentido es importante plantear una organización del código fuente, con el objeto de mejorar su mantenimiento. En función a la arquitectura planteada para el sistema, se emplea una organización del código fuente por componentes. Las Tablas 3 y 4 muestran la organización por paquetes de los componentes ‘Modelo’ y ‘Controlador’.

Tabla 3: Paquetes del Componente 'Modelo'.

Paquete	Descripción
com.obelisco.dao.*	Contiene las clases con la lógica de manejo de persistencia, a ser usados por los EJB de Entidad.
com.obelisco.dto.*	Contiene las definiciones de las clases que sirven para el intercambio de datos entre los distintos componentes del sistema.
com.obelisco.ejb.*	Contiene los EJB de Sesión, Entidad y de Mensajes.
com.obelisco.interfaces.*	Contiene las Interfaces Locales y Remotas de los EJB de Sesión, Entidad y de Mensaje.
com.obelisco.pk.*	Contiene las Clases de tipo Primary Key para los EJB de Entidad.

Fuente: El autor de la investigación.

Tabla 4: Paquetes del Componente 'Controlador'.

Paquete	Descripción
com.obelisco.struts.formas	Contiene las especializaciones de la clase ActionForm del Framework STRUTS que sirven para validar las entradas de usuario del lado del cliente y del servidor.
com.obelisco.struts.acciones.*	Contiene las especializaciones de la clase Action del Framework STRUTS que maneja las peticiones de los clientes y las redirecciona a la lógica de negocio depositada en el componente 'Modelo' o a la capa de presentación representada por el componente 'Vista'
com.obelisco.utilidades.*	Clases de utilidad general, como por ejemplo clases dedicadas a la conexión con el componente 'Modelo'.

Fuente: El autor de la investigación.

El componente ‘Vista’ esta compuesto por una serie de archivos JSP que sirven para presentar los datos del ‘Modelo’ y las interfaces de usuario necesarias para la interacción con el sistema.

Iteración N° 1

Objetivo

Desarrollar las siguientes funcionalidades para el sistema:

- Consulta Documentos de usuario.

Planificación

A partir de este momento el desarrollo de cada iteración se basa en un diagrama de casos de uso que especifica las tareas a ejecutar, la Figura 8, muestra el conjunto de tareas necesarias para Consultar Contenido.

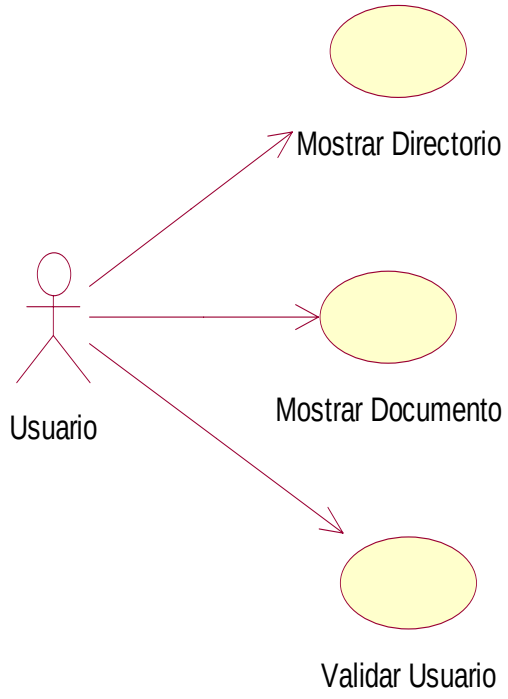

Figura 8. Diagrama de Casos de Uso para Consultar Contenido, Iteración N° 1.
Fuente: El autor de la Investigación.

Para comenzar el desarrollo del sistema es necesario plantear un esquema de seguridad que regule la entrada de los usuarios y las actividades que éstos puedan ejecutar en los directorios y documentos depositados en el sistema. Para este esquema se utilizaron las clases del diagrama en la Figura 9.

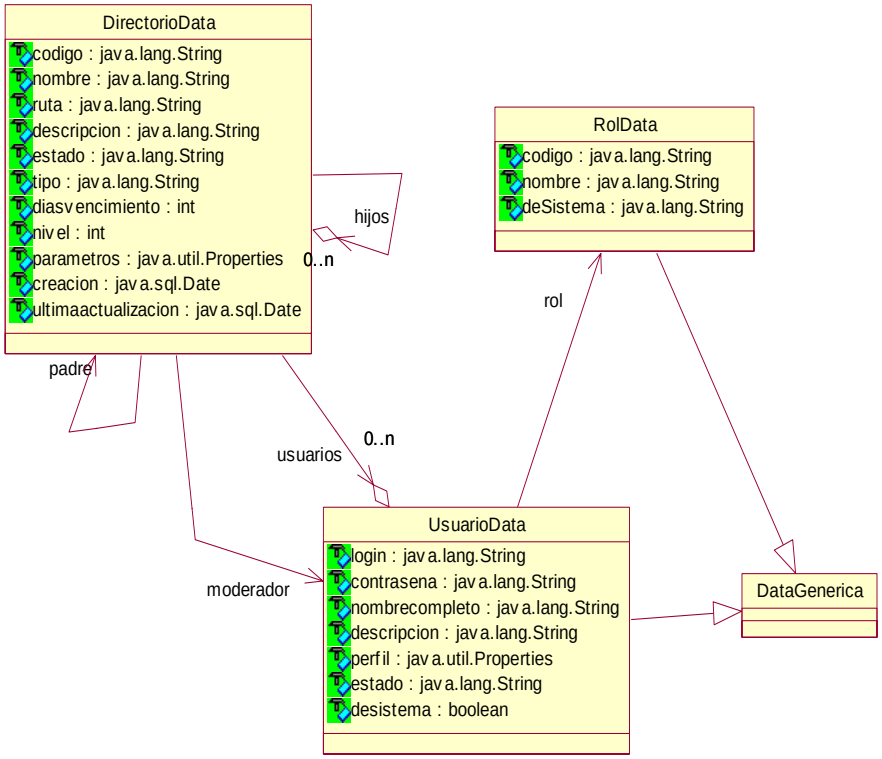

Figura 9. Diagrama de Clases que modela la seguridad de OBELISCO.
Fuente: El autor de la investigación.

El diagrama expone la posibilidad de que cualquier usuario en el sistema tenga un Rol, de tal forma es posible determinar si puede ejecutar determinadas opciones administrativas. En el caso en que un usuario tenga el Rol Administrador podrá ejecutar funciones como crear usuarios, modificar la estructura de directorios o definir los documentos que forman parten de la página principal del sistema.

La Figura 9 igualmente muestra como un Directorio es un agregado de usuarios, esto es importante para filtrar las tareas que puede ejecutar cada uno de ellos, es así como un usuario puede escribir (Publicar Documentos) o leer (Consultar Documentos) sobre un determinado directorio.

En la Figura 9 también se muestra la presencia de un usuario llamado moderador, cuyas labores se describirán en la Iteración N° 3.

Para introducir el manejo de documentos, la Figura 10 presenta a las clases documento, recurso, movimiento, presentación y tipo de documento.

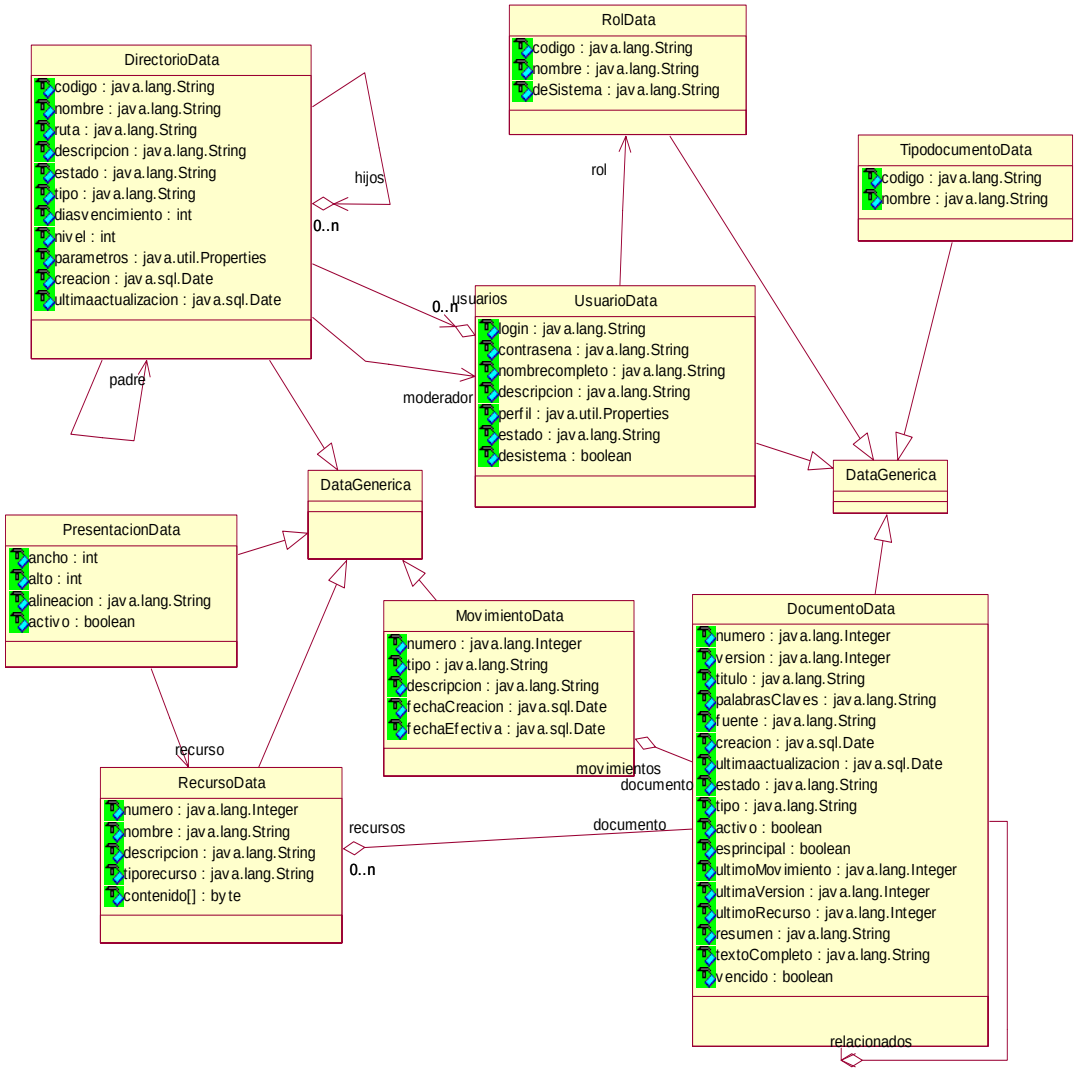

Figura 10. Diagrama de Clases que modela el contenido en OBELISCO.
Fuente: El autor de la investigación.

En cada documento del sistema OBELISCO existe una lista de recursos, que es un conjunto de archivos de sistema operativo que aportan información de valor al documento. Los recursos se usan para construir la presentación de un documento y sirve para introducir al contenido del mismo.

La Figura 10 muestra como la clase documento tiene una relación de agregado con ella misma, esto es motivado al hecho de que un documento puede estar relacionado con otros depositados en el sistema. También se aprecia una relación de documento con tipo de documento, esto se hace para tener un criterio de clasificación y organización en los documentos creados dentro de OBELISCO.

El Diagrama de Clases de la Figura 10 presenta a una clase movimiento, esta se usa para registrar los cambios de estado que ha sufrido un documento, producto de situaciones generadas por su creador o por el moderador del directorio donde reside.

La persistencia de los objetos derivados del diagrama de clases de la Figura 10 será controlada, a través del uso de EJB de Entidad, la plataforma J2EE define para un EJB de Entidad, tres componentes fundamentales un Bean, una interfaz Remota y una interfaz Home, existe un componente opcional que es una clase de tipo PK usada para los procesos de búsqueda. Al hacer un despliegue del EJB sobre el EJB Container el acceso a este se hace usando las interfaces Home y Remota. La Figura 11 muestra la organización de todas estas clases.

