INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY

CAMPUS MONTERREY

PROGRAMA DE GRADUADOS EN MECATRÓNICA Y TECNOLOGÍAS DE INFORMACIÓN

# PROTOTIPO PARA LA ADMINISTRACIÓN DE CONOCIMIENTO ORIENTADO AL CONCEPTO WEB 2.0

**TESIS**

PRESENTADA COMO REQUISITO PARCIAL PARA OBTENER EL GRADO ACADÉMICO DE:

**MAESTRO EN CIENCIAS EN TECNOLOGÍA INFORMÁTICA**

POR:

**CARLOS EDUARDO BARRADAS SERNA**

MONTERREY, N.L. Noviembre 2007

INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY

PROGRAMA DE GRADUADOS EN MECATRÓNICA Y TECNOLOGÍAS DE INFORMACIÓN

Los miembros del comité de tesis recomendamos que la presente tesis del Lic. Carlos Eduardo Barradas Serna sea aceptada como requisito parcial para obtener el grado académico de Maestro en Ciencias en Tecnología Informática.

Comité de tesis:

_____

Guillermo Jiménez Pérez, PhD.
Asesor

_____

Jorge Alejandro Álvarez Bujanos, MTI.
Sinodal

_____

David Llanas García, MCT.
Sinodal

_____

Graciano Dieck Assad, PhD.
Director del Programa de Graduados en Mecatrónica
y Tecnologías de Información.
Noviembre de 2007

# PROTOTIPO PARA LA ADMINISTRACIÓN DE CONOCIMIENTO ORIENTADO AL CONCEPTO WEB 2.0

POR:

**CARLOS EDUARDO BARRADAS SERNA**

**TESIS**

Presentada al Programa de Graduados en Mecatrónica
y Tecnologías de Información

Este trabajo es requisito parcial para obtener el grado de

**Maestro en Ciencias en Tecnología Informática**

INSTITUTO TECNOLÓGICO Y DE ESTUDIOS
SUPERIORES DE MONTERREY

NOVIEMBRE 2007

## Dedicatorias

**A mis padres**
*Quienes son mi ejemplo de fortaleza y dedicación.*
*Por ellos he aprendido que siempre hay que trabajar*
*para alcanzar lo que anhelamos.*

**A Marisol**
*Porque este logro también es de ella.*
*Estuvo conmigo en todo momento y siempre me bridó su apoyo,*
*pero sobre todo, su inmenso amor y comprensión.*

**A toda mi familia**
*Porque siempre desean lo mejor para mí.*
*Espero ser para todos un ejemplo de voluntad y superación.*

## Agradecimientos

**Al Dr. Guillermo Jiménez**
*Quien con su asesoría y disposición,*
*me encaminó al logro del objetivo.*

**A mis sinodales**
*Por las aportaciones hechas para mejorar este trabajo,*
*además del tiempo invertido para ayudarme.*

**A todos los que me apoyaron de cualquier forma**
*Ya que sus palabras, acciones y buenos deseos,*
*me sirvieron para continuar trabajando.*

## Resumen

El conocimiento generado en las organizaciones es un recurso muy valioso para identificar las mejores prácticas productivas y ofrecer productos o servicios de calidad. Sin embargo, la clasificación de la información para que se convierta en utilizable es un problema que actualmente se sufre en todos los niveles de una empresa.

La administración de conocimiento pretende dar pauta para controlar y documentar de forma ordenada las acciones que se consideren relevantes en la organización; mientras que la tecnología de información ofrece las herramientas para llevar estas técnicas a sistemas de cómputo que permitan almacenar y tener la información accesible a sus usuarios.

Las tendencias recientes en el desarrollo de aplicaciones basadas en Web ofrecen nuevas ventajas en la reproducción de contenidos, lo que ahora se conoce como la generación de Web 2.0 y el término Rich Internet Application (Aplicación Rica de Internet) surge con fuerza para brindar soluciones de software del tipo aplicaciones de escritorio robustas.

La brecha tecnológica ahora es más reducida y es posible desarrollar software con herramientas de código libre (Open Source) que pretenden ganar terreno en comparación con las herramientas propietarias de desarrollo exclusivas para programadores expertos y que significa invertir importantes sumas de dinero para poder desarrollar en una plataforma específica.

Este trabajo de tesis por lo tanto, busca mediante las herramientas Open Source ofrecer una solución de software basada en las Rich Internet Applications, utilizando un framework de desarrollo de aplicaciones llamado OpenLaszlo. El resultado es un sistema de administración de conocimiento de nueva generación, que promueve a su vez el desarrollo de las iniciativas de código libre y la utilización de estándares de intercambio de datos en XML.

# Tabla de contenido

# Índice de figuras

# Capítulo 1. Introducción

## 1.1. Antecedentes

En la actualidad, podemos observar que uno de los principales recursos de las organizaciones no es la generación de capital, sino la generación de conocimiento. Gracias a éste se pueden alcanzar buenas prácticas en las empresas, ya que la identificación del conocimiento previo y experiencias vividas en el entorno laboral, ayudan a crear soluciones innovadoras [1].

De acuerdo con Garibaldi [1], día a día se generan en las empresas grandes cantidades de documentos que van desde reportes, estrategias, prototipos, prácticas productivas e información relevante para sus labores cotidianas, con ello se presenta de manera frecuente el problema de clasificar la información y su permanente mantenimiento para que siga siendo útil. Es aquí donde surge un nuevo concepto: la administración del conocimiento. Esto no es más que la transferencia efectiva del conocimiento a los actores apropiados de la empresa en el momento preciso, para que, reconociendo oportunidades, les sea posible tomar mejores decisiones, permitiendo poner el conocimiento en acción.

Según una investigación desarrollada en la Universidad de Minnesota [2], el rol del tecnólogo de información tiene una gran repercusión en la toma de decisiones y en la generación de nuevas alternativas para mejorar la productividad de las organizaciones, gracias al apoyo de sistemas de cómputo especializados como: sistemas de soporte a la decisión, sistemas expertos y sistemas de administración del conocimiento; éstos últimos se encargan de retener, transferir y mejorar el conocimiento de cualquier área que lo requiera para poder agregar valor a sus activos y encaminarse al logro de las metas planteadas.

El constante crecimiento de la tecnología de información hace cada vez más necesario el desarrollo de soluciones de software simples, amigables y sobre todo funcionales [3]. De ahí el surgimiento del Web 2.0 y el auge de los proyectos de código libre que permiten, entre otras cosas, la rápida mejora de aplicaciones (actualizaciones de código) sin que esto signifique un costo económico excesivo o una tarea exclusiva de programadores expertos, ya que por la naturaleza de estos proyectos, la documentación, acceso a los ambientes de desarrollo y soporte son gratuitos en su mayoría, permitiendo crear comunidades de programadores que comparten experiencias para promover y mantener sus iniciativas.

Gregory [4], por su parte, menciona conceptos como: procesos productivos, diseño de productos y cadena de suministro; los cuales son regularmente utilizados en el área de manufactura e implican entre otras cosas, el empleo de grandes cantidades de información y es por eso que los sistemas de gestión o administración del conocimiento están ganando terreno en esta área y aportan una ventaja

competitiva para quien los utiliza de forma adecuada, logrando con ello facilidad en la toma de decisiones, mejora de los productos y simplificación de los procesos de producción, entre otros beneficios.

## 1.2. Objetivo del proyecto de tesis

El presente trabajo de investigación se centra en el desarrollo de un sistema basado en Web orientado al término Rich Internet Application (Aplicación Rica de Internet), que permita a una compañía almacenar el conocimiento que genera en su interior, reuniendo información vigente y reutilizable que ayude a tomar decisiones oportunas, concentrando información relevante para la empresa en un solo lugar, facilitando la consulta en línea de registros multimedia (combinación de imágenes, texto, audio y video).

Este desarrollo pretende mostrar parte del funcionamiento de una aplicación para administrar conocimiento (alta / mantenimiento de información, control de usuarios y permisos en el sistema), aprovechando las bondades que ofrece el software libre. Por lo tanto, la solución propuesta se implementa tanto a nivel de código fuente como en base de datos, utilizando iniciativas de código libre (Open Source). Apoyados también del uso de estándares para acceso / entrega de datos (XML) y programando en una plataforma de desarrollo de nueva generación que permite ejecutar sus aplicaciones solo utilizando un navegador de Internet.

## 1.3. Alcance del proyecto de tesis

Existen sistemas de administración de conocimiento que se centran en la recopilación y clasificación de la información, sin embargo, su presentación e integración con otras herramientas no es la óptima, debido a que el usuario tiene la percepción de que estos sistemas no son útiles ni atractivos dada su complejidad; además, la mayoría de ellos se basa en software propietario.

Por lo tanto, lo que se persigue es ofrecer un sistema de administración de conocimiento que mediante una categorización general de la información, permita interactuar con él de manera sencilla y amigable para el usuario final, además de mostrar las bondades que ofrecen las Rich Internet Applications.

El sistema desarrollado se ha alimentado para su validación con información relevante acerca del proyecto RV-10 que el Centro de Innovación en Diseño y Tecnología del Tecnológico de Monterrey, Campus Monterrey desarrolla actualmente, con la finalidad de hacer una prueba de concepto que sustente las afirmaciones y el objetivo de este trabajo de tesis. Lo anterior se detalla en el capítulo 5.

Dicha prueba de concepto o caso de estudio pretende mostrar de forma práctica la utilización del sistema y despliegue de información, además de promover el desarrollo de software libre orientado a nuevas tecnologías basadas en Web.

## 1.4. Organización del documento de tesis

La estructura y presentación de este documento de tesis es la siguiente:

El capítulo 1 resume el problema de investigación e integra sus temas fundamentales (conocimiento, empresa y tecnología) para dar inicio con la propuesta de trabajo.

El capítulo 2 presenta antecedentes y define conceptos relacionados con el trabajo desarrollado, para clarificar el contexto respecto a la investigación realizada y el sistema diseñado.

El capítulo 3 destaca las características de la tecnología conocida como AJAX para posteriormente adentrarse en la arquitectura y modo de operación del framework OpenLaszlo que ofrece una suite de desarrollo para generar Rich Internet Applications.

El capítulo 4 define la arquitectura del sistema, diagramas de diseño de software, diccionario de datos y consideraciones en el desarrollo, detallando al mismo tiempo los módulos implementados en el sistema.

El capítulo 5 sintetiza los objetivos y alcances del caso de estudio en donde se implementó el sistema desarrollado mostrando sus pantallas y explicando los resultados obtenidos.

El capítulo 6 ofrece las conclusiones del trabajo de tesis, resume las experiencias aprendidas y destaca las mejoras posibles al sistema.

# Capítulo 2. Fundamento teórico

En este capítulo se abordan conceptos relacionados con la administración del conocimiento y su impacto en las organizaciones. A su vez, se comenta acerca de los sistemas de colaboración en el área de manufactura y algunas soluciones de software existentes.

Además, se muestra un enfoque de la tecnología de información y las nuevas tendencias en el desarrollo de aplicaciones para ofrecer soluciones innovadoras de software en general, aplicables inclusive en aquellas destinadas para administrar conocimiento.

## 2.1. Administración del conocimiento

De acuerdo con Hahn [2], la administración del conocimiento se puede definir como un proceso claro y sistémico para adquirir, ordenar y comunicar el conocimiento a los integrantes de la organización con el fin de aportar información que les pueda servir para mejorar su desempeño laboral. Es muy importante mencionar la diferencia entre datos, información y conocimiento. Los datos son una unidad de información, la información es la unión de varios datos por medio de una relación lógica y el conocimiento es un flujo de ideas, información y experiencia que al ser evaluadas y comprendidas, pueden generar otras ideas e incluso más conocimiento.

### 2.1.1. Perspectiva y retos

Garibaldi [1] plantea que el conocimiento puede ser caracterizado como tácito o explícito. Tácito es aquél que la persona solamente sabe y no es capaz de explicarlo o poderlo compartir con facilidad porque forma parte de la percepción de una persona (conocido como *know-how*). Por otro lado, el conocimiento explícito puede ser compartido y explicado sin mayores contratiempos para que los demás puedan ejecutarlo (conocido como *know-what*).

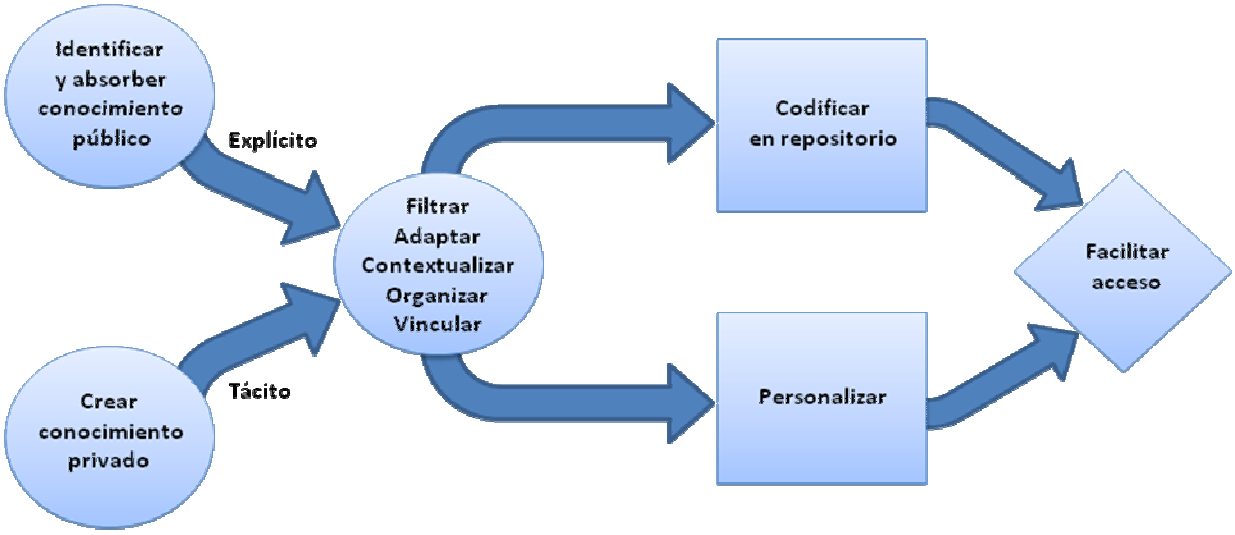


Figura 2.1
Proceso de administración del conocimiento
(Adaptado de [1])

La Figura 2.1 identifica las fuentes del conocimiento y el proceso general de clasificación que se debe aplicar para que sea posible colocarlo en un repositorio personalizable y de fácil acceso. Todo lo anterior depende del tipo de conocimiento de la organización, siendo necesario pasarlo por varios filtros que lo segmenten para facilitar su uso y distribución.

Garibaldi [1] también asegura que es aquí donde entran los sistemas de administración de conocimiento, en la clasificación del conocimiento explícito. Ya que permite almacenar grupos de ideas, experiencias vividas e incluso posibles soluciones a ciertas situaciones, con el fin de tener antecedente de lo ocurrido y saber actuar cuando se vuelva a presentar. Además, promueve la mejora continua y por ende, la calidad en el trabajo, resultando muy útil para las empresas en cuanto al logro de sus estrategias.

Según lo demuestra Hahn y Subramani en su investigación [2], uno de los retos para estos sistemas es balancear la sobrecarga de información y el contenido potencialmente útil. Esto significa que efectivamente hay que generar una conciencia de documentar lo que se realiza o se resuelve, siempre y cuando sea relevante y no repetitivo. Otro reto importante que aseguran dichos autores es que un sistema de administración de conocimiento no dará los resultados esperados si no se le alimenta información, es decir, hay que inculcarle al usuario que el sistema debe ser utilizado y debe verse como una herramienta de trabajo, para que se logre su aceptación y se promueva su uso.

### 2.1.2. Impacto en la actualidad

El tema de administración del conocimiento está llamando la atención de muchas áreas de investigación por las bondades que ofrece al ser implementada en las organizaciones [1]. Sin embargo, esto requiere de un alto compromiso del personal para compartir principalmente la información que manejan día con día y convertir poco a poco un sistema computacional dedicado a almacenar el conocimiento, en una herramienta muy poderosa y de gran valor para la organización.