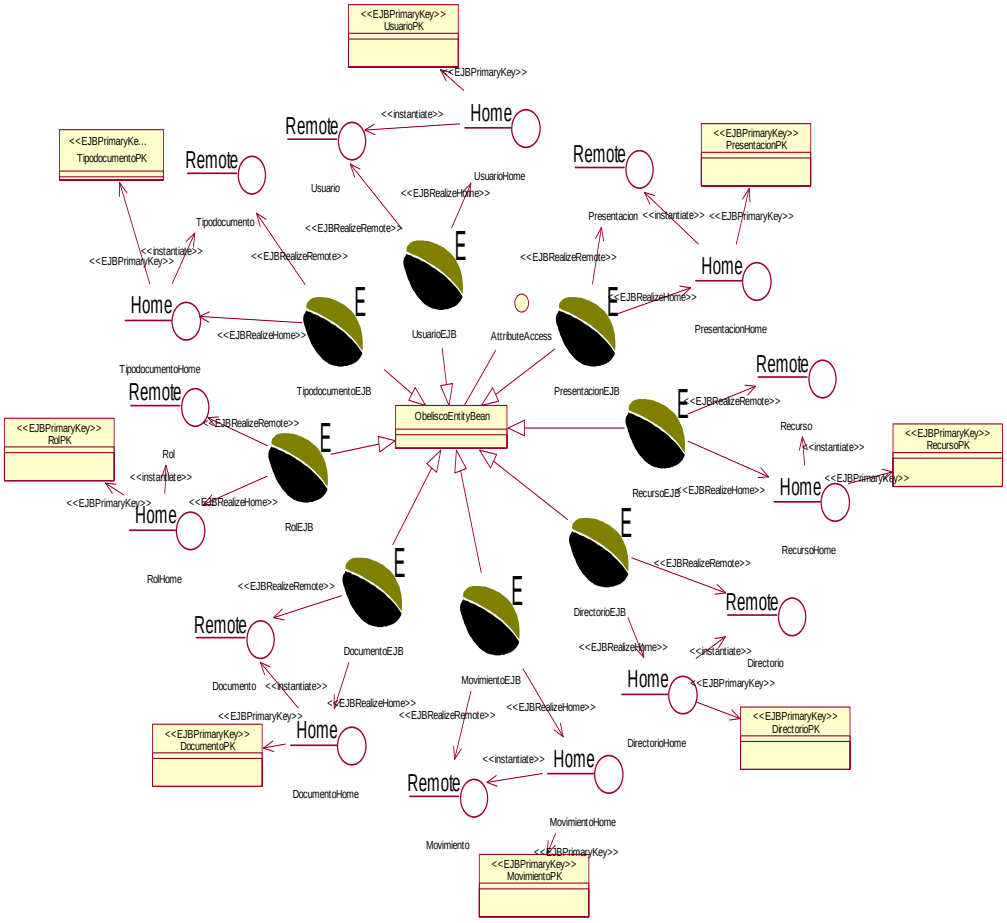

Figura 11. Diagrama de Clases de los EJB de Entidad de OBELISCO.
Fuente: El autor de la investigación.

Dada la cantidad de clases de conforman la capa de manejo de persistencia, se presenta en detalle el EJB de Entidad Documento, Figura 12, allí se muestra el tipo de interacción que existe entre el Bean, las interfaces y la clase de tipo PK, esta situación es similar para el resto de los EJB de Entidad.

Figura 12. Diagrama de Clases del EJB de Entidad Documento.
Fuente: El autor de la investigación.

En la Figura 12 se observa una interfaz llamada Generic AttributeAccess, esta se usa para implementar el Patrón de diseño Generic Attribute Access que mejora el acceso a los atributos de los EJB de Entidad creados en el sistema. En el diagrama de la Figura 13 se aprecia la implementación de este patrón.

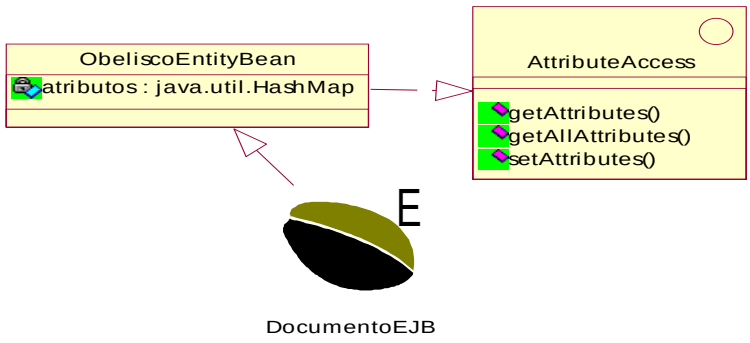

Figura 13. Diagrama de Clases del patrón de Diseño Generis Attribute Access.
Fuente: El autor de la Investigación

Para el manejo de la persistencia de los objetos, se puede hacer uso de un manejador de base de datos, archivos XML u objetos serializados. Para lograr un desacoplamiento de los EJB de la lógica de persistencia, se recurrió al patrón de diseño Data Access Object Command, originando una serie de clases que estarán al servicio de un EJB de Entidad determinado. De esta forma el EJB de Entidad Documento usará la clase DocumentoDAO para maneje los detalles de implementación con respecto al almacenamiento de objetos de su tipo. La Figura 14 muestra la serie de objetos generados.

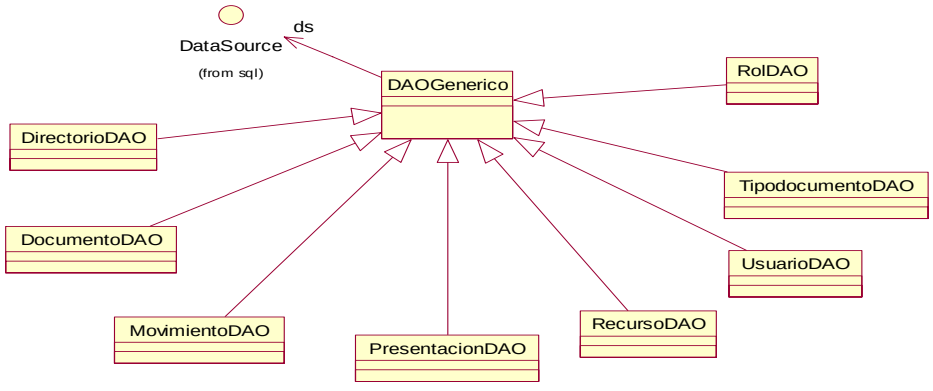

Figura 14. Diagrama de Clases del Patrón de Diseño DAO.
Fuente: El autor de la investigación

Para lograr la persistencia todas las clases DAO generadas en OBELISCO usan un manejador de base de datos, aprovechando de esta manera características presentes en los EJB Container. En el caso particular de JBoss, maneja un conjunto de conexiones a bases de datos, con las cuales es posible compartir un pequeño número de conexiones a la base de datos entre una gran cantidad de objetos, esto permite optimizar los recursos dentro del servidor que contiene el manejador de base de datos. La Figura 15 muestra la forma en que se relaciona el EJB Documento con la clase Documento DAO para almacenar los objetos.

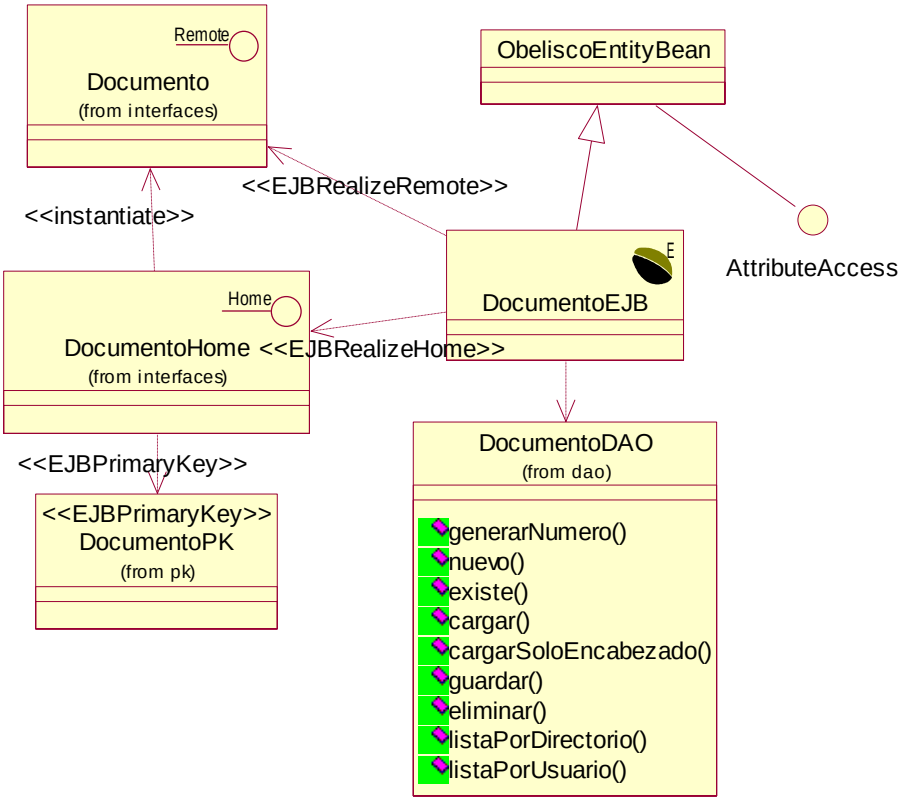

Figura 15. Diagrama de Clases del EJB de Entidad Documento.
Fuente: El autor de la investigación

En OBELISCO las operaciones y procedimientos que definen la lógica para el manejo de directorios y documentos están encapsulados en un conjunto de EJB de Sesión (Patrón de Diseño Session Facade). Todos se apoyan en los EJB de Entidad para guardar y extraer objetos del repositorio de objetos, así como en objetos de tipo DAO para ejecutar procesos de consulta. Esto último es recomendado por el patrón de diseño JDBC Reading, para garantizar un mejor desempeño a la hora de consultar contenidos en el sistema.

Para el manejo de las políticas de seguridad y el control de acceso a los distintos directorios y documentos creados en OBELISCO, se han construido dos EJB de Sesión, llamados Seguridad y GestionarUsuario. El primero esta dedicado a verificar las credenciales de usuario para entrar al sistema. El segundo determina los directorios que puede consultar un usuario, los directorios en los cuales este se comporta como moderador y es capaz de devolver los documentos que deben mostrarse en la página principal del usuario. La confección de estos EJB se puede apreciar en la Figura 16.

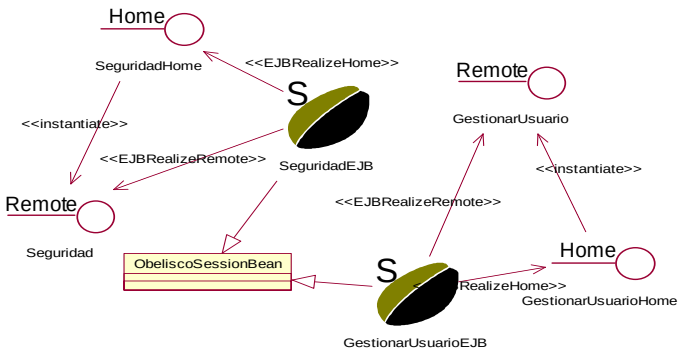

Figura 16: Diagrama de Clases de los EJB de Sesión Seguridad y GestionarUsuario
Fuente: El autor de la investigación.

Para satisfacer el objetivo de esta iteración, es necesario crear un EJB de Sesión que encapsule el proceso de consulta de contenido, esto es, consultar directorios y consultar documentos. Para mostrar este efecto se presenta el diagrama de la Figura 17.

Figura 17. Diagrama de Clases del EJB de Sesión ConsultarContenido.
Fuente: El autor de la investigación.

De esta forma el componente 'Modelo' queda estructurado por el conjunto de EJB de Sesión, EJB de Entidad y clases de tipo DAO que se han nombrado hasta el momento.

Para resolver el problema relacionado a la forma en que el componente 'Controlador' transferirá datos al componente 'Modelo', y viceversa, se recurre al

patrón de diseño Data Transfer Object (DTO), de esta manera las clases especificadas en la Figura 10 implementaran la interface java.io.Serializable, que garantiza su transportabilidad a través de la Red, usando la plataforma J2EE.

En OBELISCO la confección de la conexión del componente 'Controlador' con el componente 'Modelo', se realiza instanciando el patrón de diseño EJBHomeFactory, el cual es usado para optimizar la conexión de una aplicación cliente con un EJB Container. La Figura 18 muestra la forma como fue realizado dentro de OBELISCO.

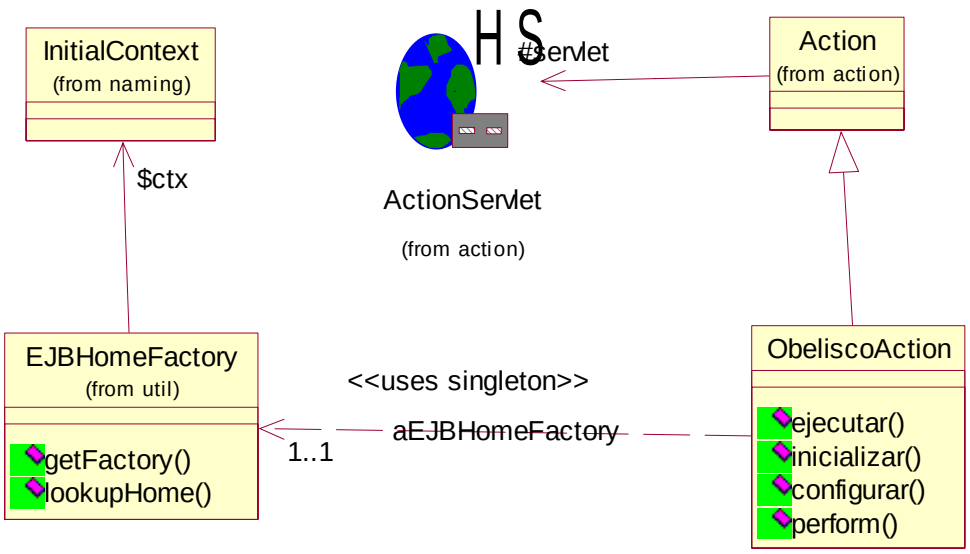

Figura 18. Diagrama de Clases del Patrón de diseño EJBHomeFactory.
Fuente: El autor de la investigación.

El diagrama de la Figura 19 muestra la manera como están organizadas las clases que forman el componente 'Controlador' de OBELISCO, se puede observar que hay una clase llamada ObeliscoAction. Ella encapsula el proceso de conexión al componente 'Modelo' y permite acceso a los EJB de Sesión, GestionarUsuario y Seguridad, que se encargan de los procesos de seguridad y control de acceso dentro

del sistema. Debido al framework Struts y para disfrutar de todas sus bondades es necesario que ObeliscoAction sea subclase de org.struts.action.struts.Action.

ObeliscoConsulta es una especialización de ObeliscoAction y se encarga de dar acceso a la lógica de negocio encapsulada en el EJB de Sesión ConsultarContenido, lógica que permite el despliegue de directorios, documentos y recursos almacenados en el sistema.

Todas las clases del componente 'Controlador' que se usaron para satisfacer los requerimientos funcionales en esta iteración son especializaciones de la clase ObeliscoConsulta, Figura 19.

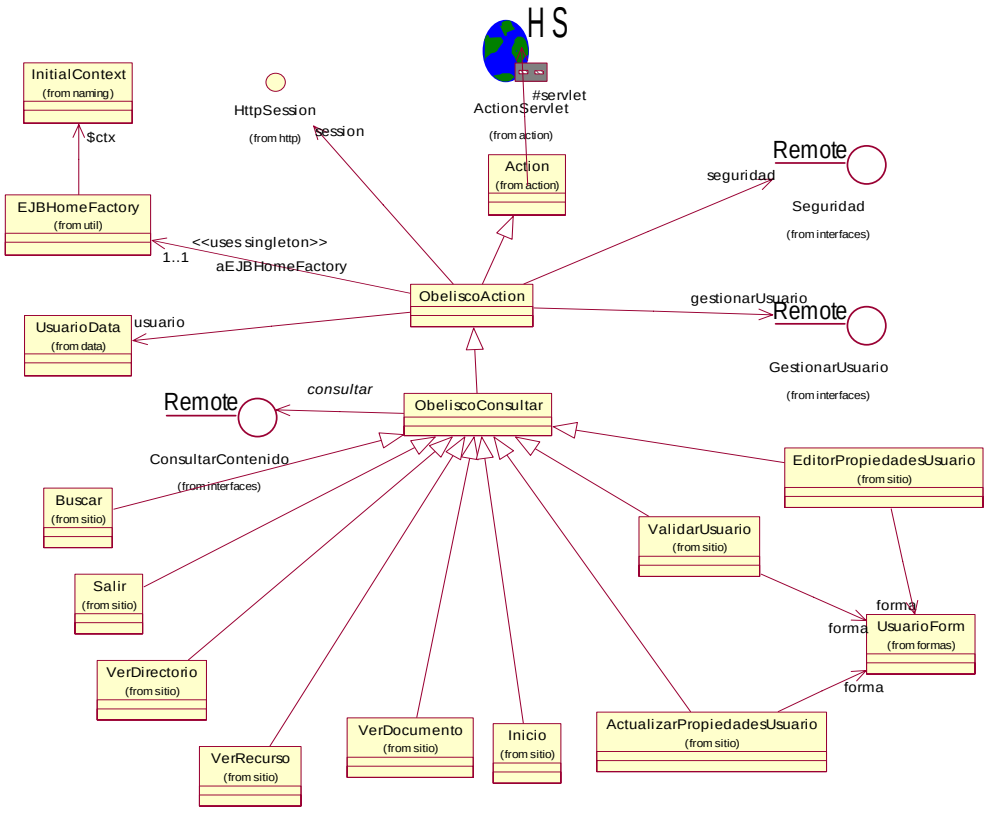

Figura 19. Diagrama de Clases del Componente 'Controlador', Iteración N°1.
Fuente: El autor de la investigación

Dentro del el componente 'Controlador' se han creado especializaciones de org.apache.struts.action.ActionForm para manejar las entradas de usuario, ellas trabajan en conjunto con las clases derivadas de org.apache.struts.action.Action para realizar procesos de validación, Figura 20.

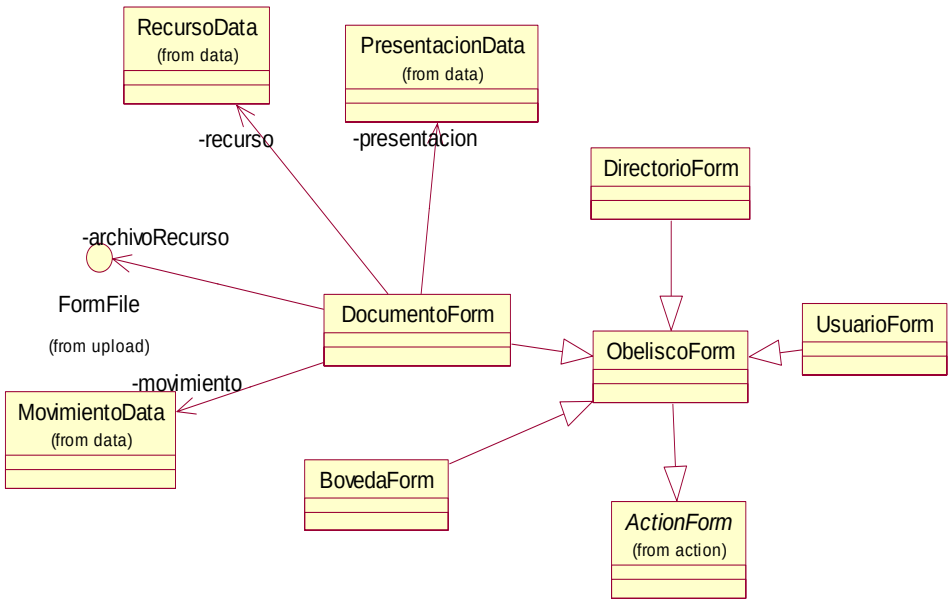

Figura 20. Diagrama de Clases, para las Clases de tipo Form,
Fuente: El autor de la investigación.

El componente 'Vista' es una serie de páginas JSP que heredan las características de una plantilla. Como lo muestra la Figura 21, la plantilla tiene cinco elementos funcionales que pueden recibir mantenimiento por separado: 1) El encabezado que es el elemento que esta presente en la parte superior de la página. Este componente mantiene el logo del sistema. 2) El menú que abarca todas las opciones u operaciones que maneja OBELISCO. 3) El pie de página comprende información con respecto a los derechos de autor u otra información que corresponda a los autores o administradores del sistema. 4) El cuerpo que despliega la información y las formas de trabajo y de administración del sistema. 5) El componente publicidad encargado

de mostrar información publicitaria relacionada a las áreas de conocimientos que maneja el sistema.

Figura 21. Apariencia de plantillaObelisco.jsp.
Fuente: El autor de la investigación

Desde un enfoque orientado a objeto las afirmaciones anteriores se pueden plantear como un diagrama de clase que se muestra en la Figura 22.

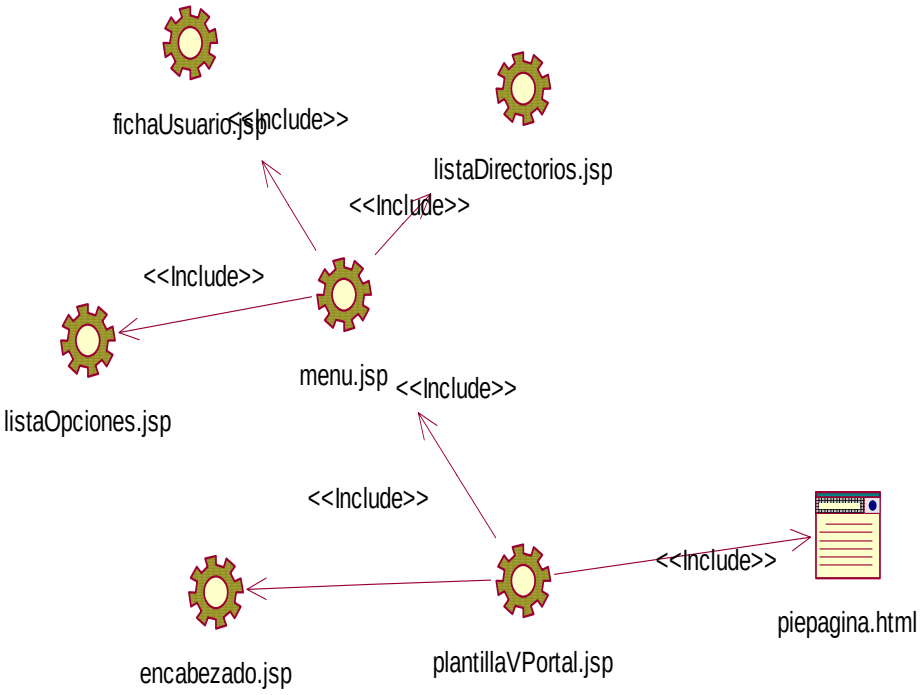

Figura 22. Diagrama de Clases de plantillaObelisco.jsp.
Fuente: El autor de la investigación

Para desplegar la información solicitada por el usuario, cada clase del componente 'Vista', utiliza un objeto de tipo HttpSession generado por el Servlet Container. Estas extraen la información que ha sido depositada, por las clases del componente 'Controlador' y que proviene a su vez del componente 'Modelo'. En detalle el proceso de comunicación entre todos los componentes ocurre de la siguiente manera: 1) El usuario del sistema recurre a una clase del Componente 'Controlador' para obtener ciertos servicios. 2) Esta clase analiza las entradas del usuario, las valida y recurre al EJB de Sesión correspondiente. 3) Los datos que se obtienen de éste son colocados en un objeto de tipo HttpSession y luego se redirecciona la ejecución del programa a la clase del componente 'Vista' que hereda toda las características de plantillaObelisco.jsp.

El diagrama de la Figura 23 muestra las relaciones que existen entre las clases del componente ‘Controlador’ y las clases del componente ‘Vista’.

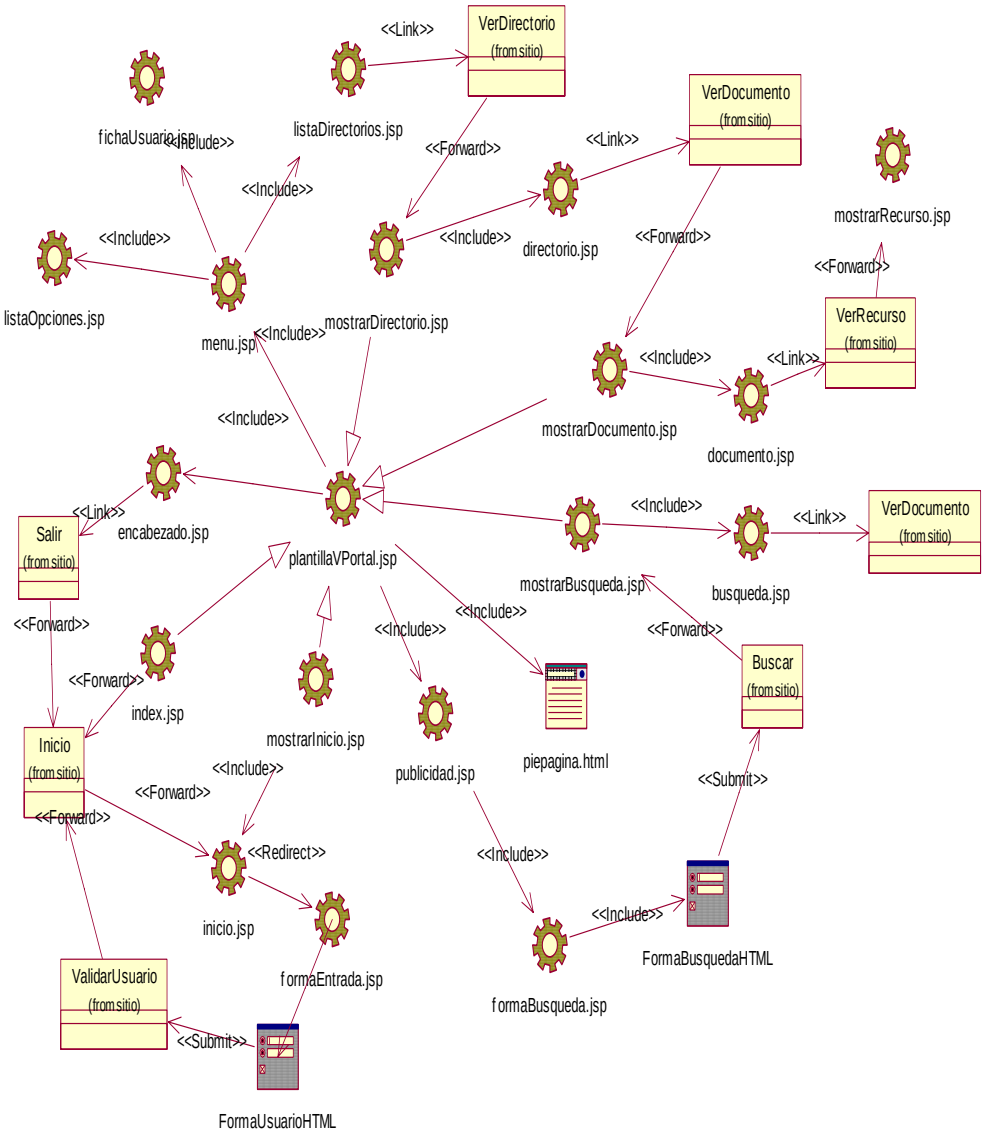

Figura 23. Diagrama de Clases del Componente ‘Vista’, Iteración N°1.
Fuente: El autor de la investigación

Para tener una idea sobre la forma en que todos los componentes del sistema interactúan, se plantea un diagrama de interacción, Figura 24, que muestra los pasos a seguir para visualizar un determinado documento.