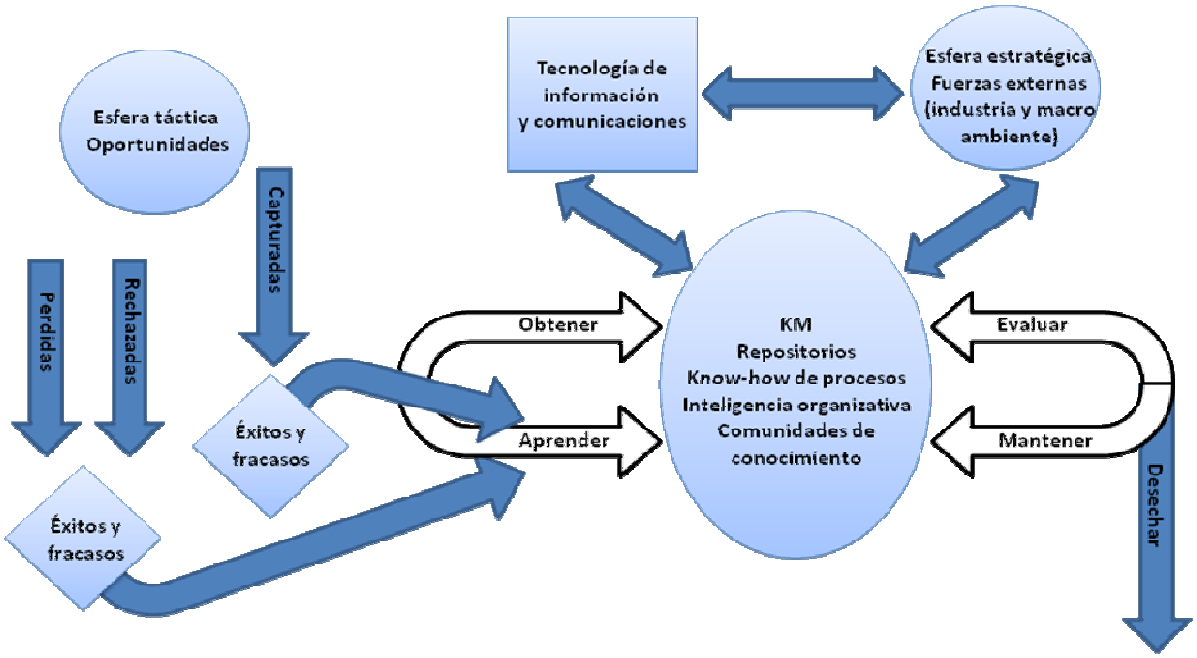


Figura 2.2
Interacción entre conocimiento, TI y estrategia
(Adaptado de [1])

La Figura 2.2 muestra una visión general de los diversos factores que involucran a la administración de conocimiento en las empresas: la importancia de las experiencias adquiridas (éxitos y fracasos), el uso de la tecnología de información para evidenciar las actividades desempeñadas y el detalle de puntos esenciales de los procesos (compartir estrategias) que conlleven a mantener el conocimiento y le permitan seguir siendo útil, pues en caso contrario, definitivamente habrá que desecharlo por obsolescencia o falta de valor agregado.

De acuerdo a los comentarios de Rowley [5], la administración del conocimiento no es más que concentrar las fuerzas de la tecnología en un solo lugar para ayudar a otros a aprender, adaptarse a la competencia y sobre todo sobrevivir. No se trata de recopilar información que nadie va a consultar, sino almacenarla con

el fin de tener una herramienta que contenga soluciones para ser adaptadas y aplicadas en el momento necesario.

El mismo autor asegura que la administración del conocimiento implica, entre otras cosas:

- Generar nuevo conocimiento.
- Acceder a él desde fuentes externas.
- Transferir conocimiento.
- Emplearlo para tomar decisiones.
- Incluirlo en los procesos y / o servicios.

Una de las funciones de la tecnología es proporcionar las herramientas adecuadas para dar un gran paso en la creación de productos innovadores y dar servicios de calidad [4]. Las redes colaborativas proveen los medios adecuados para compartir información entre audiencias especializadas o incluso distintas y con esto se aprovecha la función de la tecnología informática. Entonces, un sistema de administración del conocimiento no solo reúne ideas, sino también soluciones, por ejemplo, es posible que ayude en todo el proceso de cadena de suministro para resolver ciertos problemas o indicar estándares para no variar la calidad de los productos.

Es aquí en donde los sistemas computacionales toman fuerza para proveer soluciones a las empresas, relacionándose directamente con la administración del conocimiento. Con el desarrollo de aplicaciones orientadas hacia este tema, es posible integrar información en bases de datos en donde sus contenidos puedan consultarse en red (intranet o internet) de una forma sencilla y logrando la transferencia de conocimiento necesaria para mejorar los procesos productivos de una organización.

Un trabajo de investigación en la Universidad de Pittsburgh [6] detectó que la administración de conocimiento produce:

- Repositorios de conocimiento.
- Mejores prácticas y lecciones aprendidas.
- Redes expertas.
- Comunidades de prácticas.

Para cada uno de estos puntos se identificó que la administración de conocimiento forma parte de la visión estratégica de la empresa para tomar ventaja y dar soluciones certeras y adaptables. Al mismo tiempo ayuda a determinar si los procesos que se están llevando a cabo valen la pena, es decir, asegurarse que las cosas se hagan bien para poderlas tomar como base en casos similares.

Un detalle esencial de este tema detectado por la Universidad de Pittsburgh [6] es que por medio de un sistema puede capturarse de forma organizada el conocimiento que se maneja en la empresa para que por medio de herramientas tecnológicas y aplicaciones de sistemas se pueda construir una solución que al corto plazo es una inversión, y ayudará a la innovación, sustentabilidad, motivación, tareas y responsabilidades bien definidas, estandarización de operaciones, aseguramiento de calidad, entre otros.

Son muchas las ventajas que ofrece la administración del conocimiento, sin embargo hay que saberla aplicar [6]. La metodología existe como tal, pero habrá que hacer un análisis concienzudo del área donde desea implementarse para posteriormente identificar los requerimientos del sistema a desarrollar y realmente generar un producto de software que sea útil, tomando en cuenta las tecnologías disponibles.

Con base al planteamiento de Gregory [4], una red en la que cada participante utiliza el software más adecuado para realizar su trabajo siempre da mejores resultados, como entregas rápidas, bajos costos y mejora en la calidad. Todo esto puede ser logrado con sistemas que aporten inteligencia, es decir, sistemas de administración de conocimiento.

### 2.1.3. Sistemas de colaboración en manufactura

En el desarrollo de productos de manufactura, el software es fundamental para acceder a la información, transformarla y trabajar en equipo, al respecto de este tema se menciona que [4]:

- La colaboración es esencial para la transición hacia la adaptación en masa (mass customization).
- Hay que asegurarse que los CAD (Computer Aided Software o Sistemas Asistidos por Computadora) estén basados en Internet y sean centralizados.
- No es suficiente poner un navegador Web en las computadoras de los diseñadores.
- No confundir administración de transacciones a través de Internet con ingeniería colaborativa basada en Web.
- Una red con software adaptable y adecuado dará mejores resultados.

Como también lo demuestra Gregory [4], las diferentes herramientas utilizadas en las empresas de la industria manufacturera han hecho complicado mover información de un lado a otro, pues emplean sistemas propietarios o no estandarizados para compartir mejores prácticas o simplemente para intercambiar información. La tendencia es modelar diferentes tipos de sistemas a los que hay en la actualidad y aprovechar tanto Internet como las comunicaciones Web.

El mismo autor afirma que muchas empresas han apostado por la investigación en esta área para tener una mejor manera de hacer negocio. La administración del conocimiento entonces no debe considerarse como un conjunto de información, sino como una serie de soluciones que encajan en las necesidades actuales de las empresas para mantenerse en la competencia e incluso tomar ventaja en el mercado.

### 2.1.4. Visión general de la administración del conocimiento en las empresas

Según se desprende de los estudios realizados por Winch [7], el creciente desarrollo de la administración del conocimiento ha permitido que se reconozcan las mejores prácticas laborales y al mismo tiempo se intenten evitar por cualquier medio las anomalías en el trabajo.

Dicho estudio, remarca que el conocimiento no es importante por sí mismo, sino todo el proceso de entender, aprender y crearlo para poderlo transmitir a los demás. Uno de los principales obstáculos en las empresas respecto a la divulgación de conocimiento y buenas prácticas es la actitud de las personas, pues generalmente son unos cuantos los que poseen la experiencia o conocen los "secretos" de las labores diarias y es muy complicado hacer que se pueda compartir a los demás de una manera simple y correcta. Es por eso que un sistema de administración de conocimiento debe promover los siguientes aspectos [8]:

- Participación espontánea. Significa que los usuarios estén motivados a participar en el sistema.
- Objetivos comunes. Identificación de los usuarios para alcanzar los mismos resultados y mejorar la comunidad.
- Relaciones sociales. Promover la generación de conocimiento tácito y convertirlo en explícito para poderlo compartir.
- Repertorio común. Especificar un lenguaje previamente acordado para que todo el personal pueda comprenderlo de primera mano y personal especializado pueda actuar conforme a lo capturado en el sistema.

En consecuencia, según lo reportado por Agostini [8], el objetivo es reutilizar el conocimiento generado para aprovechar su potencial y dar nuevas alternativas de solución a las organizaciones. Sus resultados son muy sencillos de evaluar: la efectividad y eficiencia de los usuarios del sistema posterior a su consulta. Es decir, gracias a qué tanto alimentan y extraen información al sistema, esto hace que el usuario amplíe sus conocimientos para desempeñar aún mejor su labor.

## 2.2. Tecnología de información y desarrollo de aplicaciones

O'Reilly [9] asegura que la tecnología de información ha sabido abrirse paso a lo largo de tiempo y se ha ido involucrando cada vez más en la vida cotidiana a tal grado que dependemos en gran medida de ésta. No es la excepción en las empresas, que también han ido avanzando conforme la tecnología marca la pauta, desde el surgimiento del comercio electrónico, el apogeo de las empresas "punto com", los sistemas de colaboración, el auge de Internet hasta la expansión de servicios inalámbricos. Todo lo anterior resulta de la necesidad de seguir innovando y con ello surgen nuevas tendencias en el desarrollo de las aplicaciones.

### 2.2.1. Web 2.0

El término Web 2.0 se considera un concepto, más que una tecnología, asegura Cortés [10] según sus estudios, dado que su construcción se basa en infraestructura y lenguajes de codificación existentes en donde la novedad radica en las herramientas construidas. Dichas herramientas promueven la "Inteligencia colectiva", es decir, la comunidad de usuarios de forma conjunta crean y mejoran conceptos y procesos.

El principio básico de la Web 2.0 es una cultura de participación compartiendo conocimiento o experiencia, según lo comenta dicho estudio. Este concepto se puede aprovechar en las empresas en al menos dos maneras que ayudan a generar ventaja competitiva [10]:

- Para entender las comunidades donde sus productos y servicios son introducidos.
- Para mejorar la administración de conocimiento e incrementar la productividad.

Según el autor, la Inteligencia colectiva se vuelve un tema de alta relevancia en las empresas al ofrecer una comunicación con los clientes para conocer la imagen de sus productos ante el mercado, además de proveer las buenas prácticas o proponer mejoras en sus procedimientos laborales.

Por otro lado, Soto [11] afirma que la Web 2.0 puede referirse a cualquiera de las siguientes características:

- Web de lectura – escritura.
- Sitios que facilitan la colaboración o creación colectiva. (Figura 2.3)
- Sitios con una interfaz de usuario visualmente rica y altamente interactiva, a través de diversas tecnologías de nueva generación.
- Web como plataforma para crear aplicaciones u ofrecer servicios Web para crear composición de páginas en programas de terceros.

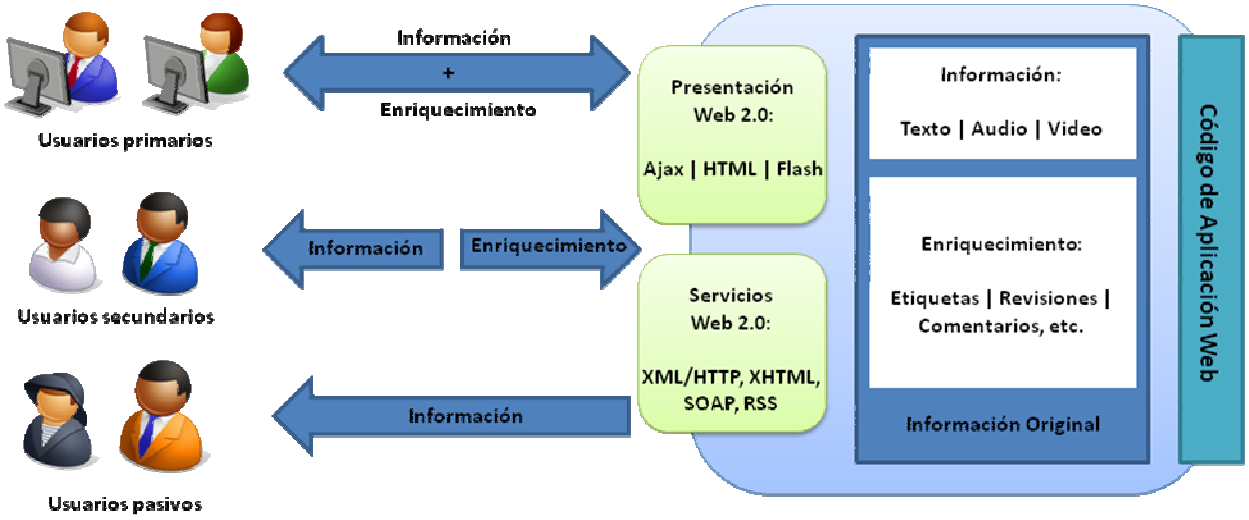


Figura 2.3
Web 2.0 Modelo de participación e información
(Adaptado de [12])

La Figura 2.3 resume los estudios realizados por Hinchcliffe [12], quien asegura que hay 3 tipos de roles en la Web 2.0, los cuales definen la forma de trabajar e interactuar con las aplicaciones:

- Los usuarios primarios, son muy participativos porque solicitan información, la proveen, e igualmente la enriquecen. Haciendo que la aplicación se mantenga activa y sobre todo útil.
- Los usuarios secundarios en la mayoría de los casos consumen información, aunque ocasionalmente pueden contribuir con observaciones o comentarios. Su papel es muy importante pues probablemente son un grupo más grande que los primarios y algunos se encaminan a convertirse en usuarios primarios conforme adquieren experiencia.
- Por último, los usuarios pasivos solamente consumen recursos (recopilan información) y sin duda son el mayor número de usuarios en la Web, utilizando la información en un solo sentido y sin retroalimentarla.

### 2.2.2. XML

Por sus siglas en inglés, eXtensible Markup Language (lenguaje de marcas extensible), XML es un meta lenguaje para crear vocabulario basado en texto y hecho a la medida que tiene como objetivo fundamental describir datos [13].

Es además un estándar desarrollado por el W3C (World Wide Web Consortium) [13] que propone el intercambio de información estructurada entre

diferentes plataformas, permitiendo la compatibilidad entre sistemas para compartir información de forma fiable, segura y fácil.

Según lo propuesto por la W3C [14], XML se puede resumir en varios puntos fundamentales:

- Es para estructurar datos. No es un lenguaje de programación, solo pretende facilitar la tarea de generar y leer datos, asegurando que su estructura no sea ambigua.
- Es parecido a HTML. Ambos usan etiquetas, pero HTML especifica cómo debe aparecer la información en el navegador de Internet. Mientras que XML solo delimita las piezas de datos, dejando la interpretación de éstos a la aplicación que los acceda.
- Es texto, pero no está pensado para ser leído. Por su simplicidad, es posible darle un vistazo al archivo XML dada su estructura, pero está destinado para ser utilizado por aplicaciones que lo interpreten.
- Es explícito por diseño. Dada su utilización de etiquetas, generalmente los archivos XML son más grandes y detallados que los HTML o binarios.
- Es una familia de tecnologías. El concepto por sí mismo es una especificación que define las etiquetas y los atributos, pero en su totalidad es un conjunto creciente de módulos que ofrecen servicios útiles para realizar tareas demandadas. En donde se involucran términos como Xpointer, Xlink, CSS, XSL, DOM, XSLT, XML Schemas, todos involucrados en la tecnología XML.
- Es reciente, pero no inmaduro. Aunque su desarrollo es de 1996, podría considerarse como una tecnología inmadura. Sin embargo, posee las mejores partes de los estándares HTML y SGML.
- Es modular. Permitiendo definir un formato de documento combinado y reusar otros formatos. Y mediante un XML Schema se soporta la modularidad a nivel de la definición de la estructura de documentos XML.
- Es la base de RDF (Resource Description Framework) y de la Web Semántica. RDF es un formato de texto XML que soporta aplicaciones de descripción de metadatos y recursos. Esto es para establecer mecanismos para acordar términos de comunicación e intercambio de información de manera efectiva.
- Es gratuito, independiente de plataforma y bien soportado. Haciendo una analogía, es como utilizar SQL para bases de datos, en donde existen programas que manipular la información, pero hay diversas opciones, sin tener que estar atado a un proveedor único dado que hay una creciente comunidad que ofrece soporte.

### 2.2.3. Modelos de creación o rediseño de aplicaciones

En la actualidad, dentro del área de desarrollo de software hay diversas maneras de crear o rediseñar aplicaciones, pero todas están orientadas en 3 modelos principales [15]:

- De escritorio. En este, la aplicación o herramienta informática mantiene el control principal del modelo de datos, creando una total dependencia entre la aplicación, el sistema operativo, el formato de los datos y la organización del sistema. En ese modelo ajustamos al usuario a las propiedades características del software. Por ejemplo, un procesador de textos, una manejador de hojas de cálculo, etc.
- Web 1.0 o "la vieja escuela" de las aplicaciones. Aquí, la funcionalidad se basa en el uso de formas electrónicas que transmiten información hacia un servidor de aplicación, usando un esquema de entrega – respuesta que depende en su gran mayoría en que la información vaya de ida y vuelta sin problemas de conectividad, uso de cookies y sesiones de usuario, que combinadas con las formas electrónicas antes mencionadas, permiten armar aplicaciones transaccionales de pasos múltiples.
- RIA (Rich Internet Application). Este modelo pretende facilitar aún más el acceso a aplicaciones robustas, simplemente con el uso de un navegador Web, aquí no debe importar la computadora que se esté utilizando o su sistema operativo. Se busca la mayor productividad estando en la computadora de trabajo o en la de casa, pues la información viaja usando estándares como XML y los contenidos multimedia pueden ser mostrados sin contratiempos usando plugins que los navegadores recientes incluyan.

### 2.2.4. Rich Internet Application y la Web 2.0

Eichorn [16] destaca que Internet ha cambiado mucho desde su creación, ya que empezó como un tipo de comunicación basado en texto simple, que a su vez permitiera generar otros medios más poderosos de comunicación. En la actualidad, es posible crear aplicaciones multimedia interactivas y también muy poderosas, por lo que el mismo autor afirma que la Web y especialmente las aplicaciones en Web son uno de los campos de desarrollo de software más importantes y que además, están creciendo con mayor velocidad.

Según explica Eichorn [16], las aplicaciones en Internet ofrecen una gran cantidad de beneficios cuando se comparan con una aplicación normal [16]. Pues son altamente accesibles, no requieren instalación, es posible actualizarlas en cualquier momento y sobre todo permiten acceder a una gran cantidad de información sin involucrar redes complejas. Al menos, estas ventajas permiten costos menores en el desarrollo y soporte cuando se comparan con una aplicación "habitual". Aunque en algunas ocasiones las aplicaciones en Internet ofrecen menor usabilidad debido a que son más simples, siguen reemplazando a las aplicaciones de escritorio.

Continuando con su estudio, Eichorn [16] asegura que el concepto Rich Internet Application (RIA) se refiere a una aplicación en Internet que pretende llenar ese hueco de usabilidad mencionado anteriormente mediante la colocación de más

código en el navegador, ofreciendo mayores niveles de interactividad, asemejándose a una aplicación nativa o de escritorio.

Las Rich Internet Applications son una tecnología íntimamente asociada a una nueva tendencia de soluciones de Internet llamada Web 2.0 que ofrecen [17]:

- Aplicaciones interactivas basadas en Web, tal como aplicaciones de escritorio.
- Desarrollo de aplicaciones sin restricciones o limitaciones de índole económico (desligarlas de las grandes empresas creadoras de software).
- Mayor fortalecimiento hacia el usuario a través del uso de estándares, permitiendo interoperabilidad entre sistemas.
- Preservación del patrimonio de las tecnologías Web (no olvidar que deben ser accesibles sin importar la conexión).

Cortés [10] asegura que las RIAs permiten crear aplicaciones con elementos visuales sofisticados e interactivos que brindan una mejor experiencia que lo que se puede lograr con tecnologías de las Web 1.0. Actualmente la tecnología que más se utiliza para desarrollar RIAs es Ajax, pero también se encuentran entre los principales Adobe Flex y OpenLaszlo.

### 2.2.5. ¿Qué son los programas Open Source?

Las características principales que categorizan a una aplicación como Open Source son [18]:

- Sus usuarios deben tener acceso al código fuente del programa.
- Empleo del software como se desee y para lo que se desee, en tantas computadoras se necesite y técnicamente en cualquier situación.
- Tener el software a disposición para que cubra sus necesidades, permitiendo mejorarlo, corregir errores, aumentar funcionalidad(es) y entender su operación.
- Redistribuir el software para otros usuarios, pudiendo hacerlo de manera gratuita o con algún cargo económico.

La filosofía del software Open Source propone libertad y no obligación, significando entonces que no es forzoso hacerle correcciones, mejorarlo o distribuirlo inclusive, pero en gran medida la comunidad de usuarios son los que hacen que un proyecto de este tipo crezca o quede en el olvido. Aunque para este caso existen las licencias de software, mismas que pretenden definir claramente los derechos y obligaciones que los usuarios tienen sobre él. Como por ejemplo, garantizar su redistribución, citar al autor del software original y a los trabajos que derivaron de éste.

Las motivaciones para el uso y promoción del Open Source son variadas. Como ventajas se pueden listar las siguientes [18]:

- Disponibilidad del código fuente y el derecho de modificarlo. Esto permite un mejoramiento ilimitado, pues esto significa, entre otras cosas, la facilidad de portar el código hacia un hardware nuevo y adaptarlo a condiciones diferentes para alcanzar un entendimiento detallado del funcionamiento del sistema. La disponibilidad del código fuente para un programador facilita el aislamiento de los errores y permite corregirlos sin mayores requerimientos.
- Derecho de redistribuir las modificaciones y mejoras al código para su reutilización. La capacidad de que el software sea modificable es un factor decisivo para que pueda ser compartido y mejorado por grupos de programadores con la finalidad de impulsar la generación de soluciones de software libre y proveer de valor agregado a sus productos sin depender de software propietario.
- Derecho de usar el software para lo que sea necesario. Por sí solo el software debe ser útil, agregarle las etiquetas de "libre y redistribuible" aporta popularidad y atrae seguidores para continuar trabajando en el proyecto. Aunque en este rubro entran cuestiones éticas, por el uso que se debe dar al software y que no son abordados en este trabajo de tesis, emplear el software para lo que se necesite promueve la diversidad de iniciativas que finalmente pueden convertirse en excelentes soluciones informáticas con el paso del tiempo y su frecuente utilización.
- No puede haber alguien con el poder de restringir de forma unilateral el modo en que se emplee el software. Esta ventaja va en el siguiente sentido: imaginemos a una empresa vendedora de software que ya no ofrezca actualizar las versiones de su aplicación en alguna plataforma no reciente, eso significa que sus clientes no tienen alternativa, deben actualizar su arquitectura o entorno de operación para ajustarse a la aplicación. Pero con Open Source, se puede financiar, por ejemplo el desarrollo de la solución requerida para dicha plataforma o buscar otros proveedores que provean la solución.
- No hay dependencia en el futuro hacia los desarrolladores de software propietario. Esto evita que si el desarrollador cierra sus puertas, se descontinúe inmediatamente la aplicación. Siempre hay oportunidad de tener el código y tratar de mantenerlo vivo hasta que surja alguna alternativa o nueva versión.
- No proliferan las "cajas negras". Es un punto muy importante para el Open Source, pues se evita la dependencia de los servicios que provee un software determinado. Con el acceso al código fuente es posible realizar una profundización de los algoritmos con la finalidad de hacer implementaciones personalizadas.
- Siempre está la posibilidad de bifurcarse. Un proyecto de Open Source permite crear alternativas de software que posiblemente no fueron

consideradas en el producto original y con el crecimiento de las nuevas tecnologías, esto nos permite trazar varios caminos para desarrollar versiones estables y confiables.

- Hay menores prioridades conflictivas generadas por presiones de mercado. Usualmente un software Open Source es liberado hasta que se encuentra listo y no es expuesto al mundo por cuestiones comerciales que al poco tiempo necesita de parches, service packs y demás adecuaciones que pudieron ser previstas. Sin embargo, esto puede verse como desventaja al mismo tiempo, pero en este caso la sana competencia entre proyectos puede ayudar, permitiendo la bifurcación, comentada en el punto anterior.

### 2.2.6. OpenLaszlo

Es una tecnología Open Source que se enfoca en la creación de Rich Internet Applications [19]. Inicialmente basó su funcionamiento en la reproducción de contenido multimedia por medio del reproductor Flash, que sin importar la plataforma, provee una mejorada experiencia al usuario. A partir de marzo del 2007, con su versión 4.0 también puede entregar aplicaciones como HTML dinámico o DHTML (por sus siglas en inglés) además de Flash.

El código de programación de OpenLaszlo es un lenguaje de marcas basado en XML, llamado LZX, también es posible introducir código JavaScript para definir la interacción con el usuario y los datos. Los archivos resultantes que son interpretados por el reproductor de Flash dentro del navegador son de tipo SWF (gráficos vectoriales).

OpenLaszlo produce aplicaciones independientes de plataforma y que pueden ser utilizados con un navegador Web, pues se incrusta la aplicación en un documento HTML.

### 2.2.7. Software para la administración de conocimiento

Existen herramientas para la administración del conocimiento aplicables a diversas áreas, entre los que destacan:

- Entrieva, Knowledge Engineering Workbench (eKEW) [20].
  Sistema que organiza el conocimiento por medio de taxonomías de forma automatizada basándose en el contenido de la información de varios orígenes en línea (archivo de taxonomía) y genera tantas clasificaciones como sean necesarias para relacionarlos con los archivos que están en el sistema.
- Open Text, Livelink for Knowledge Management [21].
  Repositorio de activos de información empresarial que asocia los metadatos (o información extra relacionada) con los documentos que están en el sistema, permitiendo su clasificación y búsquedas basadas en criterios.

- Convera, RetrievalWare [22].
  Suite de aplicaciones que rastrea, almacena y monitoriza desde sitios Web, repositorios o sistemas de archivos para indexar, extraer y filtrar contenidos, permitiendo recuperar información y generar categorías. Todo lo anterior se conoce como servicios de Knowledge Discovery y la suite está pensada para grandes volúmenes de información.
- University of Cambridge, Tool for Action Plans Selection (TAPS) [23].
  Proyecto desarrollado por el Instituto de Manufactura de dicha Universidad, es una herramienta que retiene el conocimiento de las resoluciones de problemas de tipo administrativo para capturar la experiencia institucional, diseminar sus lecciones y traducir dicha experiencia en acción, permitiendo tomar decisiones [24]. Este proyecto aplica al área de manufactura y se ha implementado en empresas grandes del Reino Unido, y en su momento fue considerado para su comercialización.
- Digital Quality Systems, Zavanta [25].
  Implementación de una base de conocimientos para capturar y transferir los detalles críticos de la operación de una empresa permitiendo su combinación para crear sitios dinámicos en línea.

Los proyectos antes mencionados en su mayoría son de índole comercial y su acceso es limitado para su evaluación, sin embargo cada uno posee características muy específicas que los convierten en opciones serias para implementarlos en áreas de aplicación diferentes, teniendo que ser ajustados a las necesidades del problema a atacar.

## 2.3. Resumen del capítulo

Se puede afirmar por todo lo anterior que la administración de conocimiento no es un tema reciente para las organizaciones, sin embargo, el impacto de llevarla a cabo como una filosofía de trabajo es lo que genera complejidad, pues se trata de evidenciar las actividades, técnicas, procedimientos y demás conocimiento generado en el interior de la organización con la finalidad de encaminarlo hacia los actores principales en la toma de decisiones o en la operación de tareas rutinarias, para promover una cultura de reutilización y mejor continua.

A lo largo del capítulo se abordó la importancia de los sistemas de colaboración en el área de manufactura y cómo la administración de conocimiento también aplica en este ámbito, se mencionaron herramientas de software existentes relacionadas a este punto, además de resaltar sus características primordiales.

Finalmente se estableció una relación directa con la tecnología de información y las nuevas tendencias de desarrollo de aplicaciones, en donde surge la gran expectativa de las Rich Internet Applications y cómo pueden ofrecer soluciones

innovadoras aprovechando el uso de Internet, los beneficios de los proyectos Open Source y la colaboración del concepto llamado Web 2.0.

El siguiente capítulo describe por lo tanto, la esencia de una Rich Internet Application, las tecnologías que involucra y la plataforma de desarrollo conocida como OpenLaszlo, que permite desarrollar aplicaciones de este tipo.

# Capítulo 3. Desarrollo de una Rich Internet Application

Como se mencionó en el capítulo anterior, las nuevas tendencias en el desarrollo de software se enfocan más en la experiencia con el usuario, aprovechando las ventajas que ofrece Internet y los proyectos Open Source.

En un esfuerzo para lograrlo, surgen términos como AJAX para ofrecer una experiencia visualmente rica y entre los frameworks de desarrollo orientado a las Rich Internet Applications sobresale OpenLaszlo. En este capítulo se abordarán ambos temas y se detallarán sus características que los convierten en opciones versátiles para programar aplicaciones basadas en Web.

## 3.1. AJAX

Según Mesbah [26], una nueva especie de aplicaciones Web aparecieron en respuesta a la limitada interactividad de las aplicaciones Web de la generación 1.0, acompañadas de la mano del concepto Rich Internet Application y de la técnica de desarrollo conocida como AJAX.

AJAX es el acrónimo de **A**synchronous **J**avaScript **A**nd **X**ML. Es un conjunto de modernas tecnologías para el desarrollo de aplicaciones que incorpora estándares de presentación basados en XHTML (HTML Extensible), hojas de estilo conocidas como CSS, despliegue, interacción, manipulación e intercambio de datos, recuperación asíncrona de información y el enlace de todos estos elementos por medio de JavaScript [26].

De acuerdo con los estudios de Dailey [27] los componentes que conforman a AJAX fueron desarrollados y estandarizados durante la última década y se han perfeccionado recientemente, haciéndolos más adecuados para el uso empresarial. Aquí se resumen las funciones de los elementos mencionados por dicho autor:

- DHTML. Ayuda a los diseñadores a desarrollar páginas más interactivas, por ejemplo: al pasar el cursor sobre un elemento de una página DHTML, el color puede cambiar o el tamaño del texto puede hacerlo. Inclusive, el usuario puede arrastrar y soltar imágenes (drag and drop) en diferentes lugares de la página.
- XML. Es empleado para codificar datos y transferirlos entre el servidor y el navegador Web. El objetivo fundamental de XML es permitir la interoperabilidad entre plataformas para el intercambio de datos en Internet.
- Hojas de estilo CSS (Cascading Style Sheets). Provee a los desarrolladores más control en cómo los navegadores Web despliegan las páginas para otorgar un diseño y apariencia específica a cada uno de los elementos que conforman la interfaz de usuario.

- JavaScript. Interactúa con el código HTML y hace las páginas de AJAX más activas, haciendo llamadas al servidor (usando XML generalmente) para solicitar datos nuevos y desplegarlos en el contenedor de la aplicación mientras el usuario continúa interactuando con el programa.

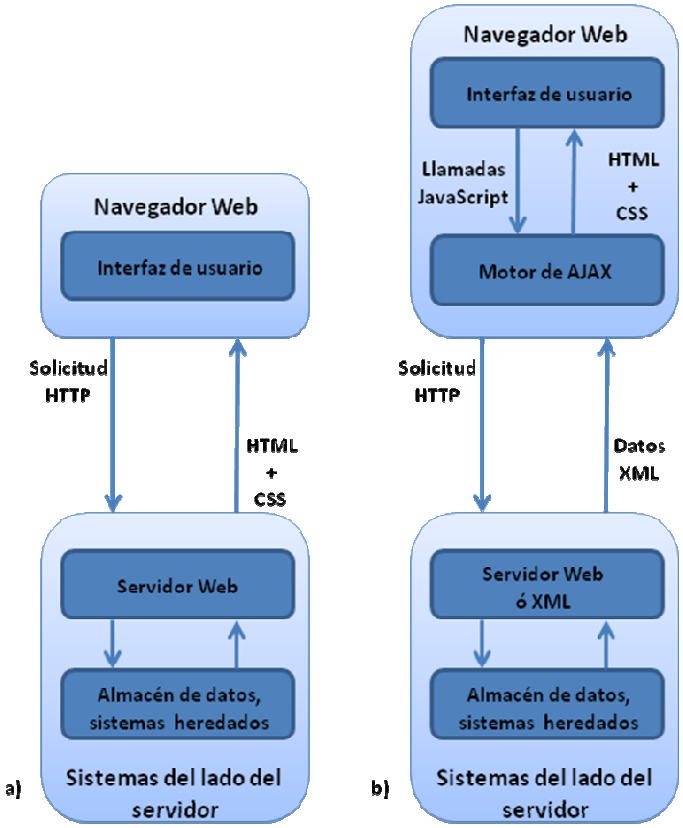


Figura 3.1
Aplicación Web tradicional vs Aplicación AJAX
(Adaptado de [27])

La Figura 3.1 muestra el comparativo de funcionamiento de una aplicación Web tradicional (Web 1.0, inciso a) contra una aplicación AJAX (inciso b). En la aplicación Web 1.0 el usuario activa una solicitud HTTP hacia el servidor Web, quien la procesa y regresa una página HTML al cliente y las subsecuentes recargas de información bloquean la aplicación hasta que el sistema actualiza la página. Mientras que una aplicación AJAX mediante un motor de JavaScript que corre en el navegador Web intercepta las entradas de datos del usuario, despliega el material solicitado y maneja la interacción del lado del cliente. En caso de necesitar material del servidor, lo realiza en segundo plano mientras permite que el usuario interactúe con la aplicación.

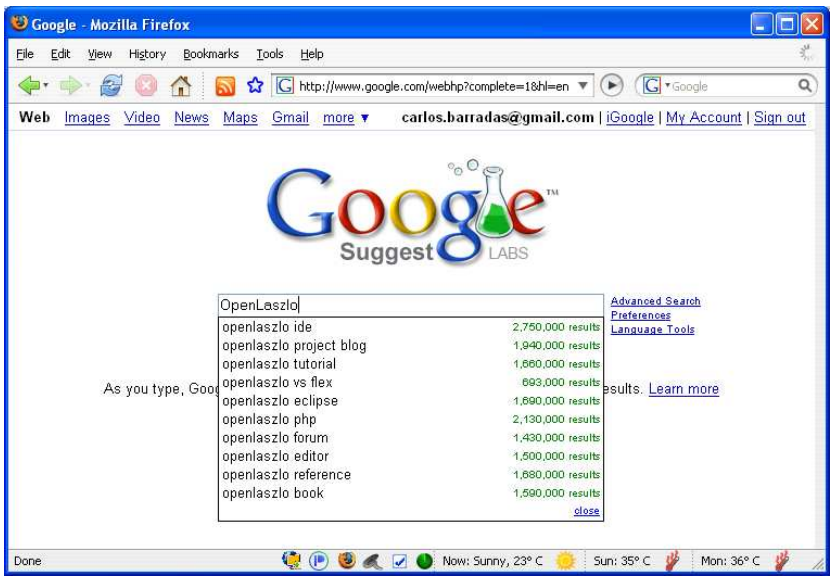


Figura 3.2
Google Suggest, ejemplo de aplicación AJAX
(Adaptado de [28])

Un claro ejemplo de una aplicación AJAX se muestra en la Figura 3.2, en donde Google ofrece un servicio de sugerencias basado en el texto que el usuario introduce en el cuadro de búsqueda y según se va conformando la palabra o frase automáticamente aparecen criterios de búsqueda relacionados para elegir entre las alternativas mostradas. Aquí la interacción con el usuario es más rápida e incluso la experiencia visual es más rica en contenidos y formas de despliegue a diferencia de aplicaciones de la generación Web 1.0.