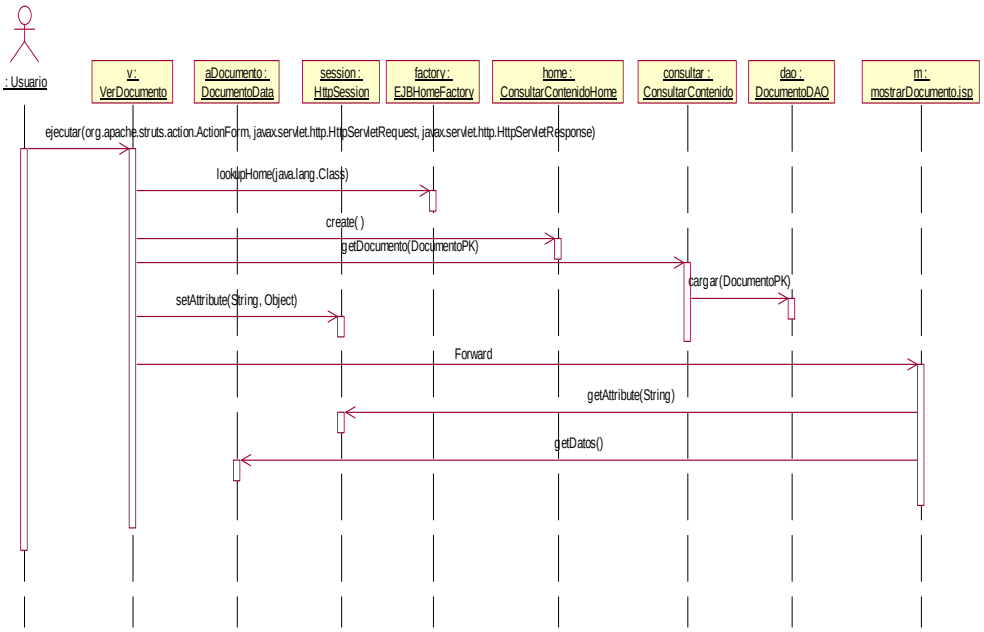

Figura 24. Diagrama de Interacción del proceso Consultar Documento.
Fuente: El autor de la investigación.

Desarrollo

Partiendo del proceso de planificación donde se marcó la pauta para definir cada una de las clases que conforman los componentes de la aplicación, se procedió a desarrollar los módulos que satisfacen los objetivos planteados al principio de esta iteración. La Figura 25 muestra la página de inicio del sistema, el lado izquierdo presenta una región denominada 'Secciones', en donde se coloca la lista de directorios que puede consultar el usuario actual y al centro una serie de documentos, que se llaman documentos principales.

Figura 25. Pantalla de Inicio de OBELISCO.
Fuente: El autor de la investigación.

Cuando un usuario necesite conocer el contenido de un directorio, debe hacer click en alguno de los enlaces desplegados dentro del área llamada ‘Secciones’. Al hacerlo se muestra una ventana similar a la de la Figura 26, en donde se aprecian los documentos principales de ese directorio, indicando su titulo, resumen, fecha de la ultima actualización e imagen de presentación.

Figura 26. Pantalla de Consulta de Directorio.
Fuente: El autor de la investigación.

Cuando se despliega un directorio en la página de inicio del sistema, el contenido de un documento lo puede conocer un usuario haciendo click en su titulo o imagen de presentación. El efecto de esta acción es una pantalla como la de la Figura 27. Allí se aprecian detalles como, el Directorio al cual pertenece, titulo, resumen, la fecha de última actualización, el usuario que lo genero y una sección con el conjunto de recursos o archivos que le han sido anexados.

Figura 27. Pantalla de Consulta de Documento.
Fuente: El autor de la investigación.

Iteración N° 2

Objetivo

Desarrollar las siguientes funcionalidades para el sistema:

- Administrar los Documentos de Usuario

Planificación

Para el desarrollo de esta iteración se utilizara el diagrama de casos de uso de la Figura 28.

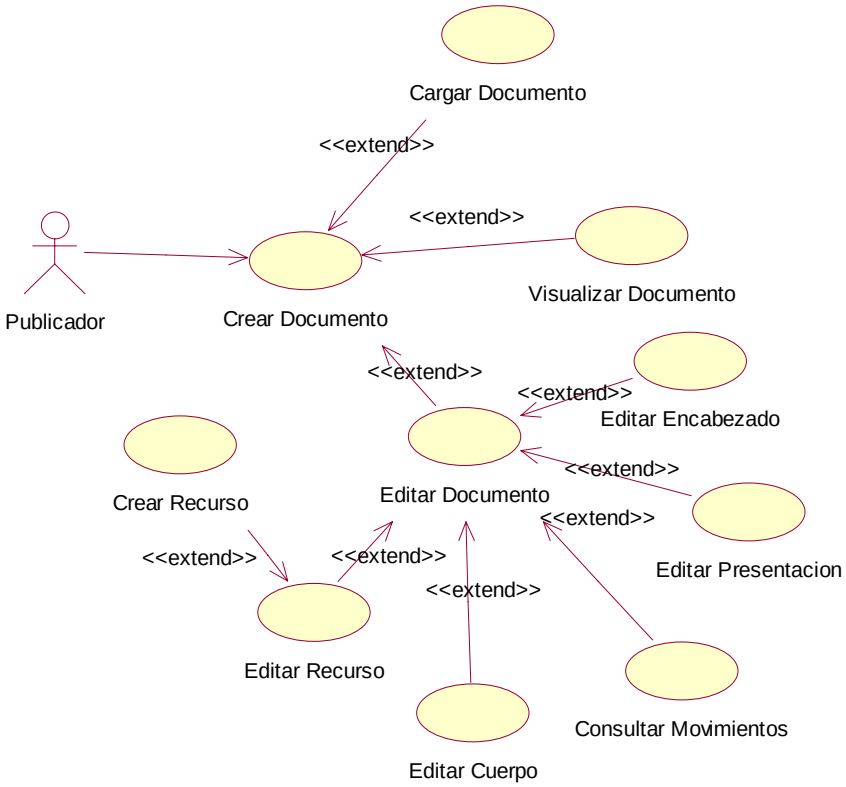

Figura 28. Diagrama de Casos de Uso para Administrar Documentos del Usuario, Iteración N° 2.
Fuente: El autor de la investigación

En esta se desarrollan todas las actividades relacionadas al proceso de crear un documento como:

- Crear el Encabezado y Cuerpo.

- Definir los recursos que lo conforman.
- Establecer la imagen de presentación.
- Consultar los Movimientos del Documento

La consulta de Movimientos en un documento es una actividad muy importante para el publicador porque permite mejorar el proceso de confección y construcción de documentos dentro del sistema. Los movimientos son generados por el moderador del directorio donde reside, ocurren por aprobación o rechazo. Cuando un documento es aprobado se le puede definir la fecha de publicación; en caso contrario se define la razón por la cual fue rechazado. El diagrama de estado que se presenta en la Figura 29 muestra la situación.

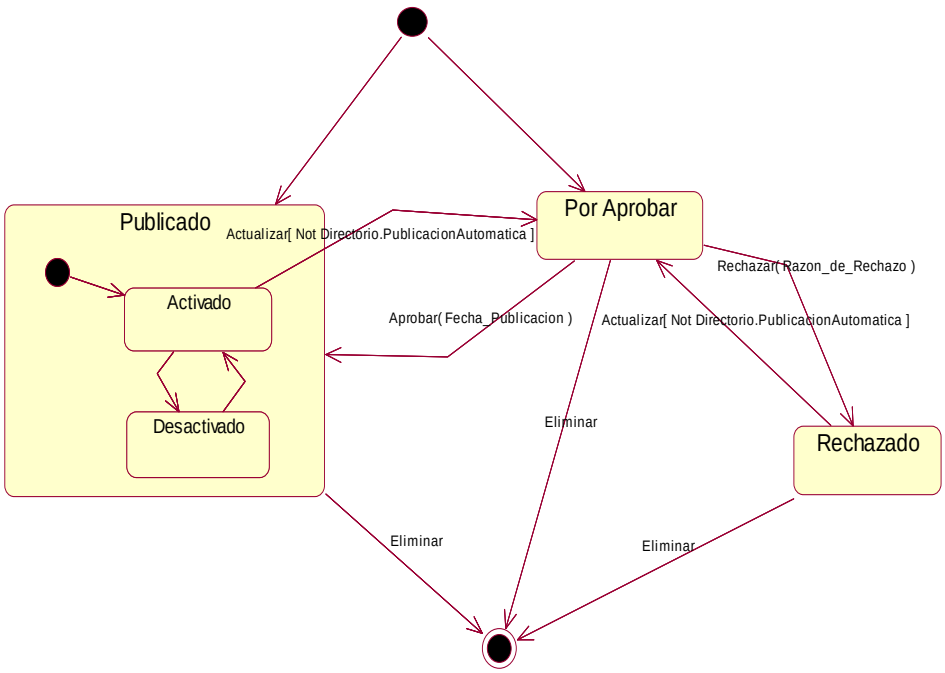

Figura 29. Diagrama de Estado correspondiente a un Documento.
Fuente: El autor de la investigación.

Obsérvese como una vez creado el documento puede pasar directamente al estado Publicado. Esto ocurre cuando el directorio que lo contiene ha sido definido por el moderador como No Moderado.

Por lo tanto existen directorios que son moderados, es decir necesitan de la intervención del moderador para la publicación de sus documentos y directorios No Moderados, es decir los documentos que son creados en él son aprobados automáticamente.

En función a los objetivos trazados para esta iteración se necesitan crear varias clases en los componentes 'Modelo', 'Vista' y 'Controlador'. Se comenzará por modelar las actividades de generación de contenido en un EJB de Sesión llamado PublicarContenido, cuya estructura se puede apreciar en la Figura 30.

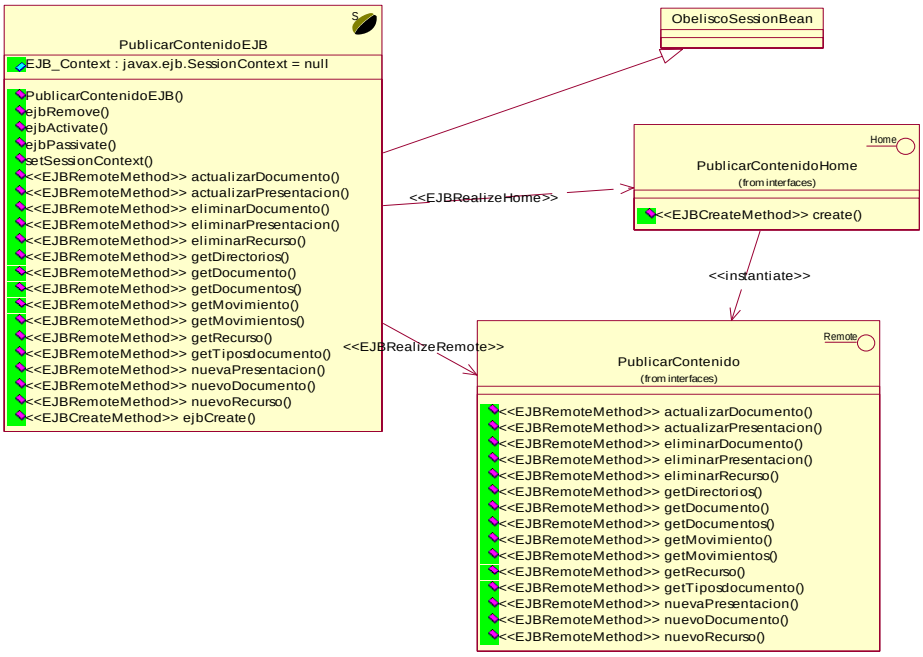

Figura 30. Diagrama de Clases del EJB de Sesión PublicarContenido.
Fuente: El autor de la investigación.

En el componente control se crea una clase ObeliscoPublicacion, que se encarga de instanciar al EJB de Sesión PublicarContenido. Con ello se pone la lógica de negocio relacionada a la publicación de documentos al servicio del resto de clases que derivan de ella.

ObeliscoPublicación se convierte en una especialización de ObeliscoConsulta, esto hace posible utilizar el EJB de Sesión GestionarUsuario, para validar permisología cuando se realizan procesos de escritura en un directorio. La Figura 31 muestra las nuevas clases de tipo control generadas y la forma en que están relacionadas con objetos de tipo ActionForm que facilitan el proceso de validación de los datos.