De acuerdo con Laszlo Systems [29], es importante entender que AJAX es un método general para resolver un problema, no es un lenguaje, producto o conjunto de herramientas descargable de algún proveedor. Es solamente un término que incluye gran variedad de conceptos, plataformas y arquitecturas.

## 3.2. OpenLaszlo y el desarrollo de aplicaciones AJAX

OpenLaszlo, según afirma Coremans [30], fue desarrollado por Laszlo Systems, una compañía de software asentada en San Mateo California. OpenLaszlo se modeló como una suite de desarrollo de aplicaciones AJAX y su lenguaje de programación es llamado LZX, el cual se basa en XML y JavaScript por lo que facilita la interpretación y sintaxis del código, incluso para programadores principiantes. LZX es un lenguaje de programación orientado a objetos sensible a mayúsculas. Mientras que JavaScript se emplea para controlar eventos.

OpenLaszlo incluye un compilador que convierte los archivos fuente LZX hacia 2 tipos de formatos: Flash o HTML Dinámico (DHTML). Este compilador está escrito en Java, pero no es necesario entender o conocer Java para codificar una aplicación en LZX. Sin embargo, es necesario instalar el framework de Java y la suite de OpenLaszlo en el mismo lugar para que se puedan ejecutar los programas desarrollados en LZX.

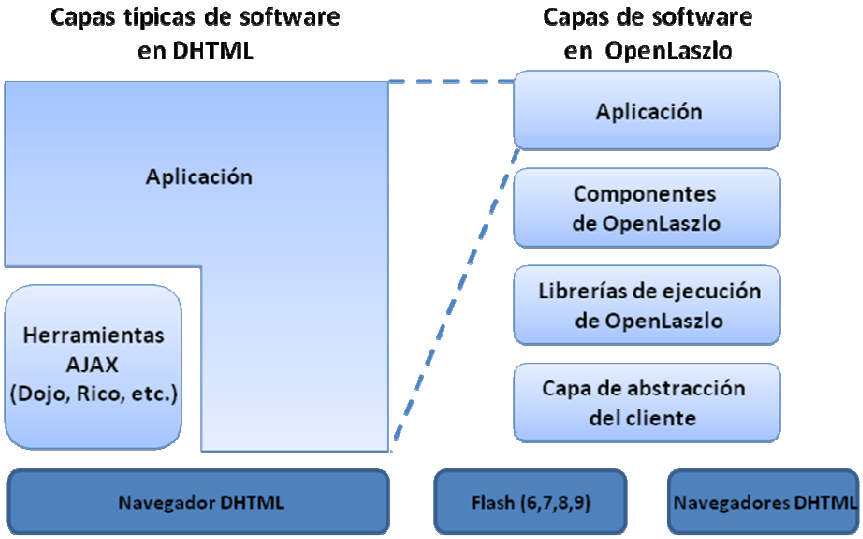


Figura 3.3
Comparativo entre frameworks de AJAX y OpenLaszlo
(Adaptado de [29])

En la Figura 3.3 se muestra un comparativo general de las capas que conforman a una aplicación de AJAX desarrollada con las herramientas tradicionales, mismas que carecen en su mayoría de frameworks de alto nivel en desarrollo y de librerías ricas en componentes, lo cual se traduce en mayor complejidad para codificar y dependencia de las versiones del navegador Web. Por otro lado, OpenLaszlo cubre estas carencias proporcionando librerías de ejecución y componentes de interfaz completos, en donde está presente una capa de abstracción que aísla al desarrollador de las características concretas de los navegadores Web y proporciona interfaces desplegables en versiones Flash variadas, así como en DHTML.

Para efectos de este trabajo de tesis se hace referencia únicamente al tipo de datos de Flash (SWF) como salida estándar para interpretación del navegador Web. Sin embargo, es posible decidir cuál formato se adapta mejor a un desarrollo en particular, de acuerdo a las observaciones realizadas por Laszlo Systems [29]:

- Reproductor Flash. Se considera un exitoso producto de software que está instalado en el 97% de las computadoras de los usuarios. El plug-in de Flash

es ligero, altamente confiable y operable entre diversos sistemas operativos, navegadores Web y diferentes dispositivos. Incorpora soporte para formatos de audio y video. Las nuevas versiones de este reproductor ofrecen un mejor desempeño y características especiales, sin embargo, pocos usuarios tienen instaladas las versiones recientes. Por este motivo OpenLaszlo permite entregar en formatos swf6 al swf9 inclusive. (Figura 3.4)

- DHTML (AJAX). Es un estándar que generalmente no requiere de plug-in y funciona mucho mejor en aplicaciones centradas en documentos o despliegue de información textual combinada con HTML. Sin embargo en el despliegue de videos o audio su rendimiento puede ser menor que el reproductor de Flash.

| | Reproductor Flash 6 | Reproductor Flash 7 | Reproductor Flash 8 | Reproductor Flash 9 |
|---|---|---|---|---|
| Mercados maduros | 99.3 % | 99.3% | 98.5% | 90.3% |
| EU / Canadá | 99.4 % | 99.4 % | 98.7 % | 90.5 % |
| Europa | 99.3 % | 99.1 % | 98.2 % | 90.5 % |
| Japón | 99.8 % | 99.8 % | 99.0 % | 89.8 % |
| Mercados Emergentes | 97.7 % | 97.6 % | 94.4 % | 89.4 % |

Figura 3.4
Presencia mundial del Reproductor Adobe Flash (Junio 2007)
(Adaptado de [31])

La figura anterior muestra la aceptación de los usuarios hacia el Reproductor Adobe Flash, lo cual indica que la gran mayoría de las computadoras lo tienen instalado en sus navegadores Web para desplegar contenidos multimedia en Internet. Los mercados maduros incluyen a Estados Unidos, Gran Bretaña, Francia, entre otros. Mientras que los mercados emergentes incluyen a China, Corea del Sur, Rusia, India y Taiwán.

### 3.2.1. Arquitectura

La arquitectura de OpenLaszlo, según sus desarrolladores [19], combina el poder y usabilidad de un diseño cliente / servidor con las ventajas administrativas y la efectividad que proveen las aplicaciones Web, entre las que destacan: software

más rápido, ligero y robusto, además de la disponibilidad del servicio y la facilidad de acceder a ellas remotamente.

#### 3.2.1.1. Modelo Cliente / Servidor

El servidor OpenLaszlo es una aplicación en Java que se puede comunicar con otros servidores de back end y orígenes de datos empleando una variedad de protocolos, de acuerdo con Laszlo Systems [19]. Las aplicaciones escritas en código LXZ se compilan por el OpenLaszlo server y éste las entrega en un formato que pueda ser interpretado por un navegador Web. Lo anterior se conoce como front end.

En el contexto de OpenLaszlo, cliente significa una aplicación LZX siendo ejecutada en el navegador Web del usuario. Mientras que servidor se refiere al servidor de OpenLaszlo. El cliente LZX y el servidor OpenLaszlo se comunican a través de HTTP; el servidor envía código máquina (bytecode) y la aplicación envía XML.

La Figura 3.5 que se muestra a continuación, presenta las capas que conforman la arquitectura de OpenLaszlo, en donde el navegador Web está instalado del lado del usuario. Allí es donde reside el reproductor de Flash y éste contiene a la aplicación Laszlo una vez que fue solicitada al servidor Web. La comunicación entre ambos es por medio de http y gracias a ello es posible acceder a contenidos multimedia, servicios Web y bases de datos que son coordinados por el Servidor de Presentación Laszlo.

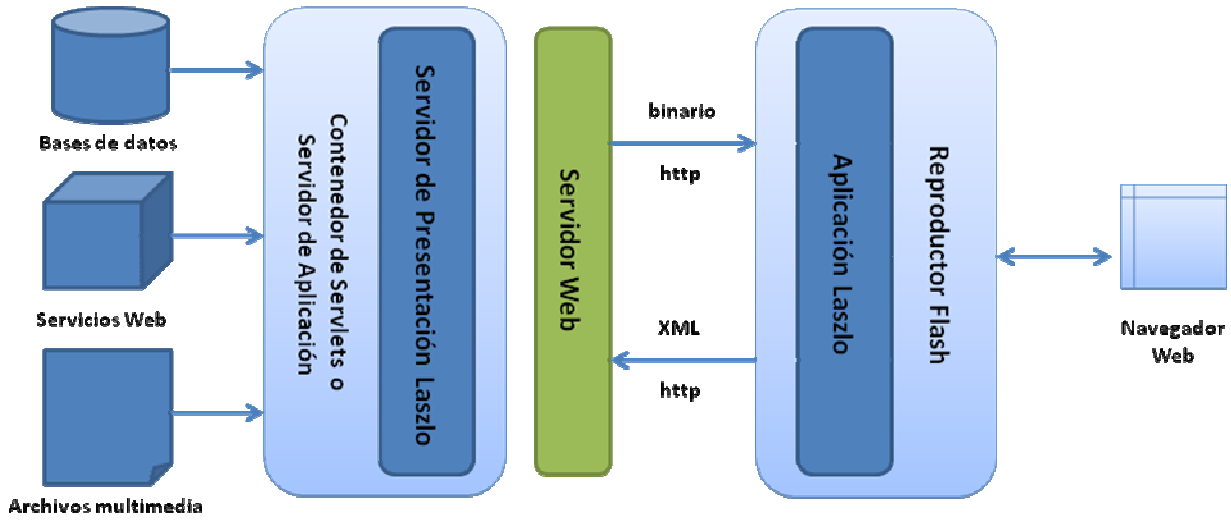


Figura 3.5
Arquitectura Laszlo Cliente - Servidor
(Adaptado de [19])

### 3.2.1.2. Servidor Laszlo

El servidor se ejecuta en un contenedor de servlets (tal como Apache Tomcat, WebSphere, Weblogic, Jetty, etc.) que esté corriendo Java Runtime Environment (JRE) versión 1.4 o superior. Consta de 5 subsistemas principales que se ilustran en la Figura 3.6 y se describen a continuación.

- El compilador, se encarga de interpretar las etiquetas LZX y de JavaScript y las traduce a código ejecutable hacia el ambiente del cliente.
- El codificador de medios, convierte un amplio rango de tipos de datos multimedia para colocarlos en el reproductor Flash.
- El administrador de datos, convierte todos los datos en un formato legible para las aplicaciones OpenLaszlo permitiendo recuperar información XML vía http.
- El administrador de conexión persistente, maneja la autenticación y comunicación en tiempo real de los sistemas que lo requieran y estén programados para ello.
- El caché, contiene la versión compilada más reciente de cualquier aplicación de OpenLaszlo. La primera vez que es solicitada, se compila y el archivo correspondiente (SWF o DHTML) es enviado al cliente. Haciéndose una copia de ésta en el servidor para que las solicitudes subsecuentes no sean compiladas nuevamente.


Figura 3.6
Arquitectura Servidor Laszlo
(Adaptado de [19])

La figura anterior muestra además que el Servidor de Presentación Laszlo (llamado LPS, por sus siglas en inglés) trabaja sobre las capas de Sistema Operativo y el Ambiente de Ejecución Java. Las solicitudes http se dirigen hacia el LPS y éste primero asocia la solicitud a un destinatario por medio del Administrador de conexión persistente, posteriormente el compilador detecta el código LZX a ser interpretado, establece las conexiones a los orígenes de datos y genera en el caché los contenidos resultantes para regresar la respuesta al solicitante. De esta forma, la comunicación entre los diversos componentes del LPS se da de manera interna en esa capa.

#### 3.2.1.3. Cliente Laszlo

La arquitectura del cliente, según sus desarrolladores [19], consiste en una librería compilada en cada aplicación de OpenLaszlo, llamada OpenLaszlo Runtime Library (ORL). Ésta provee servicios de temporalidad y presentación de contenidos gráficos y de sonido. Ninguno de estos componentes depende del modelo de objetos de Flash, pues está abierto a entregar contenidos en DHTML y existe la posibilidad de extenderlo hacia otros formatos que surjan en el futuro.

Cuando una aplicación de OpenLaszlo está en ejecución, todo el potencial necesario para ejecutar una aplicación LZX se descarga desde el principio al cliente. Además, aunque el usuario no esté realizando operación alguna sobre ella, se mantiene una conexión con el servidor [19].

Los cuatro componentes principales en la parte del cliente (ORL), son [19]:

- El sistema de eventos, reconoce y maneja los eventos de la aplicación tal como los clics del mouse o peticiones de datos hacia el servidor.
- El cargador / enlazador de datos, sirve como director de tráfico aceptando conexiones desde la red hacia el servidor y los enlaza a sus componentes visuales para que se desplieguen correctamente.
- El sistema de diseño y animación, provee una interfaz entre la programación y los elementos visuales de la aplicación, controlando sus estados, animaciones, posición y tiempos en pantalla.
- El sistema de servicios de OpenLaszlo, incluye el manejo del tiempo, sonido, cuadros de diálogo y textos informativos de la aplicación.

En la Figura 3.7 que se muestra a continuación, se ilustran las clases básicas de una aplicación OpenLaszlo del lado del usuario, en donde se establece toda la comunicación y temporalidad de los componentes que la conforman, todo esto embebido en el motor de Flash para efectos de la reproducción multimedia y el despliegue efectivo de los datos en el navegador Web, que forma parte de las interfaces de aplicación del sistema operativo.

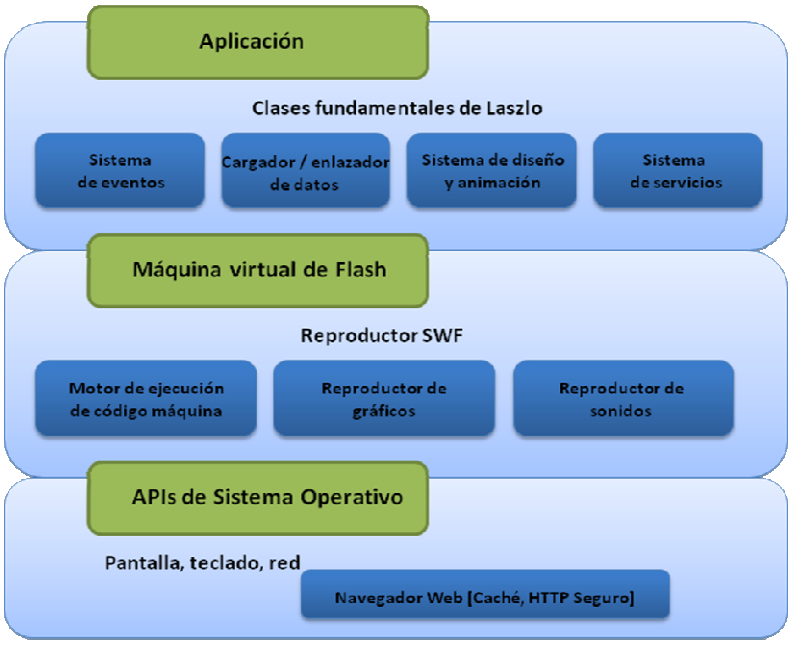


Figura 3.7
Arquitectura Cliente Laszlo
(Adaptado de [19])

Esta arquitectura de cliente permite dedicar las acciones concernientes a OpenLaszlo exclusivamente al nivel de la capa de Aplicación, mientras que los eventos manipulados para los efectos visuales son enviados e interpretados por la Máquina virtual de Flash. En la figura anterior se observa dicha separación y cada componente actúa de forma limitada, pero mantiene una comunicación directa para enviarle los eventos a la capa correspondiente, resultando en una relación bien estructurada.

### 3.2.2. Flujo general de datos

Una aplicación OpenLaszlo de acceso a base de datos, opera de la siguiente manera [19]:

- El usuario introduce una URL en su navegador Web.
- El servidor localiza el recurso solicitado por el cliente, generalmente un archivo LZX.
- Se compilan por única vez los gráficos, fuentes, sonidos y demás componentes de la aplicación y se convierten en código máquina (bytecode).
- El servidor de OpenLaszlo entrega un archivo ejecutable (SWF) al navegador Web incluyendo las clases fundamentales de Laszlo para que pueda ser interpretado.

- El navegador Web recibe el archivo SWF e invoca al reproductor Macromedia Flash para desplegar la aplicación.
- El usuario interactúa con la aplicación solicitando acceso a datos.
- El servidor OpenLaszlo (LPS) invoca al conector apropiado para recuperar la información solicitada (XML, SOAP, Web Service).
- LPS compila los datos, de ser necesario, y envía la información de vuelta al cliente.
- Las clases fundamentales de Laszlo del lado del cliente ligan los datos hacia los objetos apropiados de la interfaz y los elementos de las pantallas con actualizados con la información recién recibida.

La Figura 3.8, que se muestra a continuación, resume el proceso por el que los datos fluyen en una aplicación OpenLaszlo desde el inicio de la petición de URL, hasta la actualización de los contenidos dependiendo de las acciones solicitadas del usuario al momento de acceder a contenidos dinámicos (bases de datos o Web Services).

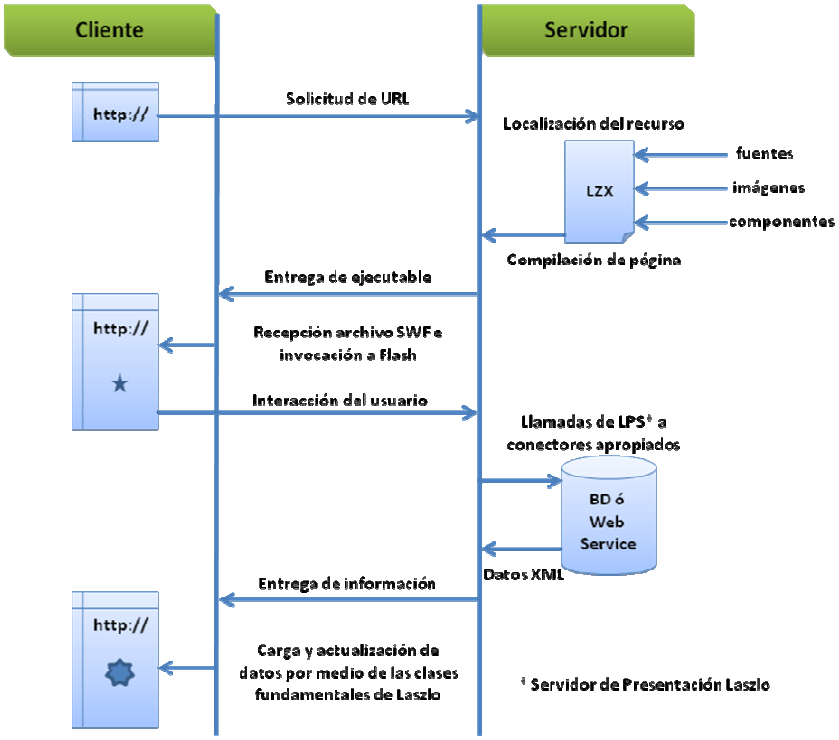


Figura 3.8
Flujo de datos en una aplicación OpenLaszlo
(Adaptado de [19])

Laszlo Systems [19] asegura que en las aplicaciones de OpenLaszlo la lógica de presentación está separada de la lógica de negocio y se ejecuta de forma local en

el cliente. El servidor envía al cliente en formato binario y comprimido en vez de enviarlo como texto, reduciendo la cantidad de datos transmitidos en comparación con las aplicaciones basadas en HTML y otro tipo de aplicaciones Web. El caché tanto del lado del cliente como del servidor elimina la ejecución de código y transmisión de datos innecesarios.

### 3.2.3. Visión general del desarrollo de una aplicación OpenLaszlo

El ciclo de programación según datos del sitio oficial de Laszlo [19] se resumen en 7 sencillos pasos:

1. Iniciar el servidor OpenLaszlo.
2. Escribir el código fuente, utilizando un editor de texto y guardarlo con la extensión .lzx
3. Colocar el archivo en la carpeta adecuada dentro del servidor.
4. Compilar la aplicación.
5. Depurar y modificar el programa.
6. Repetir desde el paso 2 al 5 hasta la perfección del programa.
7. Liberar la aplicación.

La compilación de la aplicación puede darse de 2 formas: se puede cargar directamente desde el navegador Web, haciendo una pre-ejecución del programa y mostrando en línea sus errores, si los hay; ó se puede invocar la compilación desde la línea de comandos, dependiendo del sistema operativo en uso.

La Figura 3.9 ilustra el proceso que involucra desarrollar una aplicación en OpenLaszlo, iniciando desde la edición del código fuente, incluyendo los orígenes de datos como pueden ser archivos multimedia o conexiones a bases de datos, todo siendo manipulado por archivos LZX que siguen el formato de XML. Cuando se envía a compilación, el ambiente de desarrollo de OpenLaszlo carga sus librerías, enlaza los componentes para mostrar en tiempo y forma los elementos visuales y genera el archivo destino que es la aplicación que despliega el navegador Web del cliente. De la misma manera, el proceso de compilación puede ser invocado por el navegador Web directamente hacia el servidor de desarrollo y colocando la aplicación en el contenedor de servlets para poder acceder al archivo destino.

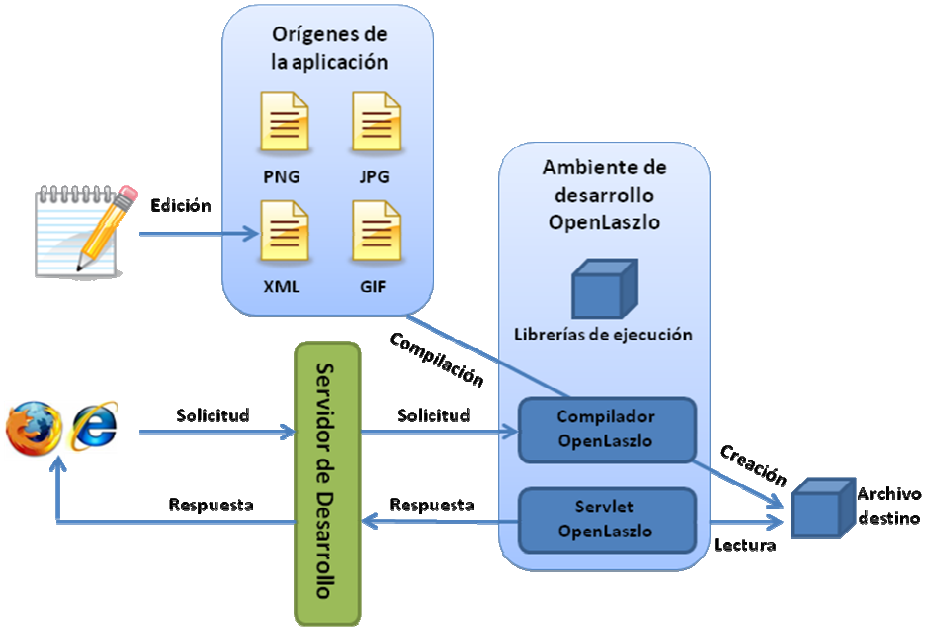


Figura 3.9
Ciclo de desarrollo de una aplicación OpenLaszlo
(Adaptado de [19])

En esta misma figura, dentro del ambiente de desarrollo de OpenLaszlo se encuentran las librerías de ejecución, mejor conocidas como clases fundamentales de Laszlo o LFC (por sus siglas en inglés) que incluyen componentes de interfaz de usuario, enlace de datos y servicios de red.

Las clases fundamentales de Laszlo soportan las siguientes características [19]:

- Componentes. Librería enriquecida de elementos de interfaz de usuario que incluyen componentes de formas Web, cuadrículas configurables para mostrar información estilo hojas de cálculo y vistas de árbol (estructuras jerárquicas).
- Diseño. Variedad de componentes de interfaz de usuario donde fácilmente se manipulan sus dimensiones.
- Animación. Un sistema de animación por medio de declaraciones en código aplicables a los elementos antes mencionados.
- Enlace de datos. Manejador de componentes de información que por medio de grupos de datos en XML (datasets) colocan los datos obtenidos por una consulta en los elementos visuales adecuados (cuadrículas, cajas de texto, etc).
- Servicios XML. Manejador de solicitudes HTTP para XML, SOAP, XML-RPC y JavaRPC Services, todos dedicados a la comunicación entre procesos.

- Depuración. Incluye un programa especial para encontrar errores en el código (debugger) mediante línea de comandos que inclusive muestra advertencias en el contexto de la aplicación.

Al igual que en figuras anteriores, la ventaja de tener componentes diversos y bien delimitados en cada capa, permite una ejecución controlada de actividades relacionadas con la compilación y entrega de contenidos en el navegador Web, en donde cada componente se integra con los demás para formar un resultado unificado, que se resume en el archivo destino.

### 3.2.4. Estructura general de una aplicación OpenLaszlo

Típicamente una aplicación de OpenLaszlo consta de las siguientes partes [19]:

- Canvas (en inglés). Es el nodo raíz de toda aplicación OpenLaszlo y solo debe existir uno por archivo LZX. Este elemento es el contenedor lógico de todos los nodos del programa. Visualmente es un rectángulo que se despliega en el área de contenido del navegador Web y de manera explícita se pueden definir en él características generales como alto y ancho, tal como se muestra en las Figuras 3.10 y 3.11.

```
<canvas width="500" height="150" bgcolor="green">
</canvas>
```

Figura 3.10
Ejemplo del elemento Canvas

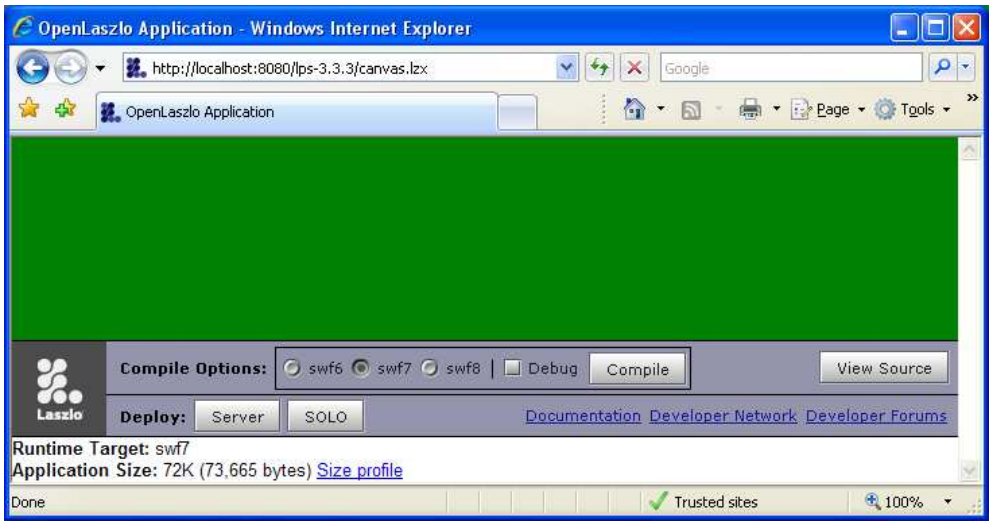


Figura 3.11
Resultado en modo depuración del elemento Canvas

- View (en inglés). Es el elemento visible básico de una aplicación OpenLaszlo. Cualquier componente que se despliega en el Canvas es un objeto View, como pueden ser imágenes, gráficas, texto, videos, etc.

La Figura 3.12 muestra a continuación un ejemplo de la sintaxis del elemento View y la Figura 3.13 el resultado correspondiente en el navegador Web.

```
<canvas height="200">
  <view bgcolor="red" x="50" y="50" width="100" height="100">
    <view bgcolor="yellow" x="50" y="50" width="60" height="105"/>
  </view>
</canvas>
```

Figura 3.12
Ejemplo del elemento View

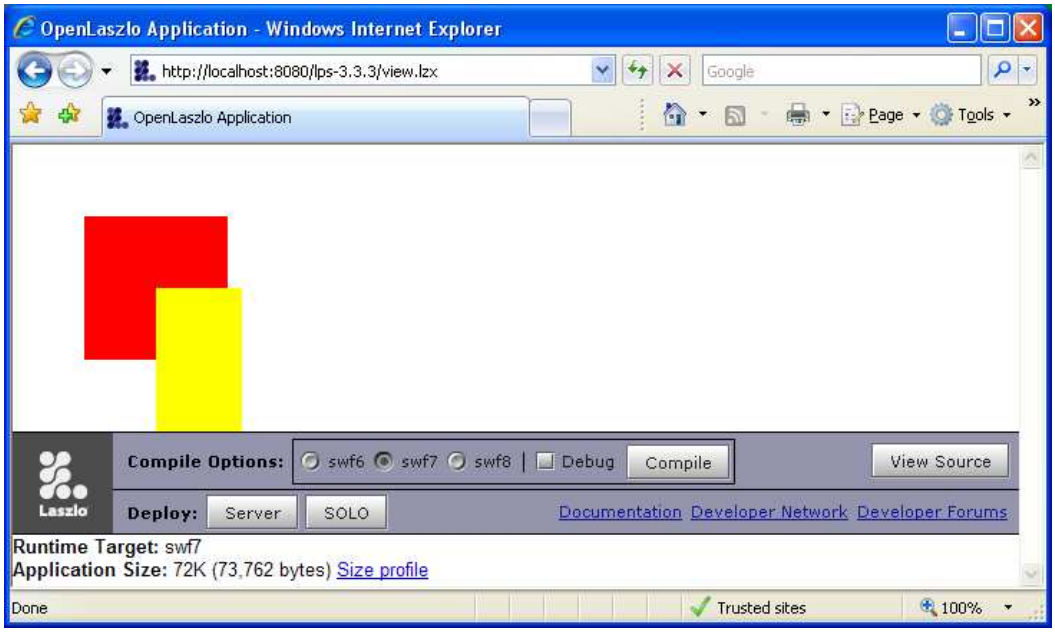


Figura 3.13
Resultado en modo depuración del elemento View

- Data (en inglés). LZX deriva mucho de su potencial gracias a la manera particular de implementar el enlace de datos, en donde los contenidos de un elemento View son determinados por los elementos de un Dataset (conjunto de datos). Las Figuras 3.14 y 3.15 resumen lo anterior.

```
<canvas height="100">
  <dataset name="ds">
    <record x="10" y="20" name="primero" color="332136432"/>
    <record x="5" y="5" name="segundo" color="56521236432"/>
    <record x="20" y="2" name="tercero" color="1565336432"/>
  </dataset>
  <simplelayout axis="y"/>
  <view datapath="ds:/record">
    <text datapath="@name" bgcolor="$path" x="$path"/>
  </view>
</canvas>
```

Figura 3.14
Ejemplo del enlace simple de datos

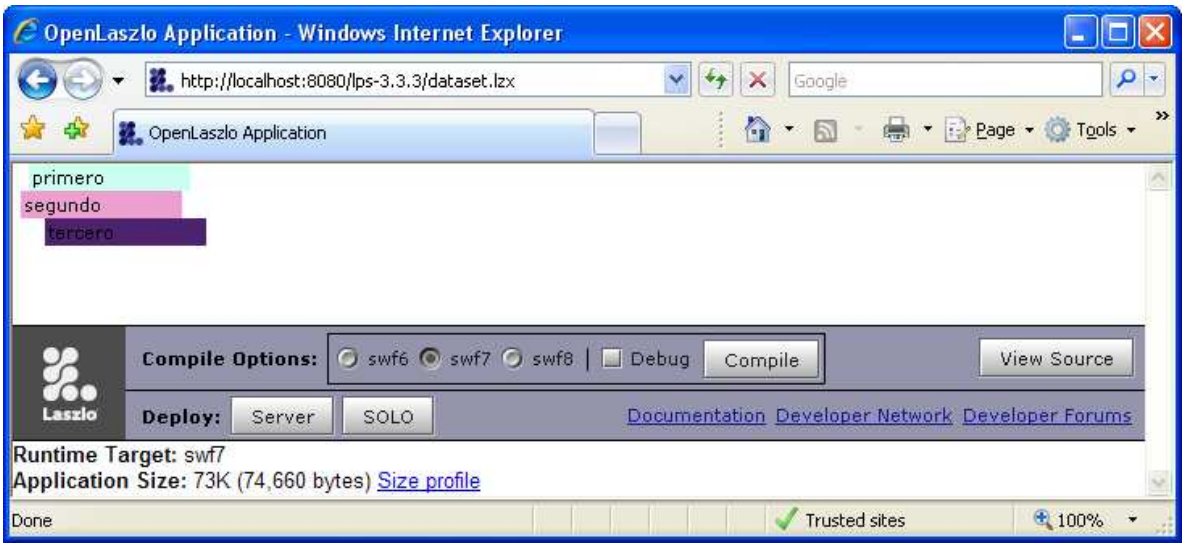


Figura 3.15
Resultado en modo depuración del enlace simple de datos

- Includes y Libraries (en inglés). El código fuente de una aplicación LZX puede contenerse en un solo archivo o es posible dividirlo en archivos pequeños para incrementar su facilidad de mantenimiento o la comprensibilidad de la aplicación, lo anterior también es con el fin de mejorar la modularidad y organización del código.

```
<canvas height="100">
  <include href="library.lzx"/>
  <miventana/>
</canvas>
```

Figura 3.16
Ejemplo del elemento Include

La Figura 3.16 muestra la sintaxis al utilizar el elemento Include, en donde necesariamente debe existir otro archivo, en este caso library.lzx, que es invocado por el elemento Canvas al ser cargada la aplicación.

```
<library>
  <class name="miventana" extends="window" title="Titulo personal"
      width="150">
    <button> Haz clic aqui ! </button>
  </class>
</library>
```

Figura 3.17
Ejemplo del elemento Library, archivo library.lzx

La Figura 3.17 ilustra la sintaxis necesaria para hacer uso de las Librerías en OpenLaszlo, de tal manera que los archivos invocados permitan crear los componentes que la aplicación en ejecución necesita. Esta figura complementa a la Figura 3.16 y su resultado en el navegador Web se muestra en la Figura 3.18.