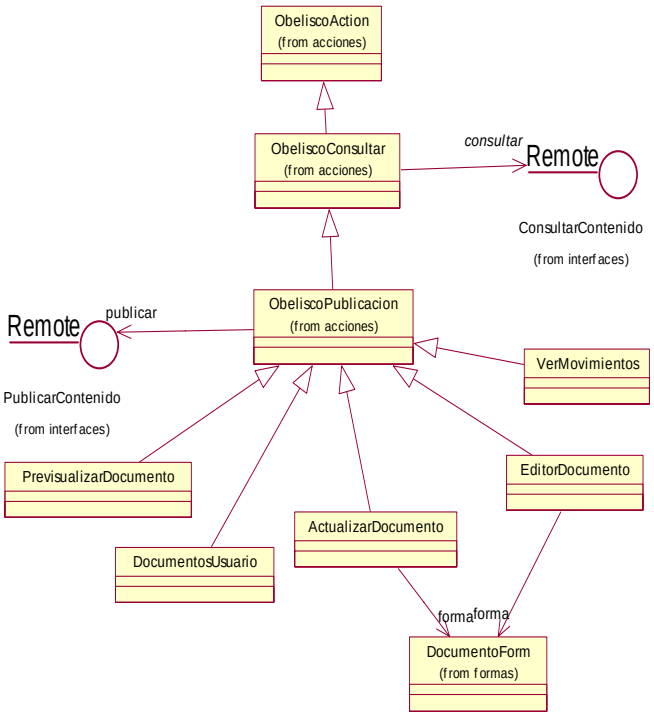

Figura 31. Diagrama de Clases del Componente 'Controlador', Iteración N° 2.
Fuente: El autor de la investigación.

El diagrama de la Figura 32 muestra las clases en el componente 'Vista' y la manera en que se relacionan con las clases del componente 'Controlador'.

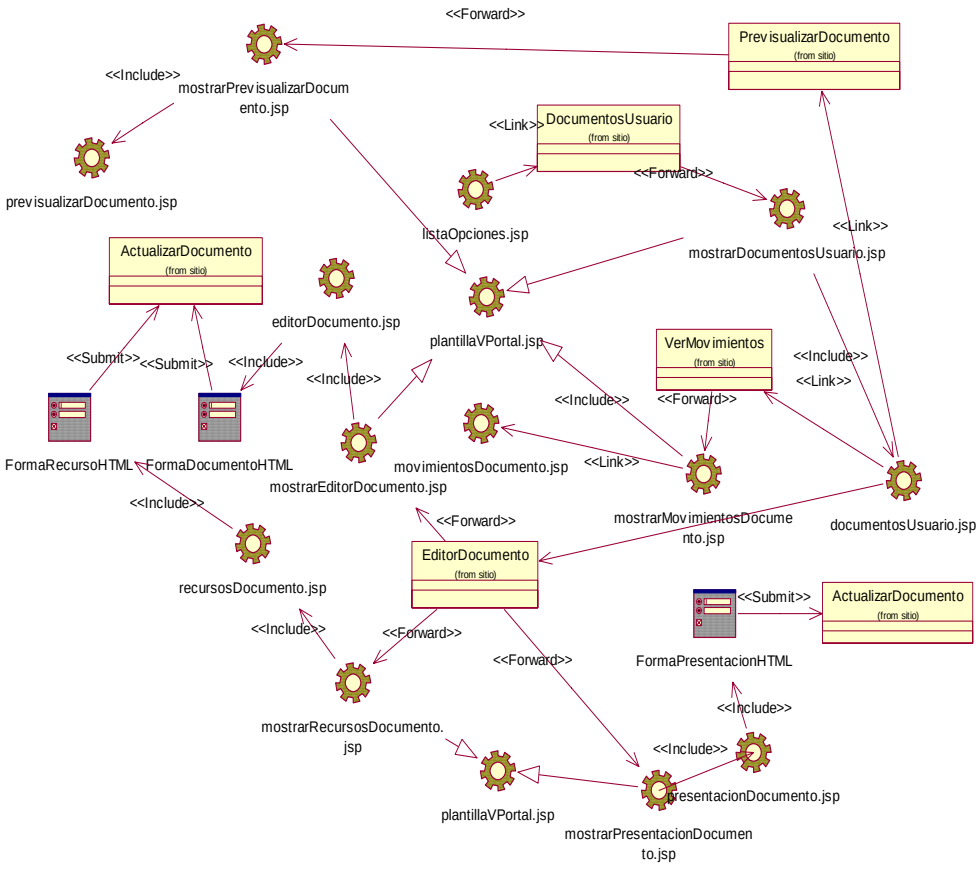

Figura 32. Diagrama de Clases del Componente 'Vista', Iteración N° 2.
Fuente: El autor de la investigación.

Como una de las actividades más importantes es el almacenamiento de un documento, el diagrama de la Figura 33 muestra el orden en cual las actividades deben desarrollarse para lograr este objetivo.

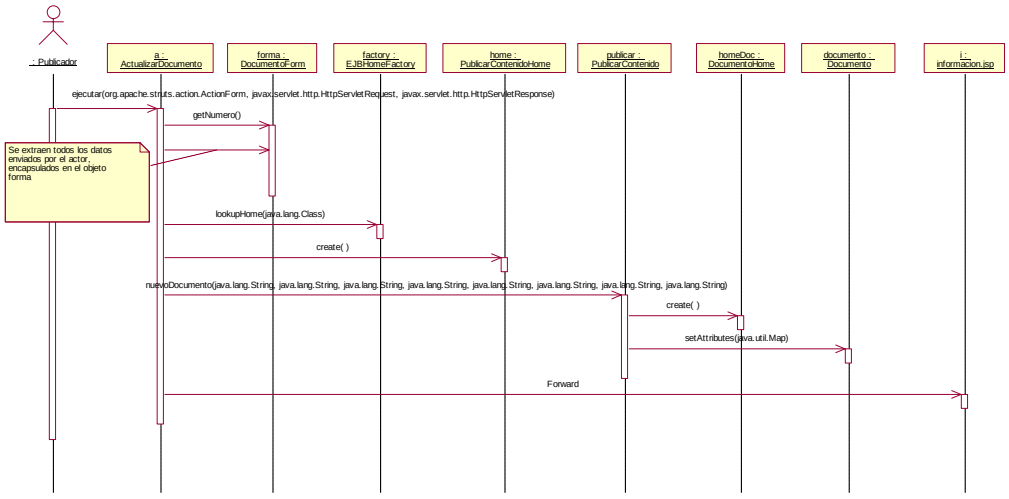

Figura 33. Diagrama de Interacción del proceso Crear Documento.
Fuente: El autor de la investigación.

Desarrollo

Para determinar si un usuario puede usar las funcionalidades de publicación de contenido, necesita ser validado. Para ello debe proporcionar las credenciales en la pantalla que se muestra en la Figura 34.

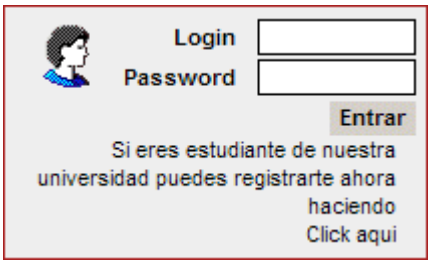

Figura 34. Pantalla de entrada a OBELISCO.
Fuente: El autor de la investigación.

Si se comprueba que las credenciales proporcionadas son correctas, se muestra la pantalla de la Figura 35, en donde se puede detallar: 1) El menú de Secciones, que contiene un listado actualizado de los directorios sobre los cuales el usuario actual puede hacer procesos de consulta. 2) El menú de opciones del Sistema. 3) El menú de opciones del usuario.

Figura 35. Pantalla de OBELISCO cuando las credenciales de usuario son validadas y aprobadas.
Fuente: El autor de la investigación.

El menú de opciones del sistema es similar a la Figura 36, con él se ejecuta el 'Administrador de Documentos', que se usa para crear, editar y conocer el estatus de los documentos que un usuario ha creado en el sistema.

Figura 36. Menú de opciones de OBELISCO, Iteración N° 2.
Fuente: El autor de la investigación

Con el 'Administrador de Documentos' es posible realizar varias tareas, como definir la presentación de un documento, modificar la estructura de sus recursos, ver sus movimientos y editar su contenido, Figura 37.

Figura 37. Administrador de Documentos.
Fuente: El autor de la investigación.

Al hacer click en el enlace Nuevo Documento o en el título del documento dentro del 'Administrador de Documentos', se ejecuta el 'Editor de Documento'. Esta

herramienta es usada para definir las estructura de un documento: titulo, resumen, directorio al cual pertenece y texto completo, Figura 38.

Figura 38. Editor de Documento.
Fuente: El autor de la investigación

Con el 'Administrador de Documentos', se llama al 'Editor de Presentación', encargado de definir la imagen de presentación, que es un recurso que se emplea para introducir al usuario dentro del contenido de un documento, Figura 39.

Figura 39. Editor de Presentación.
Fuente: El autor de la investigación

Un documento puede ser visto como un conjunto de recursos, dentro de OBELISCO, el cual puede ser controlado a través del 'Administrador de Recursos', que permite agregar y eliminar archivos que se han anexado a un documento, Figura 40.

Figura 40. Administrador de Recursos.
Fuente: El autor de la investigación

Una vez que el usuario ha creado un documento, el moderador genera los movimientos que inciden en su estado. Esto pueden ser consultados por el propietario del documento, a través del 'Visor de Movimientos', que se llama desde el 'Administrador de Documentos', Figura 41.

Figura 41. Visor de Movimientos.
Fuente: El autor de la investigación

Iteración N° 3

Objetivo

Desarrollar las siguientes funcionalidades para el sistema:

- Moderar Directorio.

Planificación

El diagrama de casos de uso que se presenta en la Figura 42 muestra las funcionalidades que deben desarrollarse.

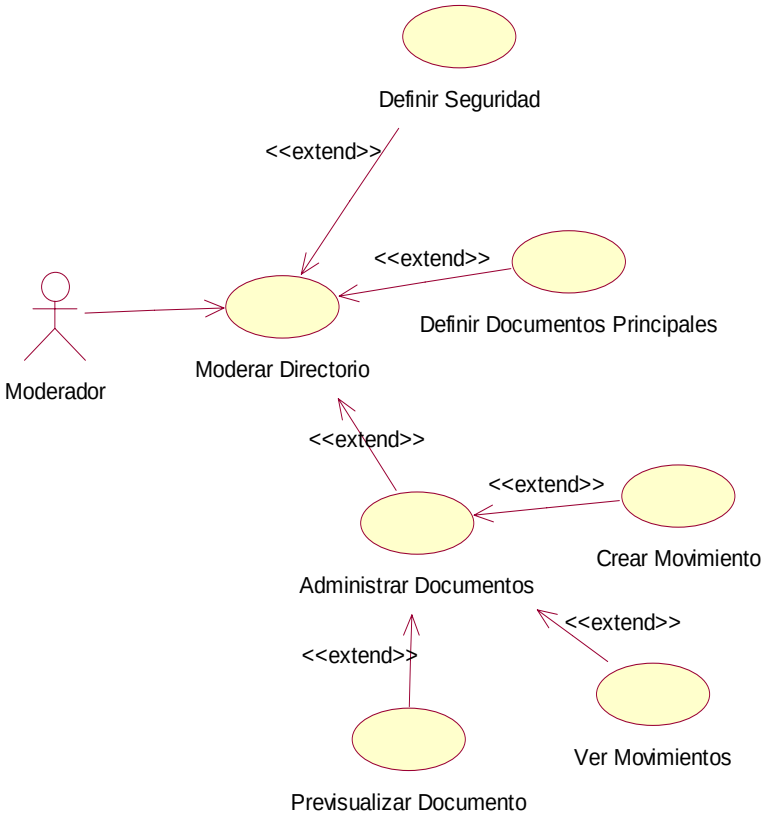

Figura 42. Diagrama de Casos de Uso para Moderar Directorio, Iteración N° 3.
Fuente: El autor de la investigación

En esta iteración se desarrollaron los componentes de software necesarios para que un moderador pueda realizar actividades de control, monitoreo y administración sobre un directorio, esto incluye:

- Definir datos generales: Esto es actualizar los datos del directorio que tienen incidencia en la forma en la cual se administran los documentos depositados sobre él. Por ejemplo, tiempo de vencimiento de los documentos, aprobación obligatoria de documento.
- Definir la seguridad: La seguridad del directorio se maneja en función a dos grupos de usuario, un grupo de lectura, en el cual están los usuarios que pueden consultar los documentos publicados y un grupo de escritura, que incluye a los usuarios del sistema que pueden crear documentos.
- Administrar los documentos principales y secundarios: Esta funcionalidad establece cuales son los documentos publicados más importantes, destacados o más actualizados.
- Controlar el estado de cada uno de los documentos generados: Consiste en visualizar el registro de movimientos creados por el moderador para un determinado documento del directorio.
- Aprobar o Rechazar los documentos creados por otros usuarios: Un moderador pueda generar movimientos a un documento creado en el directorio, de dos tipos: Aprobación y Rechazo. Cuando el movimiento sea una aprobación, debe especificarse la fecha en que será publicado el documento. Cuando sea un rechazo, es necesario indicar la razón de rechazo del documento. Estos movimientos pueden ser consultados por el propietario del documento a través del 'Administrador del Documentos' desarrollado en la Iteración N° 2 o por el moderador.