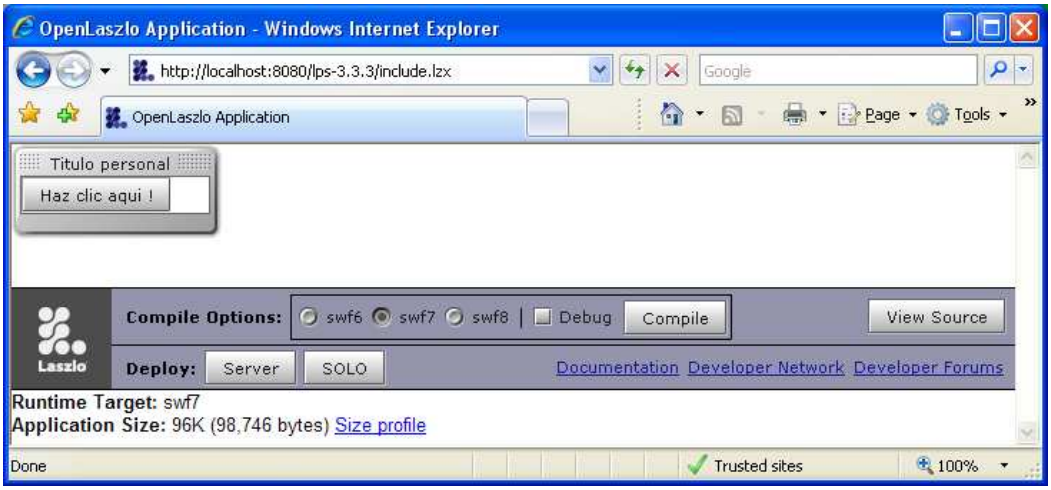


Figura 3.18
Resultado en modo depuración del elemento Include, referenciando al archivo library.lzx

- Comentarios. Muy útiles para asistir en la depuración de errores o sólo para incluir explicaciones de código, como lo muestra la Figura 3.19.

```
<canvas height="100">
  <simplelayout/>
  <!-- Este es un elemento View color rojo  -->
  <view bgcolor="red" width="100" height="20"/>
  <?ignore
  <!-- Este es un elemento View color azul -->
  <view bgcolor="blue" width="100" height="20"/>
  <!-- Este es un elemento View color verde -->
  <view bgcolor="green" width="100" height="20"/>
  ?>
</canvas>
```

Figura 3.19
Ejemplo de comentarios en el código

## 3.3. Resumen del capítulo

En este capítulo se discutió acerca de AJAX, sus características generales y ventajas sobre las aplicaciones Web de la generación 1.0. A su vez, se abordó con detalle el framework de desarrollo conocido como OpenLaszlo y se destacaron sus diferencias sobre otros desarrollos que también permiten la elaboración de aplicaciones basadas en AJAX.

Desde una perspectiva de arquitectura de software se mostró el funcionamiento de OpenLaszlo y la estructura de sus programas, resaltando la importancia de XML como base de su código fuente en un lenguaje llamado LZX que tiene su propia sintaxis y reglas de compilación.

OpenLaszlo además de ser un proyecto Open Source, se ofrece bajo la licencia Pública Común Certificada por OSI, que le permite ser libre para desarrollar e implementar. Aunado a las demás características como experiencia cinemática, entrega de información vía Flash o DHTML, interoperabilidad entre plataformas y adaptación a un rápido desarrollo de prototipado, fue considerado como una excelente alternativa para desarrollar el sistema de administración de conocimiento que se detalla en el siguiente capítulo.

# Capítulo 4. Sistema de administración de conocimiento

En el capítulo previo se detallaron las características de las aplicaciones AJAX y en concreto de OpenLaszlo, que es una plataforma Open Source para desarrollar aplicaciones Web que ofrecen interfaces de usuario con capacidades del tipo de software de escritorio.

Este capítulo describe la implementación de OpenLaszlo para desarrollar el sistema de administración de conocimiento, considerando su arquitectura, diagramas UML, diccionario de datos y módulos desarrollados.

## 4.1. Arquitectura del sistema

De acuerdo a lo comentado en capítulos anteriores, la arquitectura del sistema desarrollado se diseñó orientada a 3 capas fundamentales que se ilustran a en la Figura 4.1:

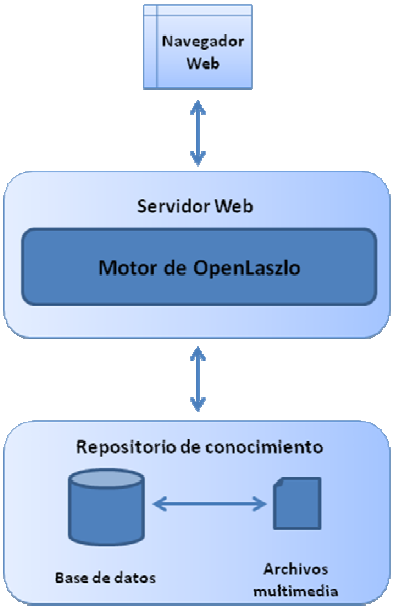


Figura 4.1
Arquitectura del sistema desarrollado

- Navegador Web. Enfocado exclusivamente al usuario y a la presentación de la información.
- Servidor de aplicación Web. Donde se encuentra la lógica del sistema y la comunicación entre las capas exteriores.
- Repositorio de conocimiento. Almacén de información estructurada en base a registros ordenados y manejo independiente de los archivos anexos que pueden contener video, audio y / o texto.

El objetivo de esta arquitectura consiste en mantener en un solo repositorio, información relevante del conocimiento generado en una organización como lo son: procesos, técnicas, estándares y modos de operación. La información puede ser muy variada, pero debe seguir una estructura específica para poder alimentar el sistema de administración de conocimiento, ésta clasificación se detalla en el modelo de datos.

La ventaja de implementarlo en Web radica en la facilidad para el usuario de poderlo acceder con un navegador mediante una URL predefinida, semejante a consultar el correo electrónico o la página de la organización.

### 4.1.1. Herramientas tecnológicas empleadas para el desarrollo

Es imprescindible el uso de las herramientas tecnológicas recientes para lograr desarrollos innovadores, como lo es el caso de una Rich Internet Application. Por este motivo se eligieron los siguientes programas / herramientas para la creación de este sistema:

- XAMPP Versión 1.5.5. Paquete de software libre independiente de plataforma que incluye: base de datos relacional MySQL, servidor Web Apache, intérprete para lenguajes de script PHP y Perl, además de una herramienta de administración para MySQL llamada phpMyAdmin, servidor FileZilla para transferir archivos vía FTP y un servidor de mail llamado Mercury [32]. El nombre de esta suite de aplicaciones para desarrollo proviene de las iniciales de sus herramientas principales: **A**pache, **M**ySQL, **P**HP, **P**erl, mientras que la **X** indica que se puede instalar en diferentes sistemas operativos.
- Java Development Kit Versión 5.0. Entorno de la tecnología Java para desarrolladores, que incluye un compilador, componente para archivar, generador de documentación, depurador y el entorno de ejecución para correr sobre aplicaciones sobre la máquina virtual de Java [33]. Es requisito para que el servidor de OpenLaszlo pueda ejecutarse.
- OpenLaszlo Versión 3.3.3. Plataforma de desarrollo Open Source basado en XML para desarrollar Rich Internet Applications [19]. Utiliza el contenedor de servlets Tomcat versión 5.0.24 y esta versión de OpenLaszlo solamente entrega contenidos compilados en formato Flash (SWF).
- MySQL Connector Versión 5.0.4. Controlador para conectarse a la base de datos MySQL y poder operar con ella mediante queries en SQL. Permite hacer las operaciones necesarias para ingresar y recuperar datos en los registros [34]. Necesario para que OpenLaszlo pueda acceder a la base de datos desde el código.
- Eclipse Versión 3.1.2. Entorno integrado de desarrollo (IDE por sus siglas en inglés) de tendencia Open Source enfocado al desarrollo de aplicaciones por

medio de frameworks, herramientas y librerías para crear y mantener software [35].

- Eclipse Tomcat Launcher plugin Versión 3.1. Pieza funcional en Eclipse para controlar de una manera sencilla el servidor de aplicación Tomcat y poderlo integrar en el ambiente de desarrollo de manera natural, controlando eventos en la interfaz gráfica y no por medio de línea de comandos [36].
- Web Tools Platform (WTP) Project Versión 1.0.3. Herramientas extendidas de Eclipse para desarrollar aplicaciones Web [37]. Incluye editores gráficos para una gran variedad de lenguajes, asistentes para crear aplicaciones y simplificar el desarrollo. Es elemento necesario para instalar el IDE4Laszlo.
- IDE4Laszlo. Plugin que se integra en Eclipse para programar en OpenLaszlo. Inicialmente fue desarrollado por IBM y posteriormente Eclipse Foundation lo clasificó como proyecto archivado [38]. Permite entre otras cosas el coloreado, autocompletado y formato del código LZX, previsualización de la aplicación y diseño integrado de los componentes visuales.

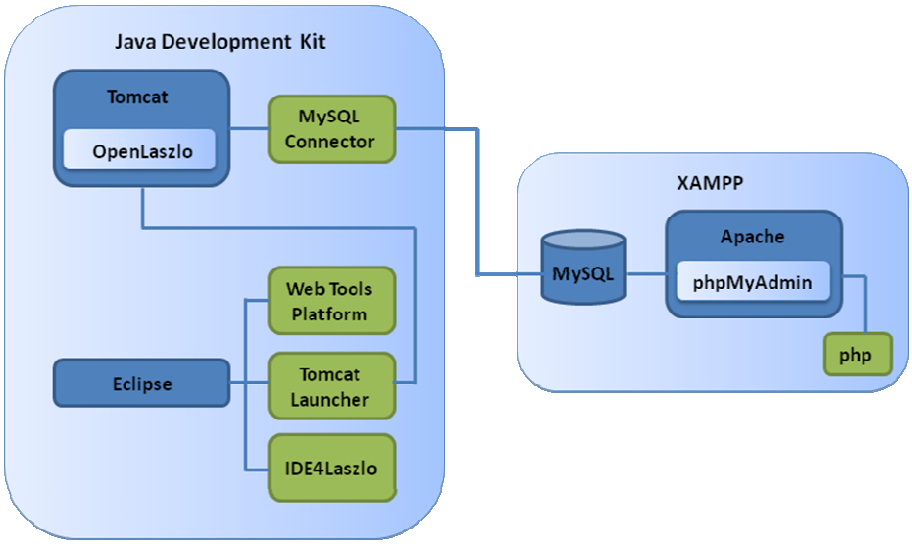


Figura 4.2
Relación entre las herramientas de desarrollo

La Figura 4.2 muestra las relaciones en las herramientas involucradas en el desarrollo del sistema, donde XAMPP funge primordialmente como la suite en Web para administrar la base de datos MySQL por medio de phpMyAdmin. Mientras que el Kit de Desarrollo de Java soporta al servidor de aplicación Tomcat y al entorno de programación Eclipse, el primero para poder ejecutar OpenLaszlo y el último para poder programar el código fuente.

## 4.2. Casos de uso

El Lenguaje Unificado de Modelado (UML, por sus siglas en inglés) permite comprender el análisis y diseño de un sistema de software. Por esta razón, se ilustran en la Figura 4.3 los casos de uso de UML para el sistema de administración de conocimiento desarrollado:

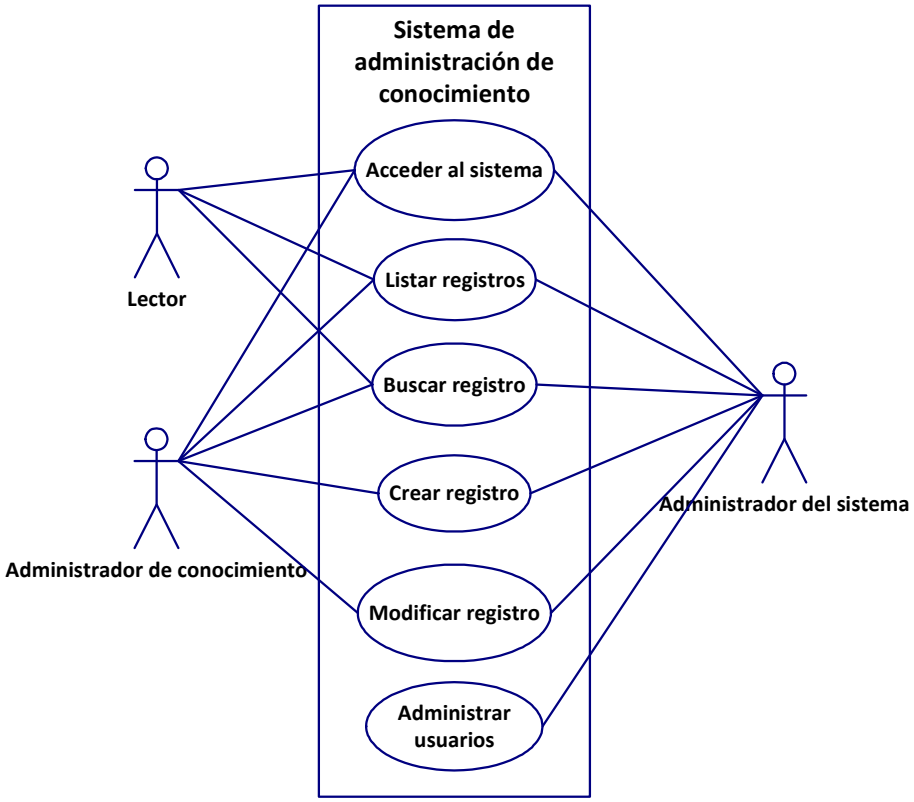


Figura 4.3
Diagrama de casos de uso

La figura anterior evidencia que en el sistema hay definidos 3 tipos de usuario: Lector, Administrador de conocimiento y Administrador del sistema, todos con un nivel de permiso determinado. Esto es con la finalidad de mantener bien delimitados los alcances de cada tipo de usuario que acceda a la aplicación.

Las operaciones permitidas en el sistema están orientadas a crear o agregar conocimiento, más que a eliminarlo de la aplicación (aunque sí es posible eliminar conocimiento de ésta).

Las funciones básicas en la operación del sistema son listar y buscar registro, las cuales son compartidas para cualquier tipo de usuario. En cuanto a la creación / borrado / modificación de conocimiento es una operación de los niveles

administrativos (de conocimiento y de sistema). Mientras que la administración de usuarios es exclusiva del Administrador del sistema.

## 4.3. Diagrama de clases

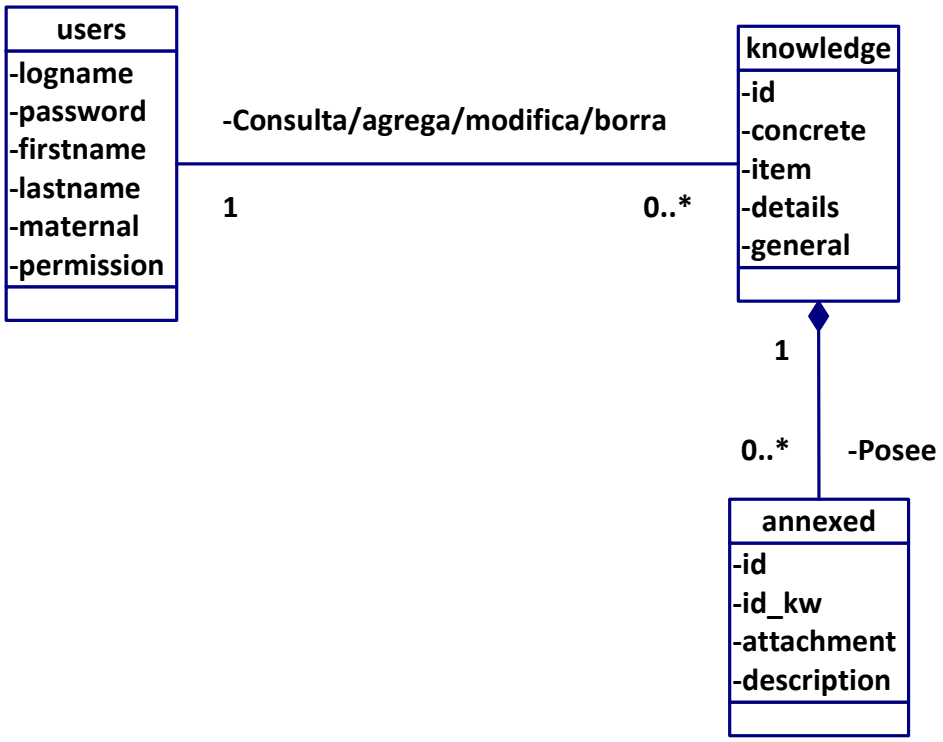


Figura 4.4
Diagrama de clases

Las clases involucradas en el funcionamiento del sistema se muestran en la Figura 4.4 y se detallan a continuación:

- Users. Entidad que contiene información relevante de la persona que tiene una cuenta en el sistema, se almacenan datos personales y el tipo de acceso permitido (Lector, Administrador de conocimiento o Administrador del sistema). Los usuarios no son propietarios de los registros almacenados, todos podrán consultar la misma información, solo que con diferentes niveles de acceso.
- Knowledge. Entidad contenedora de información relacionada a un artículo de conocimiento. Posee datos concernientes a un elemento almacenado en el sistema.
- Annexed. Registro independiente de los archivos anexos que se adjuntan en la descripción de un registro de conocimiento. Pueden existir múltiples archivos asociados a un mismo artículo de conocimiento.

## 4.4. Diccionario de datos

Mientras que con el diagrama de clases es muy sencillo identificar la relación de los elementos principales del sistema, con el diccionario de datos podemos entender la estructura de la base de datos, conocer sus tablas y los atributos de los campos que la conforman. Todo esto se muestra en Figura 4.5:

**annexed**

| Campo | Tipo | Nulo | Comentarios |
|---|---|---|---|
| id | int(11) | No | ID del anexo |
| id_kw | int(11) | No | ID del registro de conocimiento |
| attachment | varchar(200) | No | Nombre del archivo anexo |
| description | varchar(250) | No | Detalle del archivo anexo |

**knowledge**

| Campo | Tipo | Nulo | Comentarios |
|---|---|---|---|
| id | int(11) | No | ID del registro de conocimiento |
| concrete | varchar(50) | No | Caracteristicas especificas del item |
| item | varchar(50) | No | Nombre del registro |
| details | varchar(50) | No | Descripcion del item |
| general | varchar(50) | No | Caracteristicas generales del item |

**users**

| Campo | Tipo | Nulo | Comentarios |
|---|---|---|---|
| logname | varchar(50) | No | Usuario del sistema |
| password | varchar(50) | No | Clave del usuario |
| firstname | varchar(50) | No | Primer nombre del usuario |
| lastname | varchar(50) | No | Apellido paterno del usuario |
| maternal | varchar(50) | No | Apellido materno del usuario |
| permission | varchar(25) | No | Tipo de usuario |

Figura 4.5
Diccionario de datos

## 4.5. Clasificación de la información

Gracias al diagrama de clases y al diccionario de datos es posible tener una vista general de cómo se estructura la información en el sistema y cómo es que se almacena dentro de la base de datos para poderla recuperar.

El proceso de clasificación de la información del sistema de administración de conocimiento desarrollado en este trabajo, puede aplicarse de 2 maneras:

- Almacenar datos relevantes de todo un proyecto. En donde cada una de sus etapas, fases o componentes son guardados como registros dependientes y relacionados al mismo nivel. Esto ofrece la ventaja de detallar los niveles inferiores, dado que se pueden agregar tantos archivos anexos como se necesite en un solo registro, dependiendo de los límites de espacio del servidor (base de datos y directorio de anexos).
- Almacenar información de proyectos independientes, en donde cada registro contiene solamente la información general de un proyecto particular. Aquí el nivel de detalle es menor que en el caso anterior. Pero puede funcionar como un contenedor general de proyectos y sus archivos más relevantes.

La Figura 4.6 demuestra lo previamente descrito:

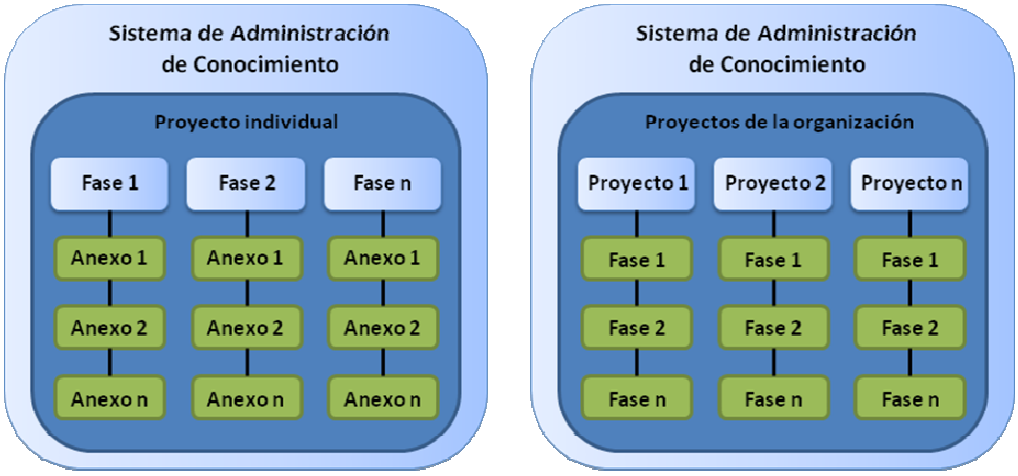


Figura 4.6
Tipos de clasificación de información en el sistema

En ambos casos, es necesario que el usuario con el rol de Administrador de conocimiento previamente tenga clasificada la información para iniciar la carga de datos, pues esto le ayudará a clarificar la estructura que seguirá la información almacenada. A su vez, es necesario dar a conocer a los demás administradores de conocimiento la categorización elegida, para que la información se vaya almacenando de manera coherente.

Es importante mencionar que no es posible cambiar de clasificación cuando el sistema ya tiene datos, esto significa que debe estar vacío el repositorio de conocimiento para cambiar de un modo de clasificación a otro.

## 4.6. Arquitectura de tecnologías utilizadas

Sustentando la propuesta de la Figura 4.1, donde se muestra de forma general la arquitectura del sistema de administración de conocimiento; en la Figura 4.7 que se ilustra a continuación, se destaca la relación e integración de las tecnologías utilizadas para el funcionamiento del sistema desarrollado.

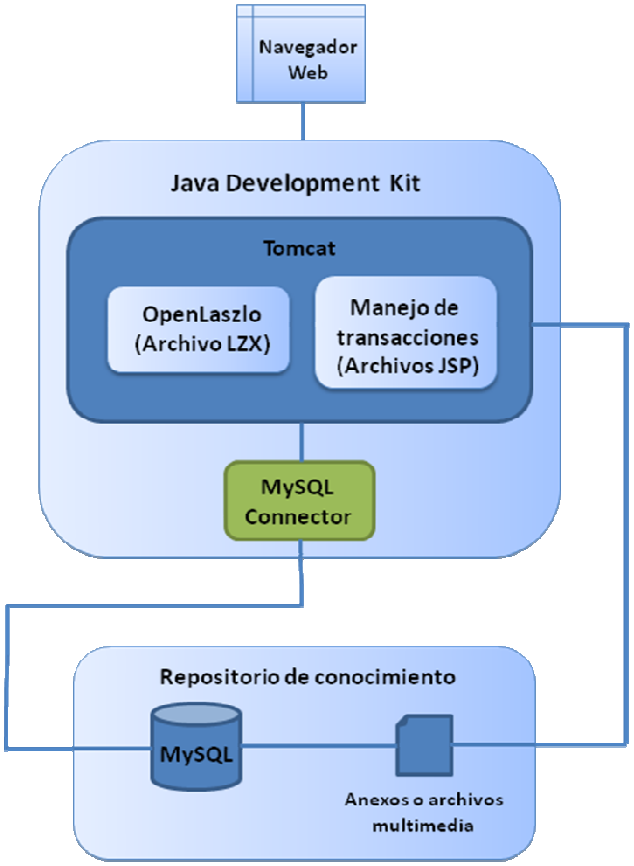


Figura 4.7
Diagrama de despliegue del sistema

La base del funcionamiento de la aplicación se da gracias al entorno de ejecución de Java, en donde se ejecuta el servidor Web Tomcat. OpenLazlo por su parte, atiende las solicitudes realizadas por el cliente cuando hacen llamadas a la aplicación en general (al sistema de administración de conocimiento).

La lógica de la aplicación en cuanto al manejo de las conexiones con la base de datos y la recuperación de registros es por medio del conector de MySQL y la ejecución de archivos JSP que son interpretados por Tomcat exclusivamente. El caso particular del manejo de los archivos anexos también se controla con JSP's, para dejar la lógica de presentación a OpenLaszlo y regresar las respuestas al cliente en una sola vía.

## 4.7. Diagramas de robustez

Con la finalidad de resumir en diagramas sencillos y comprensibles las operaciones del sistema y los objetos involucrados en cada proceso, resultan de gran utilidad los diagramas de robustez, que representan el curso básico de los casos de uso sin caer en el detalle de los diagramas de flujo de información. Los siguientes 4 diagramas demuestran lo anterior.

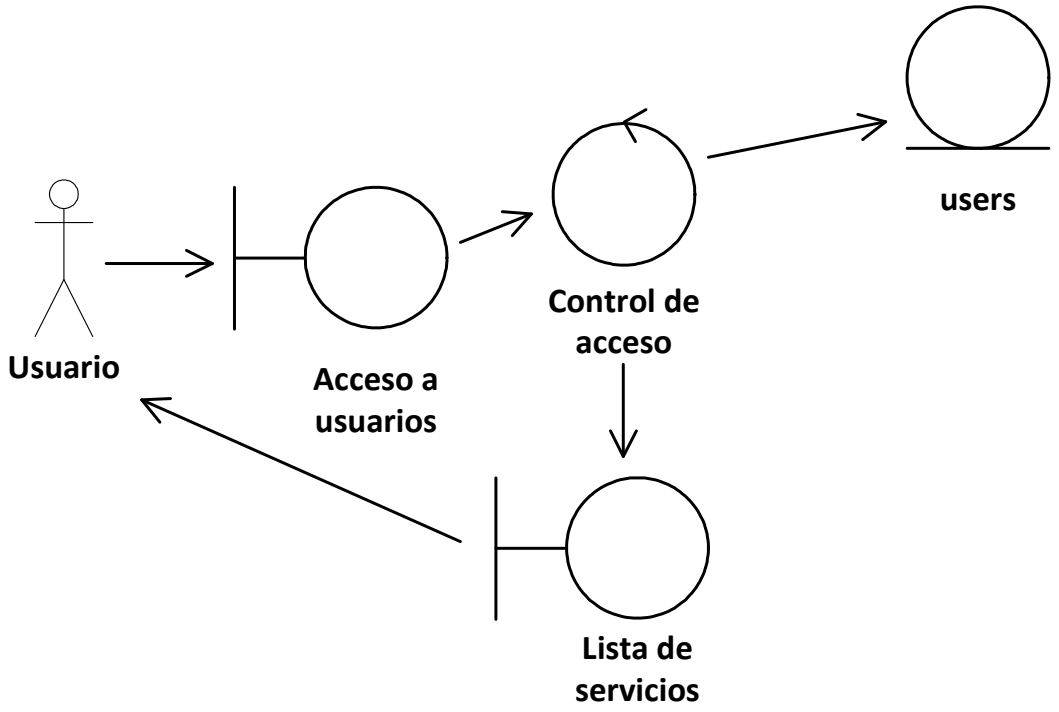


Figura 4.8
Diagrama de robustez de acceso al sistema

La Figura 4.8 resume el acceso a la aplicación, la cual inicia cuando el usuario introduce una URL en el navegador Web para posteriormente mostrar la pantalla de identificación o página de login. Esta es la que se considera interfaz de acceso a usuarios.

El control de acceso consiste en validar la información que el usuario proporciona contrastándolo con los datos contenidos en el repositorio de usuarios, donde se establece quién y con qué permisos pueden ingresar al sistema. En caso de introducir información no válida, se le informa al usuario que el acceso le ha sido negado mediante la misma interfaz.

El listado de servicios es una interfaz que despliega al usuario identificado las funciones disponibles dependiendo de sus permisos para poder operar el sistema.

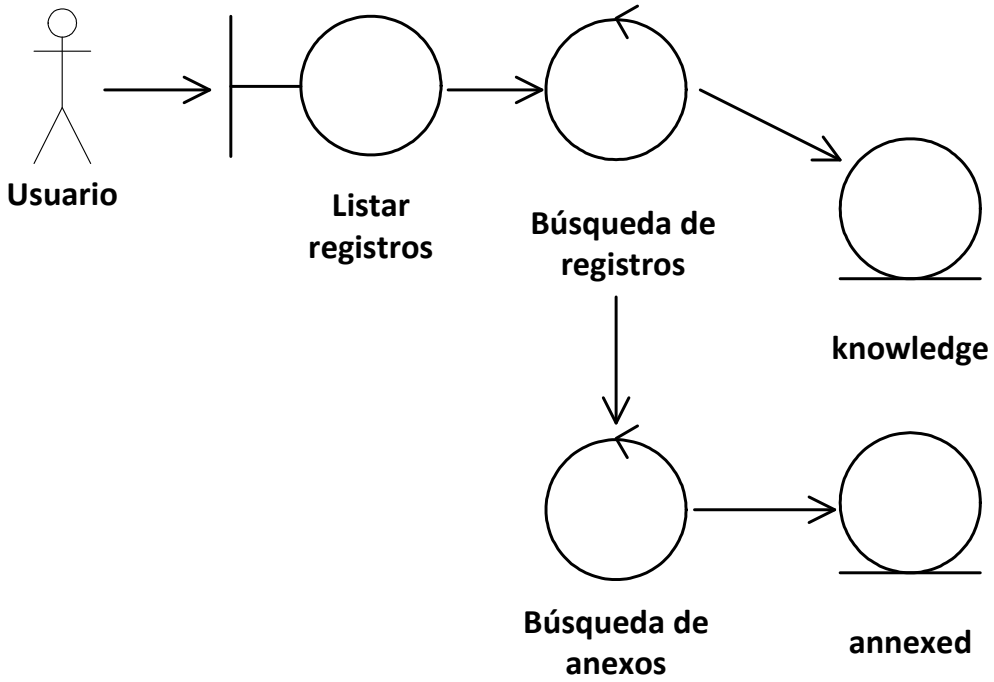


Figura 4.9
Diagrama de robustez del listado y búsqueda de registros de conocimiento

Cuando el usuario está identificado en el sistema es posible obtener un listado de todos los registros de conocimiento almacenados para posteriormente acceder a ellos. También es posible hacer una búsqueda particular de registros por clave.

En ambos casos, el funcionamiento es el mismo según lo muestra la Figura 4.9, donde la interacción con el usuario se da mediante la interfaz de listado de registros, la cual lanza las operaciones necesarias para localizar los registros de conocimiento que coincidan con el criterio de selección definido (todos o algunos registros), asimismo de forma paralela se realiza la búsqueda de los archivos anexos relacionados con los registros resultantes para poder operar sobre ellos (acceder, alterar, añadir o borrar). Cada objeto de control realiza las consultas en los repositorios correspondientes para que la respuesta al usuario sea de nueva cuenta por medio de la interfaz de resultados o listado de registros.

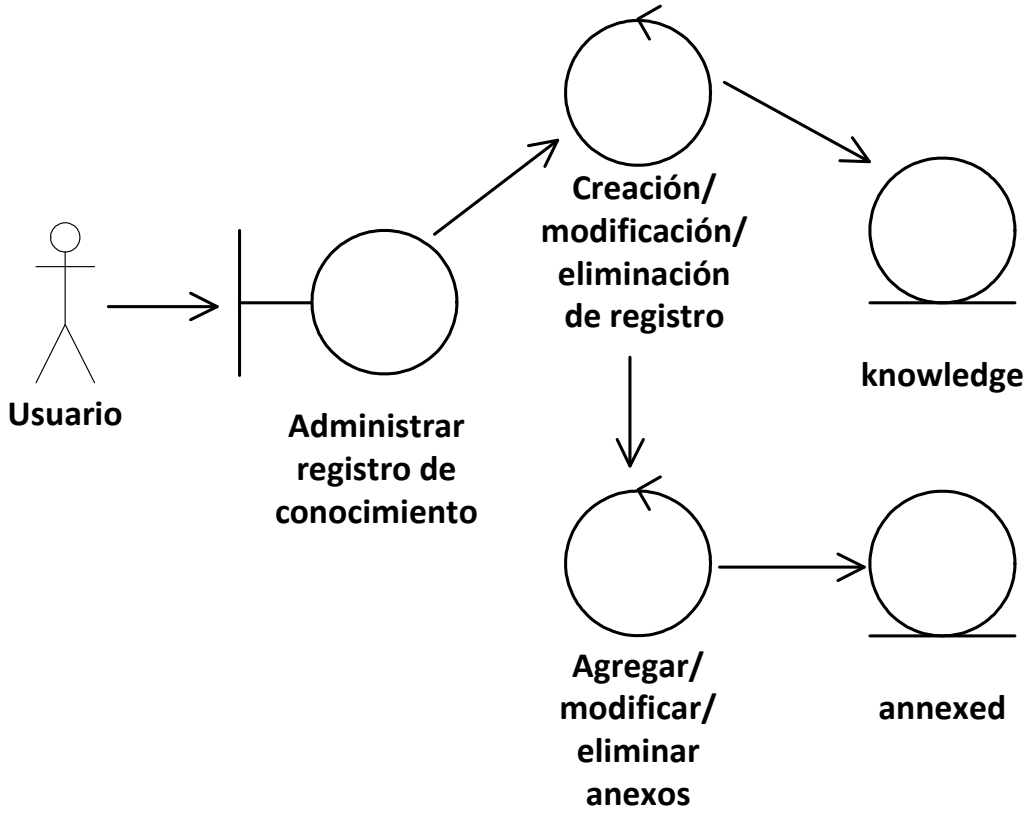


Figura 4.10
Diagrama de robustez de la administración de un registro de conocimiento

Administrar un registro de conocimiento lo ilustra la Figura 4.10 y significa que es posible hacer modificaciones en la información almacenada en el sistema. Esto aplica a todos los niveles de usuario excepto para el Lector, ya que solo se le permite acceder al registro en forma de consulta.

La interfaz de administración de registros es también la lista de registros mostrada en la Figura 4.9, donde se despliega en un solo contenedor la información almacenada dependiendo de un criterio de búsqueda particular. Siendo posible acceder a dicha información por medio de eventos del teclado o del mouse que lanzan el detalle de los registros de conocimiento para consultar sus datos generales y archivos anexos. Permitiendo su despliegue en línea y reproduciendo sus contenidos multimedia (audio, video y texto) si es que aplica.

Además de la información proporcionada por los diagramas anteriores, es importante destacar que el sistema de administración de conocimiento está preparado para reconocer archivos multimedia y abrirlos con el programa de reproducción instalado por defecto en la máquina cliente. El detalle de esto, las consideraciones técnicas y de operación se describen en la sección 4.8.

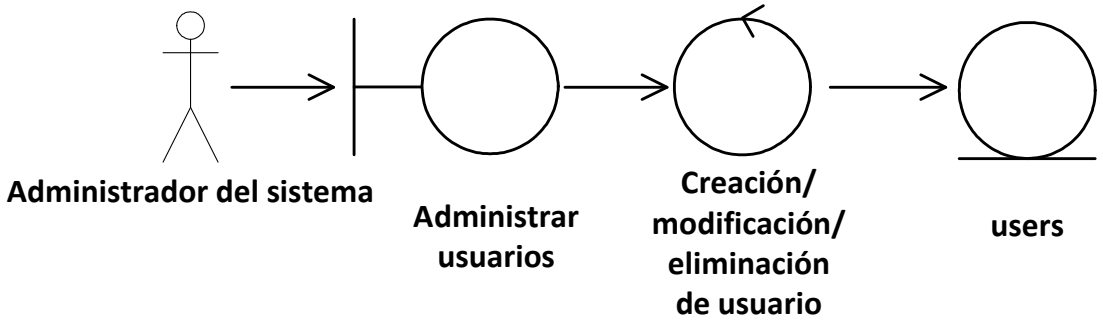


Figura 4.11
Diagrama de robustez de la administración de usuarios

La administración de usuarios es un módulo muy importante en la aplicación, pues en él se define quién y con qué permisos (rol de usuario) es posible entrar al sistema de administración de conocimiento. Los 3 roles de usuario disponibles permiten leer, administrar conocimiento o administrar el sistema.