Para comenzar el proceso de desarrollo, es necesario agrupar todas las funcionalidades descritas anteriormente en un EJB de Sesión que se llamara ModerarDirectorio, su estructura se puede apreciar en la figura 43.

Figura 43. Diagrama de Clases del EJB de Sesión ModeradorDirectorio.
Fuente: El autor de la investigación

Para el desarrollo de las interfaces de usuario es necesario agregar nuevas clases a los componentes ‘Vista’ y ‘Controlador’.

En el caso del componente ‘Controlador’ se comienza por agregar una nueva clase, ObeliscoModerar subclase de ObeliscoConsultar, que encapsula la conexión al EJB de Sesión ModerarDirectorio. A partir de ella se crean un nuevo conjunto de clases que manejaran las entradas de usuario referentes al proceso de moderado de directorios.

Las subclases de ModerarDirectorio se apoyarán en un nuevo conjunto de paginas JSP creadas en el componente 'Vista', a través de las cuales se desplegará la información necesaria y las interfaces de usuario solicitadas. El diagrama de la Figura 44 muestra las nuevas clases dentro del componente 'Controlador'.

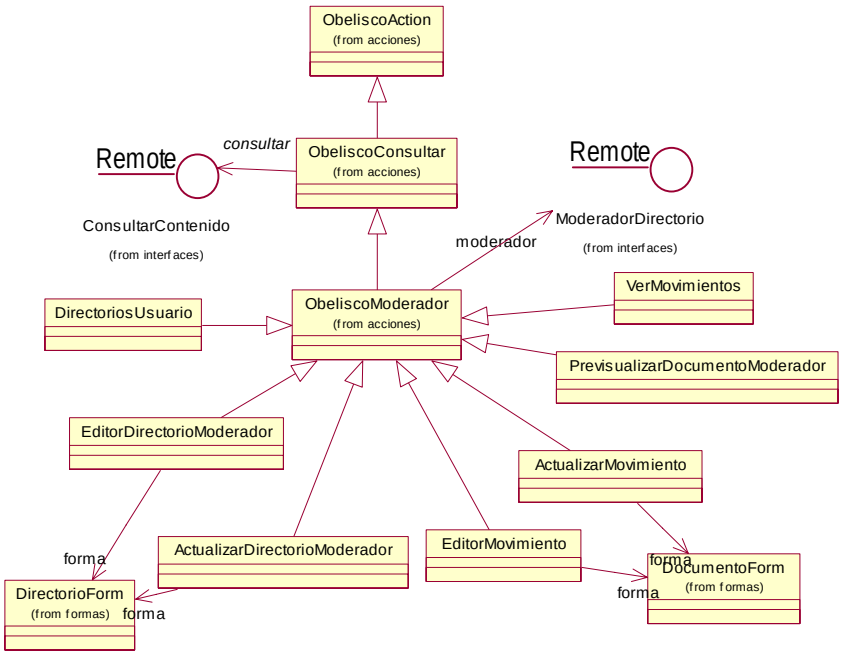

Figura 44. Diagrama de Clases del Componente 'Controlador', Iteración N° 3.
Fuente: El autor de la investigación

El diagrama de clases de la Figura 45 muestra la forma como se relacionan las nuevas clases del componente 'Controlador' con las nuevas páginas del componente 'Vista'.

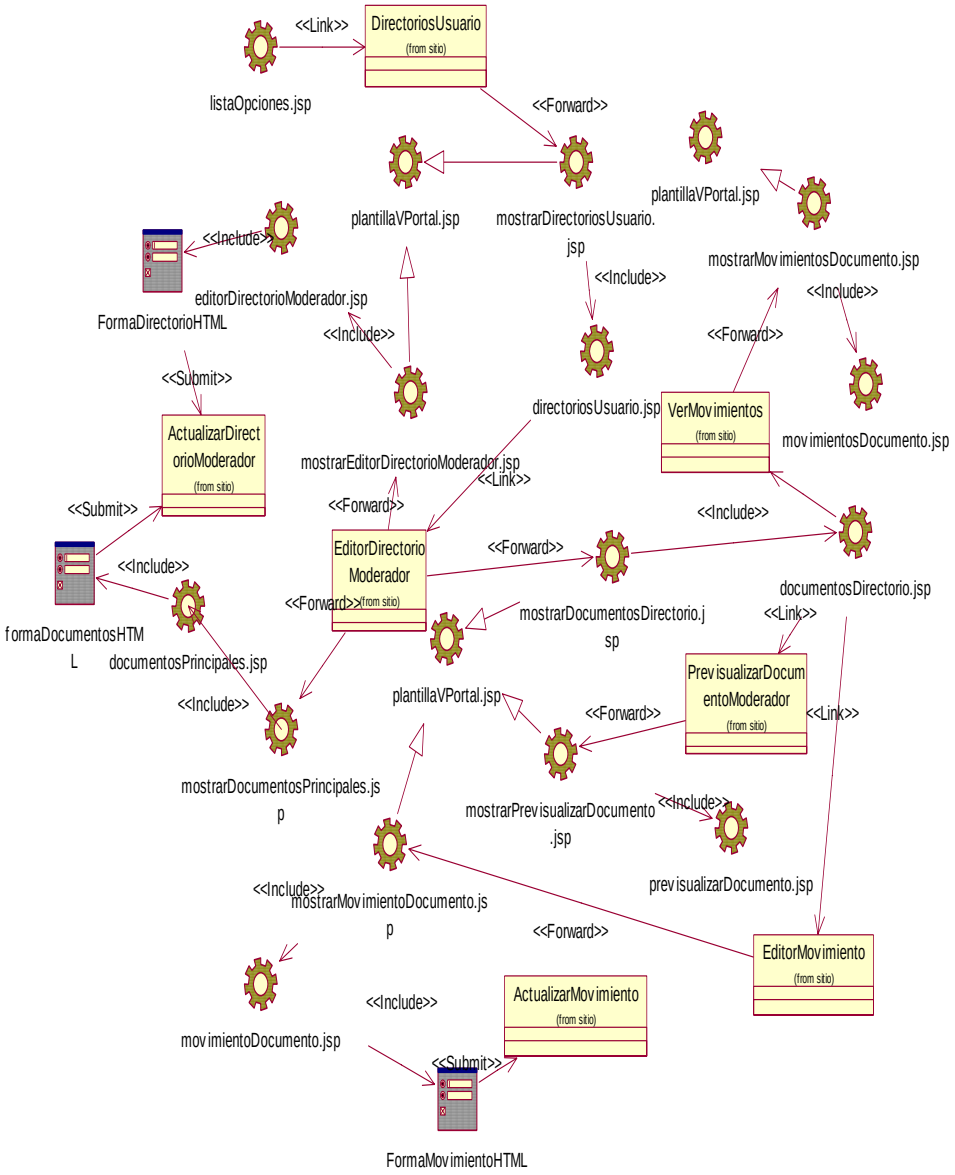

Figura 45. Diagrama de Clases del Componente 'Vista', Iteración N° 3.
Fuente: El autor de la investigación.

La figura 46 muestra la que forma como interactúan los distintos componentes del sistema para generar un movimiento a un documento.

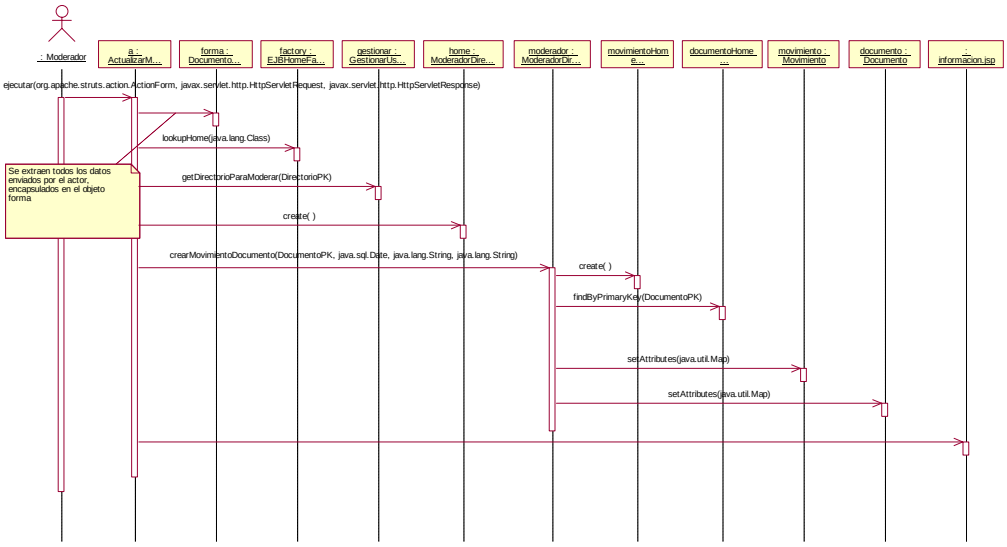

Figura 46. Diagrama de Interacción del proceso Crear Movimiento.
Fuente: El autor de la investigación.

Desarrollo

En esta iteración se agrega la opción 'Directorios' al menú de opciones del sistema, Figura 47.

Figura 47. Menú de opciones de OBELISCO, Iteración N° 3.
Fuente: El autor de la investigación.

Al hacer click en la nueva opción, se muestra el 'Moderador de Directorios', el cual despliega la lista de directorios que puede administrar el usuario actual. A través de esta herramienta, se puede ejecutar: 1) El 'Administrador de Seguridad' con el cual el moderador define los miembros de los grupos de usuarios del directorio. 2) El

'Administrador de Documentos Principales' que permite establecer los principales documentos del directorio seleccionado. 3) El 'Moderador de Documentos' que sirve para ver el estado de todos los documentos y generar movimientos de aprobación o rechazo a éstos. En la Figura 48 se muestra el 'Moderador de Directorios'.

Figura 48. Moderador de Directorios.
Fuente: El autor de la investigación.

La Figura 49 presenta el 'Administrador de Seguridad'. Allí se aprecia la lista usuarios del sistema, que pueden ser asignados a los grupos de usuarios de lectura y escritura y parámetros de importancia para el directorio, como el tiempo de vida de un documento y requerimientos de aprobación obligatoria para su publicación.

Figura 49. Administrador de Seguridad.
Fuente: El autor de la investigación.

El 'Moderador de Documentos', Figura 50, muestra los documentos publicados, rechazados y aquellos que esperan por la supervisión del moderador de directorio para su aprobación o rechazo. La interfaz permite visualizar para cada uno de los documentos del directorio y todos los movimientos que se han generado.

Desde cualquiera de los documentos que estén en estado 'Por Aprobación', se puede activar el 'Editor de Movimientos', que permite crear movimientos de Aprobación o Rechazo.

Figura 50. Moderador de Documentos.
Fuente: El autor de la investigación.

La Figura 51 muestra el 'Editor de Movimientos'. Allí se aprecia que se puede especificar el tipo de movimiento, la fecha efectiva del movimiento y una observación que es muy útil a la hora de aprobar o rechazar un documento.

Figura 51. Editor de Movimientos.
Fuente: El autor de la investigación.

El 'Visor de Movimientos', Figura 52, es una herramienta que permite consultar todos los movimientos que ha sufrido un determinado documento. Entre los detalles que se aprecian están el tipo de movimiento, las observaciones que se dieron para ese movimiento y la fecha en que se creó.

Figura 52. Visor de Movimientos.
Fuente: El autor de la investigación.

El ‘Administrador de Documentos Principales’ como se aprecia en la Figura 53 es un herramienta con la que el moderador puede establecer los documentos publicados más importantes dentro del directorio.

Figura 53. Administrador de Documentos Principales.
Fuente: El autor de la investigación.

CAPITULO V

CONCLUSIONES Y RECOMENDACIONES

Al desarrollar el presente trabajo de investigación, el autor puede establecer las siguientes conclusiones:

- El empleo de una metodología de desarrollo ágil, en conjunto con técnicas y métodos de Ingeniería de Software, permitió la construcción de una herramienta de gestión de contenido y documentos, que puede ser aplicada al proceso enseñanza-aprendizaje en el Decanato de Ciencias y Tecnología de la UCLA.
- La aplicación de entrevistas a profesores y personal administrativo en el Decanato de Ciencias y Tecnología y las consultas bibliográficas en Internet, ayudaron a detectar las necesidades y requerimientos relacionados al proceso de distribución de contenidos y documentos en el Decanato de Ciencias y Tecnología y sirvió de base para la caracterización de OBELISCO.
- El desarrollo de una arquitectura de software para OBELISCO, fue importante para ofrecer al desarrollador una visión completa del sistema, esta vista fue completada con el empleo de Lenguaje de Patrones.
- El uso de un patrón arquitectural MVC para el desarrollo de OBELISCO, permitió dividir la aplicación en tres componentes funcionales, esto hace posible que las reglas de negocio y las interfaces de usuario estén separadas, con lo que el sistema se hace más mantenible.
- MVC permitió separar las tareas y roles de desarrollo.
- El patrón arquitectural fue MVC crucial para la selección de las herramientas de desarrollo, en el caso de los componentes ‘Vista’ y ‘Controlador’ se

empleo el Framework STRUTS y para el componente 'Modelo' la tecnología EJB.

- El Framework STRUTS demostró ser una herramienta fácil de usar y con numerosos recursos para controlar las entradas de usuarios y presentar los datos provenientes del componente 'Modelo'.
- El uso de distintos patrones de diseño permitieron optimizar el funcionamiento del componente 'Modelo'.
- El uso de una metodología de Software constituye un elemento importante para llevar de manera organizada, las actividades con las que se puede conseguir productos de software de calidad y eficiencia.
- La metodología de desarrollo 'eXtreme Programming' (XP), es una metodología que centra sus esfuerzos en la satisfacción del cliente y en generar productos de software de manera rapida y con la mínima cantidad de artefactos de diseño.
- El desarrollo iterativo propuesto por XP, es una estrategia efectiva para incorporar de manera incremental las funcionalidades al sistema, así es mas fácil responder a nuevos requerimientos y cambios.
- A pesar de que la metodología XP recomienda el uso de Historias de Usuario y CRC para el diseño, se emplearon artefactos de UML como Diagramas de Casos de Uso y Diagrama de Clase, que facilitaron la percepción arquitectural del sistema.
- La plataforma de implementación de OBELISCO es económica, dado que su desarrollo esta basado en herramientas de fuentes abiertas.
- Las herramientas tecnológicas empleadas para la construcción de OBELISCO, permiten su instalación y funcionamiento en diversas plataformas de sistemas operativos y hardware.
- OBELISCO plantea un flujo de trabajo sencillo y controlable, para el contenido generado por los distintos usuarios del sistema.

- Con OBELISCO es posible dar datos descriptivos al contenidos y documentos generados en el sistema, con los que es posible establecer distintos mecanismos de búsqueda y organización.
- El sistema OBELISCO constituye un aporte importante, para el desarrollo del proceso enseñanza-aprendizaje en el decanato de ciencias y tecnología, ya que facilita el intercambio de información y documentación entre los miembros de la comunidad universitaria.
- OBELISCO representa el medio para consolidar y estandarizar el contenido producido en las actividades de investigación y docencia que se llevan a cabo dentro del decanato.

Recomendaciones

Tomando en cuenta las conclusiones planteadas, el autor considera conveniente:

- Implantar el software OBELISCO en el Decanato de Ciencias y Tecnología, dado que este cuenta con personal técnico especializado y equipos, para administrar y monitorear un software de este tipo.
- Usar OBELISCO para el intercambio de información de índole administrativo, ya que es lo suficientemente flexible para adaptarlo a la estructura organizativa del Decanato.
- Extender el uso de software, tanto para el ámbito académico como administrativo de la UCLA, esto permitirá crear un repositorio único de documentos, en el que se pueden implementar procesos de búsqueda de información rápidos y sencillos.

REFERENCIAS BIBLIOGRAFICAS

Arráez, F (1999) Gestión del Conocimiento. *Grupo de Estudios Prospectivos Sociedad, Economia y Ambiente* [Internet] <http://personales.com/venezuela/gepsea>

Balestrini, M (2002). *Como se Elabora un Proyecto de Investigación*. Caracas, Venezuela. BL Consultores Asociados. Servicio Editorial.

Barnes, S (2002) *Sistemas de Gestión del Conocimiento*. Madrid, España. Thomson Editores Spain Paraninfo, S.A.

Beck, K & Fowler, M (2000) *Planning Extreme Programming.* Addison Wesley

Booch , Rumbaugh & Jacobson (1999) *El lenguaje Unificado de Modelado*. España. Addison Wesley. 1999.

Buschmann, F. (1996). *Pattern Oriented Software Architecture, Volume 1: A System of Patterns*. Willey & Sons.

Canós, Letelier y Penadé. (2002). *Métodologías Ágiles en el Desarrollo de Software*. Valencia . España. Universidad Politécnica de Valencia

Cavaness, C. (2002). *Programming Jakarta Struts*. Sebastopol, CA, USA. O'Reilly & Associates, Inc. 2002.

Choo, C. W. (1998) *The Knowing Organization*. Oxford University Press.

Clements, P. (1996). *A Survey of Architecture Description Languages*. Alemania.

Coltell, O & Arregui, M (2004) TUTORIAL SOBRE UML Y PROCESO UNIFICADO EN INFORMÁTICA. *Sociedad Española de Informática de la Salud* [Internet] Agosto, 2005
< http://www.seis.es/inforsalud04/Inforsalud04_Tutorial_OColtell.pdf >

Cornella, A. (2002) La- Información no es necesariamente Conocimiento. *La Empresa de la Información* [Internet] <http://www.infonomia.com>

Davenport, De Long & Beers (1998). *What is a Knowledge Management Projects**?***

Davenport, T & Prusak, L (1999) *Working Knowledge: How Organizations Manage What They Know*. Harvard Business School Press.

Drucker, P (1993). *Data Mining: Creating Knowledge From Data?.*

Howard, J (2003) When is Web Content Management Right for an Intranet?. *Intranet Journal* [Internet] Diciembre, 2003 <http://www.intranetjournal.com/articles/200312/pij_12_17_03a.html>

Kenneth, K & Laudon, J. (1996). *Management Information Systems: Organization and Technology, 4th edition*. Prentice Hall, Inc.

Lee, A (1992). Workshop on two techniques for qualitative data analysis: Actino Research and Ethography. *The Thirteen International Conference on Information Systems*.

Lehaney, B. (2003) *Beyond Knowledge Management*. Hershey, PA, USA. Idea Group Publishing.

Leming, R (2002) Planning your first Knowledge Management Solutions. *Knowledge Board* [Internet] Febrero, 2005. <http://www.knowledgeboard.com/>

Nonaka, I & Takeuchi (1995) *The Knowledge-Creating Company: How Japanese Companies Create the Dynamics of Innovation*, Oxford University Press.

Palo, J (2003) JavaBeans Enterprise: Una introducción. *Java en Casterllano* [Internet] Agosto, 2005
< http://www.programacion.net/java/tutorial/javabeans/2 >

Pressman, R. (2001). *Ingeniería del Software. Un enfoque práctico*. McGraw-Hill.

Rojas, D (2000) *La Gestión del Conocimiento y los Sistemas de Información*. Barquisimeto, Venezuela. Universidad Centroccidental "Lisandro Alvarado".

Robertson, J (2002) What are the goals of a content management system?. *KM Column* [Internet] Agosto, 2002 <http://www.steptwo.com.au>

Roger, C & Kirriemuir, J (2003) Developing a Content Management System-based Web Site *D-Lib Magazine* [Internet] Mayo, 18. <http://www.dlib.org/dlib/may03/kirriemuir/05kirriemuir.html>

Ruggles, R. (1997). *Knowledge Tools: Using Technology to Manage Knowledge Better*. Center for Business Innovation. Ernst & Young.

UCLA (2002). *Manual para la Presentación del Trabajo Conducente al Grado Académico de: Especialización, Maestría, Doctorado*. Barquisimeto. Venezuela

Welicki, L (2005) Patrones y Antipatrones: Una Introducción?. *Microsoft Latinoamerica* [Internet] Agosto, 2005
< http://www.microsoft.com/spanish/msdn/comunidad/mtj.net/voices/mtj_2864.asp>

Zambrano, N & Acosta, A. (2001). *Patrones en Interacción Humano-Computador: Fundamentos Teóricos*. Caracas. Venezuela. Universidad Central de Venezuela. Facultad de Ciencias. Escuela de Computación.

Zhang, R & Chen J. (1997). *An Intranet Architecture to Support Organizational Learning*. Hershey, PA, USA. Idea Group Publishing.

ANEXOS

ANEXO 1

Cuestionario N° 1 Entrevista semiestructurada

UNIVERSIDAD CENTROCCIDENTAL "LISANDRO ALVARADO"
DECANATO DE CIENCIAS Y TECNOLOGIA
POSTGRADO EN CIENCIAS DE LA COMPUTACION
MENCION INGENIERÍA DEL SOFTWARE

DESARROLLO DE UNA HERRAMIENTA PARA LA GESTIÓN DE DOCUMENTOS y CONTENIDOS EN EL PROCESO ENSEÑANZA APRENDIZAJE DENTRO DEL DECANATO DE CIENCIAS Y TECNOLOGIA DE LA UCLA

ENTREVISTA ESTRUCTURADA

Nombre:	
Cargo:	
Departamento:	

1. ¿Que atributos considera importante asignar a un documento que se emplea en el proceso enseñanza-aprendizaje, para identificarlo, clasificarlo y ubicarlo?

2. ¿Qué tipos de documentos utiliza, para apoyar el desarrollo del proceso enseñanza-aprendizaje?

3. ¿Qué mecanismos emplea para distribuir el material instruccional a sus alumnos? ¿Considera que el mecanismo es rápido? ¿Este le garantiza la disponibilidad del material las 24 horas del día? ¿Es fácil y rápido ubicar el material instruccional generado por usted?

4. ¿En que proporción la facilidad de acceso al material instruccional, afecta el proceso enseñanza-aprendizaje?

5. ¿En que medida la disponibilidad de su material instruccional, contribuye al proceso enseñanza-aprendizaje?

6. ¿Qué impacto tendría el uso de un sistema de gestión de contenidos, en el proceso enseñanza-aprendizaje?

7. ¿Qué características desearía de un sistema de gestión de contenidos, que se aplique al proceso enseñanza-aprendizaje?

8. ¿Qué beneficios se producirán, como resultado de implementar un sistema de gestión de contenidos y documentos, en el proceso enseñanza-aprendizaje dentro del Decanato de Ciencias y Tecnología?

9. ¿Recomendaría alguna plataforma tecnológica en la cual se pueda implementar un sistema de gestión de contenidos?

ANEXO 2

Cuestionario N° 2 Entrevista semiestructurada

UNIVERSIDAD CENTROCCIDENTAL "LISANDRO ALVARADO"
DECANATO DE CIENCIAS Y TECNOLOGIA
POSTGRADO EN CIENCIAS DE LA COMPUTACION
MENCION INGENIERÍA DEL SOFTWARE

DESARROLLO DE UNA HERRAMIENTA PARA LA GESTIÓN DE DOCUMENTOS y CONTENIDOS EN EL PROCESO ENSEÑANZA APRENDIZAJE DENTRO DEL DECANATO DE CIENCIAS Y TECNOLOGIA DE LA UCLA

ENTREVISTA ESTRUCTURADA

Nombre:	
Cargo:	
Departamento:	

1. ¿Recomendaría alguna plataforma tecnológica en la cual se pueda implementar un sistema de gestión de contenidos?

2. ¿Cuenta el centro de computación con equipos que sirvan para alojar un sistema de gestión de contenidos y documentos?

3. ¿Existe personal técnico especializado dentro del centro del centro de computación, que pueda dar soporte a un sistema de gestión de documentos de contenidos basado en tecnología Web y sistema operativo Linux?

4. ¿Existe en el Decanato de Ciencias y Tecnología la infraestructura de red necesaria para garantizar el acceso y la disponibilidad, de los documentos depositados en un sistema de gestión de contenidos y documentos?

ANEXO 3

Carta de Solicitud de Validación a los Expertos

UNIVERSIDAD CENTROCCIDENTAL "LISANDRO ALVARADO"
DECANATO DE CIENCIAS Y TECNOLOGÍA

Barquisimeto, Julio de 2005

Ciudadano (a):_____

Reciba un cordial saludo en mi nombre. Sirva la presente para solicitar ante Ud. su colaboración en calidad de experto para determinar la validez de contenido de los cuestionarios con los cuales se pretende recabar información necesaria para el Trabajo de Grado titulado "Desarrollo de una herramienta para la gestión de documentos y contenidos en el proceso enseñanza aprendizaje dentro del Decanato de Ciencias y Tecnología de la UCLA", el cual será aplicado a docentes activos adscritos al Departamento de Sistemas y al Personal que labora en el Centro de Computación del Decanato de Ciencias y Tecnología de la UCLA.

Para tal propósito se anexa a la presente: 1) Matriz de Operacionalización de las Variables; 2) Formatos de Validación con sus respectivas instrucciones; 3) Cuestionarios que se aplicarán.

Su opinión al respecto representa un gran aval para esta investigación, dada la excelente labor docente, de investigación y extensión que Ud. ha realizado en pro de la formación profesional en la Universidad.

Sin otro particular al cual hacer referencia y agradeciendo de antemano su colaboración, quedo de Ud.

Atentamente,

_____
Prof. Ramón Antonio Valera Aranguren.
C.I.: 12.245.419

ANEXO 4

Formato de Validación de los Instrumentos

UNIVERSIDAD CENTROCCIDENTAL "LISANDRO ALVARADO"
DECANATO DE CIENCIAS Y TECNOLOGIA
POSTGRADO EN CIENCIAS DE LA COMPUTACION
MENCION INGENIERÍA DEL SOFTWARE

DESARROLLO DE UNA HERRAMIENTA PARA LA GESTIÓN DE DOCUMENTOS y CONTENIDOS EN EL PROCESO ENSEÑANZA APRENDIZAJE DENTRO DEL DECANATO DE CIENCIAS Y TECNOLOGIA DE LA UCLA

Formato de Validación de los Instrumentos

Ciudadano (a): _____

Para efectos de la evaluación correspondiente a los ítems planteados se determinará la validez de cada instrumento en los siguientes términos:

Se tomarán en cuenta los siguientes aspectos: a) **Pertinencia:** Es la correspondencia del ítem con el aspecto; b) **Claridad:** se refiere a la redacción precisa y sencilla del ítems; y c) **Congruencia:** entendida como la lógica interna del ítem.

Se le agradece seleccionar una de las 2 posibles opciones (Si/No) para cada ítems con el objetivo de señalar el grado de pertinencia, claridad y congruencia de los ítems.

Cuestionario N° 1

	Pertinencia		Claridad		Congruencia		
Ítem	SI	NO	SI	NO	SI	NO	OBSERVACIONES
1	☐	☐	☐	☐	☐	☐	_____
2	☐	☐	☐	☐	☐	☐	_____
3	☐	☐	☐	☐	☐	☐	_____
4	☐	☐	☐	☐	☐	☐	_____
5	☐	☐	☐	☐	☐	☐	_____
6	☐	☐	☐	☐	☐	☐	_____

7	☐	☐	☐	☐	☐	☐	
8	☐	☐	☐	☐	☐	☐	
9	☐	☐	☐	☐	☐	☐	

Cuestionario N° 2

	Pertinencia		Claridad		Congruencia		
Ítem	SI	NO	SI	NO	SI	NO	OBSERVACIONES
1	☐	☐	☐	☐	☐	☐	
2	☐	☐	☐	☐	☐	☐	
3	☐	☐	☐	☐	☐	☐	
4	☐	☐	☐	☐	☐	☐	

ANEXO 5

Descripciones textuales de los casos de uso del sistema de gestión de contenidos y documentos OBELISCO

Caso de Uso: Mostrar Directorio.
Referencia: 1
Precondición: Usuario Validado

Acción del Actor	**Respuesta del Sistema.**
1. 1. El usuario decide consultar un directorio y proporciona su identificador.	
	1.1. El sistema determina si tiene el permiso sobre el directorio. 1.2. El sistema consulta los documentos publicados y no vencidos, y los clasifica en principales y secundarios. 1.3. El sistema presenta los documentos del directorio en una interfaz que permita consultarlos.

Caso de Uso: Mostrar Documento.
Referencia: 2
Precondición: Usuario Validado

Acción del Actor	**Respuesta del Sistema.**
1. El usuario decide consultar un documento y proporciona su identificador.	
	1.1. El sistema determina si tiene el permiso sobre el directorio al que pertenece el documento solicitado y si su estado es publicado. 1.2. El sistema determina los datos generales del documento, así como, su presentación y lista de recursos. 1.3. El sistema presenta el documento en una interfaz de tal manera que permita descargar lo recursos que

	posee.

Caso de Uso: Validar Usuario.
Referencia: 3

Acción del Actor	**Respuesta del Sistema.**
1. El usuario proporciona sus credenciales al sistema.	
	1.1. El sistema valida las credenciales, determinando la lista de directorios que puede consultar, así como los niveles de acceso que tiene sobre ellos. 1.2. El sistema presenta la lista de directorios de tal manera que pueda consultarlos.

Caso de Uso: Crear Documento.
Referencia: 4
Precondición: Usuario Validado, El usuario tiene permiso de escritura en algún directorio del sistema

Acción del Actor	**Respuesta del Sistema.**
1. El usuario decide crear un nuevo documento y opcionalmente modificar los datos y componentes de un documento, esto es su presentación y lista de recursos.	
	1.1. Presentar una interfaz que permita la creación y edición del documento. Ver Casos de uso: Cargar Documento (Ref. 4.1.) Visualizar Documento (Ref. 4.2.) Editar Documento (Ref. 4.3.)

Caso de Uso: Cargar Documento
Referencia: 4.1

Acción del Actor	**Respuesta del Sistema**
1. El usuario selecciona la opción abrir documento.	
	2. Presenta la interfaz que permite seleccionar alguno de los documentos creados por el usuario actual.
3. El usuario selecciona ó indica el	

documento que quiere abrir.	4. El sistema determina los datos del documento, así como sus componentes, esto es, la presentación, lista de recursos y lista de movimientos. 5. Presenta la interfaz de edición del documento.

Caso de Uso: Visualizar Documento
Referencia: 4.2
Precondición: Existe el documento a visualizar.

Acción del Actor	Respuesta del Sistema
1. El usuario desea conocer la apariencia final de un documento de su propiedad.	2. Presenta la interfaz que muestra la apariencia final del documento del usuario.

Caso de Uso: Editar Documento.
Referencia: 4.3
Precondición: Existe el documento a editar.

Acción del Actor	Respuesta del Sistema.
1.1. El usuario selecciona editar o consultar uno de los componentes del documento:	1.1.1. En el caso de que el componente sea: Encabezado: ejecutar editar encabezado. (Ref: 4.3.1.). Presentación: ejecutar editar presentación. (Ref: 4.3.2.). Movimientos: ejecutar consultar movimientos. (Ref: 4.3.3.). Cuerpo: ejecutar editar cuerpo. (Ref: 4.3.4.). Recursos: ejecutar editar recursos. (Ref: 4.3.5.).
1.2. El usuario decide guardar el documento.	1.2.1. El sistema almacena el documento y todos sus componentes. En el caso en que el directorio sea de aprobación

	obligatoria de documentos el estado del documento será 'Por Aprobar' en caso contrario pasara al estado 'Publicado'

Caso de Uso: Editar Encabezado.
Referencia: 4.3.1
Precondición: Existe el documento a editar.

Acción del Actor	**Respuesta del Sistema.**
	1. Muestra la interfaz de edición del encabezado del documento, en ella coloca la lista de directorios donde el usuario actual, tiene permiso de escritura, así como los datos del encabezado del documento seleccionado.
2. El usuario proporciona los datos del encabezado del documento, esto es, titulo, directorio donde se va a guardar, fuente, palabras claves y resumen.	
3. El usuario confirma los datos del encabezado.	
	4. Valida si el usuario proporciono un directorio donde tiene permiso de escritura. 5. Determina si el usuario proporciono todos los datos del encabezado del documento.

Caso de Uso: Editar Presentación.
Referencia: 4.3.2
Precondición: Existe el documento a editar.

Acción del Actor	**Respuesta del Sistema.**
	1. Muestra la interfaz de edición de presentación del documento, en ella coloca la lista de recursos de tipo imagen que posee el documento a editar.
2. El usuario proporciona los datos de la presentación del documento,	

esto es, la imagen de presentación, la alineación de la imagen y su alto ancho y alto.	
3. El usuario confirma los datos de la presentación.	
	4. Valida si el usuario proporciono un recurso que pertenece al documento ha ser editado. 5. Determina si el usuario proporciono todos los datos de la presentación del documento.

Caso de Uso: Consultar Movimientos.
Referencia: 4.3.3
Precondición: Existe el documento a editar.

Acción del Actor	**Respuesta del Sistema.**
	1. Muestra la interfaz con todos los movimientos que se han generado para el documento seleccionado.

Caso de Uso: Editar Cuerpo.
Referencia: 4.3.
Precondición: Existe el documento a editar.

Acción del Actor	**Respuesta del Sistema.**
	1. Muestra la interfaz de edición del cuerpo del documento, esto el texto completo del documento.
2. El usuario proporciona los datos del texto completo del documento	
3. El usuario confirma los datos del cuerpo.	
	4. Determina si el usuario proporciono todos los datos del cuerpo del documento.

Caso de Uso: Editar Recursos.
Referencia: 4.3.5
Precondición: Existe el documento a editar.

Acción del Actor	Respuesta del Sistema.
	1. Muestra la interfaz con todos los recursos que tiene asignado el documento actual.
2. El usuario selecciona crear o eliminar alguno de los recursos del documento.	
	3. En el caso de que la operación sea crear ejecutar crear recurso. (Ref: 4.3.5.1).

Caso de Uso: Crear Recurso.
Referencia: 4.3.5.1
Precondición: Existe el documento a editar.

Acción del Actor	Respuesta del Sistema.
	1. Muestra la interfaz para agregar el recurso al documento.
2. El usuario selecciona el archivo que quiere adjuntar al documento, y proporciona una descripción de este.	
3. El usuario confirma los datos del recurso.	
	4. Determina si el usuario proporciono todos los datos del recurso.

Caso de Uso: Moderar Directorio.
Referencia: 5
Precondición: Usuario Validado, El usuario tiene permiso de moderar en algún directorio del sistema

Acción del Actor	Respuesta del Sistema.
1. El usuario decide modificar los datos generales y de seguridad de un directorio, opcionalmente modificar sus documentos principales, así como alterar el estado de cada uno de los documentos depositados en el.	
	1.1. Presentar una interfaz que permita modificar los datos generales y de

	seguridad de un directorio, los documentos principales y el estado de cada uno de los documentos depositados en el. Ver Casos de uso: Definir Seguridad (Ref. 5.1.) Definir Documentos Principales (Ref. 5.2.) Administrar Documentos (Ref. 5.3.)

Caso de Uso: Definir Seguridad.
Referencia: 5.1
Precondición: Existe el directorio, El usuario actual es moderador del directorio.

Acción del Actor	**Respuesta del Sistema.**
	1. Muestra la interfaz de edición de seguridad del directorio, mostrándose los usuarios que tienen permiso de escritura y lectura en el, así como los datos que puede modificar su moderador.
2. El usuario proporciona los usuarios que pueden escribir y leer en el directorio, tiempo de vencimiento de los documentos y si el directorio tiene de aprobación de documentos.	
3. El usuario confirma los datos del directorio.	
	4. Valida si los usuarios proporcionados existen en el sistema. 5. Determina si el usuario proporciono todos los datos modificables del directorio, esto es, tiempo de vencimiento de los documentos y tipo de aprobación de documentos.

Caso de Uso: Definir Documentos Principales.
Referencia: 5.2
Precondición: Existe el directorio, El usuario actual es moderador del directorio.

Acción del Actor	Respuesta del Sistema.
	1. Muestra la interfaz con todos los documentos en estado 'Publicado', mostrándose cuales de ellos son principales y cuales no.
2. El usuario determina de la lista de documentos cuales son los principales.	
3. El usuario confirma los documentos principales del directorio.	
	4. Determina si el usuario proporciono por lo menos un documento principal al directorio seleccionado.

Caso de Uso: Administrar Documentos.
Referencia: 5.3
Precondición: Existe el directorio, El usuario actual es moderador del directorio.

Acción del Actor	Respuesta del Sistema.
	1. Muestra la interfaz con todos los documentos, agrupándolos por su estado actual, 'Por Aprobación', 'Publicados' y 'Rechazados'.
2. El usuario decide visualizar los movimientos de un determinado documento, opcionalmente puede previsualizar de que manera luce cuando un usuario lo consulte, y también puede crearle movimientos en el caso en que se encuentre en estado 'Por Aprobación'.	
	3. Presentar una interfaz que permita visualizar los movimientos de un determinado documento, previsualizar su apariencia ante

	cualquier usuario del sistema y crear movimientos de rechazo o aprobación. Ver Casos de uso: Crear Movimiento. (Ref. 5.3.1.) Consultar Movimientos (Ref. 4.3.3) Visualizar Documento (Ref. 4.2)

Caso de Uso: Crear Movimiento.
Referencia: 5.3.1
Precondición: Existe el documento, Existe el directorio, El usuario actual es moderador del directorio.

Acción del Actor	**Respuesta del Sistema.**
	1. Muestra la interfaz para crear un nuevo movimiento al documento seleccionado, se muestra los tipos de movimiento, 'Aprobación' y 'Rechazo'.
2. El usuario selecciona el tipo de movimiento, si el tipo es 'Rechazo', proporciona la razón por la que esta rechazando el documento y si es 'Aprobación' indica la fecha de publicación del documento.	
3. El usuario confirma los datos del movimiento.	
	4. Determina si el usuario proporciono todos los datos del movimiento. 5. Guarda el Movimiento. 6. Actualiza el estado del documento en función al tipo de movimiento.

A. Currículum Vitae del Autor

Ramón Antonio Valera Aranguren, Portador de la C.I: N° V-12.245.419, nace en Barquisimeto el 20 de Noviembre de 1975. Realiza estudios de Primaria en la Escuela Estadal "Julio T. Arze". Obtiene el título de bachiller en el Colegio "San Vicente de Paúl" de Barquisimeto. Ingresa a la Universidad Centroccidental "Lisandro Alvarado" de Barquisimeto en el año 1993, para iniciar estudios de pregrado en la carrera de Análisis de Sistema, egresando en el año 1996. En el año 1998 inicia estudios de pregrado en la carrera de Ingeniería en Informática de la Universidad Centroccidental "Lisandro Alvarado", de donde egresa en el año 2001. Paralelamente a sus actividades de estudio e Ingeniería en Informática, se desempeña en el cargo de Analista Programador de Sistemas en el Centro de Computación del Decanato de Ciencias, cargo que ejerce hasta el día de hoy. En el año 2003 se inicia en la carrera docente, desempeñándose en el cargo de Docente Tiempo Convencional adscrito al Departamento de Sistemas del Decanato de Ciencias y Tecnología de la Universidad Centroccidental "Lisandro Alvarado", donde se dedica actualmente en las áreas de base de datos y sistemas de información.