La Figura 4.11 muestra que la interfaz de administración de usuarios es la vista principal para poder generar algún acceso al sistema o modificar una cuenta de usuario, el controlador de las operaciones es la entidad que hace las validaciones necesarias y se conecta directamente al repositorio de usuarios para efectuar las altas / bajas / cambios correspondientes.

## 4.8. Consideraciones importantes en el desarrollo del sistema y la reproducción de contenidos en el cliente

Si bien es necesario definir desde el principio una arquitectura de la aplicación a desarrollar, al momento de empezar a escribir el código fuente se experimentan algunas situaciones relacionadas con la interconexión entre los diferentes elementos de dicha arquitectura. Ya que OpenLaszlo trabaja sólo a nivel de presentación de la aplicación, esto permite que como motor del sistema podamos utilizar algún lenguaje de programación que dominemos para que controle la lógica de la aplicación, pero no la presentación.

Esto significa que la conectividad con la base de datos y las operaciones a realizar dentro de ella no son controladas en su totalidad por OpenLaszlo, sino por un desarrollo independiente que establezca dichas conexiones y entregue al servidor de Laszlo la información conforme la necesita, en este caso en formato XML.

Por lo anterior, en este sistema de administración de conocimiento además de desarrollar la interfaz de usuario en LZX, toda la lógica de la aplicación se codificó en JSP, aprovechando que se instala un servidor Tomcat para que se ejecute OpenLaszlo (Figura 4.7). Entonces las operaciones de búsqueda, inserción,

eliminación y modificación de registros en la base de datos son controladas directamente por JSP's y éstos los entregan en un formato específico para que OpenLaszlo muestre los resultados en pantalla dentro de la Rich Internet Application.

En los 4 diagramas de robustez mostrados en la sección 4.7, los objetos de control de información se refieren entonces a dichos archivos JSP que lanza OpenLaszlo y ellos establecen la conexión con la base de datos, realizan el comando o query específico y posteriormente generan un objeto XML que OpenLaszlo pueda interpretar.

Lo anterior se demuestra en el siguiente ejemplo, conformado por las 4 figuras siguientes:

```
. . .
<!-- Datasets con jsp's -->
        <dataset name="DStodosregs" type="http" src="getcontacts.jsp"
/>
. . .
```

Figura 4.12
Fragmento de código para definir un dataset (conjunto de datos), archivo km.lzx

La Figura 4.12 muestra cómo se define dentro de OpenLaszlo la llamada a un archivo externo, en este caso un JSP, para que el resultado de su ejecución se incorpore en un dataset o conjunto de datos que OpenLaszlo pueda interpretar.

```
. . .
    try {
        Class.forName("com.mysql.jdbc.Driver");
        connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/laszlo?user=
root&password=admin");
        Statement stmt = connection.createStatement();
        ResultSet rs = stmt.executeQuery("select * from knowledge");
        while (rs.next()) {
    %>
    <knowledge id="<%= rs.getString("id")%>"
        item="<%= rs.getString("item")%>"
        details="<%= rs.getString("details")%>"
        general="<%= rs.getString("general")%>"
        concrete="<%= rs.getString("concrete")%>"/>
    }
. . .
```

Figura 4.13
Fragmento de código JSP para conectarse a MySQL y convertir a formato XML, archivo getcontacts.jsp

La Figura 4.13 representa de una forma general el contenido de un archivo JSP donde se declara la conexión a la base de datos MySQL y se ejecuta un query determinado para posteriormente definir un objeto XML con etiquetas específicas.

```
<xml_data>
  <knowledge id="226" item="Tópicos de manufactura" details="Lista de
temas relacionados a la manufactura" general="Desarrollo de
productos" concrete="Automatización" />
  <knowledge id="227" item="RV10" details="Informacion general del
proyecto RV10" general="Fotografias del RV10" concrete="Dimensiones,
fotos, etc" />
</xml_data>
```

Figura 4.14
Contenido del objeto generado por el archivo getcontacts.jsp

La ejecución del código mostrado en la Figura 4.13 da como resultado la creación de un objeto con formato XML (Figura 4.14) en donde sus etiquetas contienen la información resuelta por el query antes mencionado.

```
. . .
<mygrid datapath="DStodosregs:/xml_data/" shownitems="7"
        showhlines="true" contentdatapath="knowledge">

        <gridtext width="50" datapath="@id" editable="false"
                sortable="true" resizable="false">
                ID
        </gridtext>
        <gridtext width="${(canvas.width / 5) - 10}" datapath="@item"
                editable="false" sortable="true" resizable="false">
                Item
        </gridtext>
        <gridtext width="${(canvas.width / 5) - 10}"
                datapath="@details" editable="false" sortable="true"
                resizable="false">
                Descripción
        </gridtext>
. . .
```

Figura 4.15
Fragmento de código para interpretar el objeto XML en OpenLaszlo, archivo km.lzx

Hasta este punto ya se tiene la información relacionada con la búsqueda de información y en formato XML, ahora es necesario que OpenLaszlo la identifique directamente dentro del objeto dataset (conjunto de datos) que se definió en la Figura

4.12 para que le sea posible tomar los datos y los pueda presentar en la aplicación. Lo anterior es ilustrado en la Figura 4.15.

En cuanto a los requisitos mínimos del usuario que accederá al sistema de administración de conocimiento son estrictamente necesarios los siguientes:

- Internet Explorer versión 7 o superior.
- Adobe Flash Player 9 o superior.
- Windows Media Player 11 o superior.

Las versiones de estos programas instalados en el cliente garantizan el correcto funcionamiento de la aplicación y un despliegue adecuado de la mayoría de los recursos multimedia que se puedan almacenar en él. Es importante mencionar que estas aplicaciones son gratuitas y por lo regular se encuentran disponibles en las instalaciones de escritorio que soportan el sistema operativo Windows XP o superior.

Además, como se ha mencionado a lo largo de este documento, la gran diversidad de información que se genera en las organizaciones, hace muy complicado que se estandarice un formato para almacenar archivos. Por lo que este sistema de administración de conocimiento tiene un módulo muy particular que detecta según la extensión del archivo, si es posible reproducirlo directamente en el navegador Web, por ser archivo multimedia, o en caso contrario, genera una liga directa para abrir el archivo siempre y cuando la aplicación necesaria esté instalada en el equipo que se está utilizando. Es decir, si tratamos de acceder a un archivo con extensión:

- Mp3
- Wav
- Avi
- Mpg
- Wmv

} Archivos de audio y video

- Png
- Jpg
- Gif
- Bmp

} Archivos de imágenes

La aplicación reproducirá dentro del propio navegador Web el contenido del archivo (si es de video o audio) o desplegará la imagen si es una fotografía. Y en caso de ser, por ejemplo un documento de Microsoft Word (incluido en la suite ofimática Office System [39]), la aplicación presentará una ventana donde se podrá acceder a dicho archivo para abrirlo directamente en el editor de documentos que tengamos instalado por defecto y que reconozca la extensión .doc (generalmente utilizada).

Es importante resaltar que mientras mayor sea la cantidad de programas instalados en el cliente (relacionados con el giro de la organización), será muy

probable que se puedan acceder los archivos almacenados en el sistema de administración de conocimiento, facilitando con ello su utilización. Esto a su vez significa que el sistema es dependiente de tener instalados programas específicos para poder visualizar los anexos. Por ejemplo, si se agrega al sistema un archivo con la extensión PDF (Portable Document Format) y se necesita abrirlo para su consulta, entonces es requisito tener instalado un visor de este tipo de archivos, como puede ser Adobe Reader [40].

## 4.9. Resumen del capítulo

El presente capítulo abordó con detalle los aspectos tecnológicos y de diseño de software involucrados en un sistema de administración de conocimiento basado en las tecnologías descritas, resaltando en su arquitectura los roles de cada componente y sobre todo, especificando la relación e interconexión de las herramientas utilizadas para el desarrollo.

Las funcionalidades del sistema fueron ilustradas por medio de diagramas diversos que facilitan su comprensión y complementan el detalle de los procedimientos descritos.

Al final, se explicó el modo de trasladar la información desde OpenLaszlo hacia componentes externos de la aplicación que trabajan detrás de ella para acceder a la base de conocimiento y manipular los datos de tal manera que sea transparente para el usuario el despliegue de los contenidos de conocimiento.

Habiendo comprendido la forma en la que está estructurado el sistema de administración de conocimiento, el capítulo siguiente describe en forma detallada las funcionalidades implementadas en un caso de estudio particular.

# Capítulo 5. Caso de estudio

En este capítulo se muestra el caso de estudio con el cual se alimentó el sistema de administración de conocimiento desarrollado. Mismo que no fue implementado para sustituir a ninguno de los sistemas existentes, el propósito es demostrar su aplicación en un proyecto activo con información real.

A continuación se aborda el proyecto en el cual se valida la funcionalidad descrita para un sistema de administración de conocimiento, utilizando la infraestructura detallada en el capitulo anterior y delimitando sus alcances, todo lo anterior empleando datos generados en el transcurso de un proyecto de ensamble.

## 5.1. Proyecto RV-10

El Centro de Innovación en Diseño y Tecnología (CIDyT) del Tecnológico de Monterrey Campus Monterrey, la Cátedra de Ingeniería Automotriz y el claustro de profesores de ésta apoyan un proyecto titulado "RV-10 Ensamble de un avión", éste da inicio en Enero del 2007 y tiene una duración proyectada de 2 años y medio [41].

Dicho proyecto consiste en el ensamble de un avión tamaño real tipo RV-10 de tren fijo con capacidad de carga para 4 pasajeros y un motor de hasta 260 caballos de fuerza (HP). Un ejemplo de este tipo de aeronave y sus partes principales se ilustran más adelante en la Figura 5.1.

Mediante la experiencia directa y práctica en la construcción del aeroplano se aprenden técnicas, métodos y procedimientos de ensamble, diseño, manufactura, control e instrumentación de vehículos aéreos.

Durante todo el proceso se generan modelos completos y detallados de las partes que componen los sub-ensambles del avión para posteriormente simular tanto su comportamiento como el proceso de ensamble con el fin de automatizarlo.

En el semestre Enero - Mayo 2007 se terminó el ensamble del empenaje o cola del avión (Figura 5.1) y se comenzó el ensamble de las alas (cuya terminación se planeó para el semestre Agosto-Diciembre del mismo año), todo esto involucró la capacitación correspondiente del grupo de trabajo, apoyado también por alumnos estudiantes del área de manufactura. Todo lo anterior está a cargo de 2 directores de proyecto, 2 coordinadores y 1 experto en aeronáutica. Dada la naturaleza del proyecto y el alcance que este tiene, requiere que se lleve a cabo la documentación del proceso conforme al estándar aeroespacial AS9100B, para lo cual también el grupo de trabajo se capacitó con el objeto de cubrir el requisito.

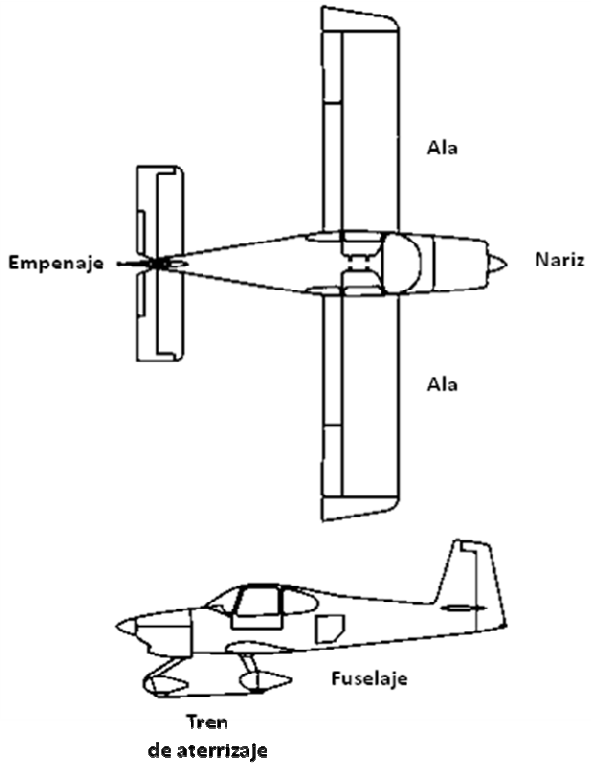


Figura 5.1
Aeronave modelo RV-10 y partes principales
(Adaptado de [42])

### 5.1.1. Objetivos y áreas de desarrollo

El objetivo general del proyecto consiste en desarrollar técnicas, métodos y procedimientos para ensamblar, diseñar y manufacturar control e instrumentación de aeronaves, en un contexto de modelado computacional de metales sólidos y hojas metálicas, modelado analítico y numérico de cinética, además de modelar ambientes reales usando ingeniería reversa.

Esto significa que se generará gran cantidad de conocimiento a partir de la práctica del ensamble del aeroplano antes descrito para crear modelos seguros, completos y detallados que describan su comportamiento, así como sus partes y subsistemas.

Los objetivos específicos a cubrir durante el año 2007 son:

- Ensamble del empenaje o cola del avión y ensamble de las alas.
    - Asistencia a capacitaciones de ensamble.
    - Asistencia y cumplimiento en capacitaciones de aeronáutica.

- Asistencia a capacitación de dinámica de vuelo.

- Documentación del proceso aplicando el estándar AS9100B.
  - Asistencia a capacitación del estándar AS9100B.

- Modelación y simulación de piezas y ensambles en CAD (CATIA-DELMIA).
  - Identificar el sub-ensamble.
  - Obtener las dimensiones de las partes que lo componen.
  - Realizar los modelos en CAD.
  - Usar herramientas PLM para el sub-ensamble.
  - Simulación del ensamble del empenaje o cola.
  - Simulación del ensamble de las alas.

- Automatización del proceso de ensamble.
  - Investigación de ideas para la automatización.
  - Investigación de posibles patentes existentes.
  - Proponer un proceso de ensamble de bajo costo para el sub-ensamble.
  - Asegurar la repetitividad identificando insumos de calidad.
  - Determinar puntos críticos en el ensamble que ocasionen defectos, para disminuir el rechazo.

- Portafolio de evidencias. Cada equipo de trabajo integrado por los alumnos es responsable de evidenciar las tareas involucradas en el desarrollo, implementación y seguimiento de un plan de actividades, las minutas de reuniones formales e informales, los reportes entregables, los análisis realizados, especificaciones, memorias de cálculo, etc.

- Investigación.
  - Ingeniería en Reversa y su aplicación en el avión a ensamblar.
  - Troquelado de esfuerzos y diseño.
  - Análisis de fluidos y su aplicación en el avión a ensamblar.

Las áreas de desarrollo involucradas con los objetivos anteriores son:

- Ingeniería electrónica. Donde se prueban, ajustan y modifican prototipos de un conjunto básico de instrumentos de un aeroplano RV-10 para proponer mejoras en el desempeño; además de utilizar e implementar la suite de CATIA llamada "Electrical wire routing" o ruteo de cableado eléctrico.
- Ingeniería industrial. Donde se investigan los procesos de mecanizado para desarrollar componentes y partes; se simulan flujos de trabajo para mejorar el desempeño en la planta de ensamble; se investigan y certifican procesos de ensamble para generar documentación respetando estándares.

- Ingeniería mecánica. Donde se analizan los fluidos para obtener su comportamiento; se estudian modelos CAD / CAE / CAM simulando ensambles que identifiquen áreas de oportunidad en el ensamble físico; se investigan y analizan los perfiles usados en la industria aeronáutica; se analizan elementos finitos para probar el diseño óptimo y sugerir mejoras; y se investigan e identifican los materiales empleados en las aeronaves para utilizarlos o proponer sustitutos.
- Diseño industrial. Donde se estudia y analiza la interacción humano – máquina y la aplicación de la electrónica a la aviación.
- Al mismo tiempo, también se relacionan los temas: aerodinámica, análisis estructural, instrumentación, diseño conceptual, manufactura digital (PLM) y ensamble.

Todo lo anterior busca que el proyecto esté bien documentado para que con el paso de las etapas sea posible retomarlo sin que esto signifique capacitación especializada para las nuevas personas involucradas.

## 5.2. Funcionalidades implementadas en el sistema

En esta sección se explica el funcionamiento del sistema de administración de conocimiento con la información del proyecto RV-10 que se cargó en él, resumiendo en acciones concretas de utilización e ilustrando con figuras cómo es posible operarlo para acceder, generar y modificar conocimiento, además de demostrar los niveles de usuario que se consideraron para convertirlo en una herramienta útil en Web.

### 5.2.1. Login o identificación de usuario

En la Figura 5.2 que se muestra más adelante, se puede observar que el acceso al sistema de administración de conocimiento se da mediante la introducción de una URL al navegador Web y después de cargarse la Rich Internet Application, se despliega la pantalla de inicio o página de login, donde es necesario identificarse como un usuario válido en el sistema.

Tomemos el caso del Administrador del sistema, que ingresa su usuario y contraseña en la página de login. Cuando presiona el botón de Iniciar Sesión, el sistema verifica que los datos introducidos sean correctos para poderle permitir el acceso a las funciones que su tipo de usuario puede acceder.

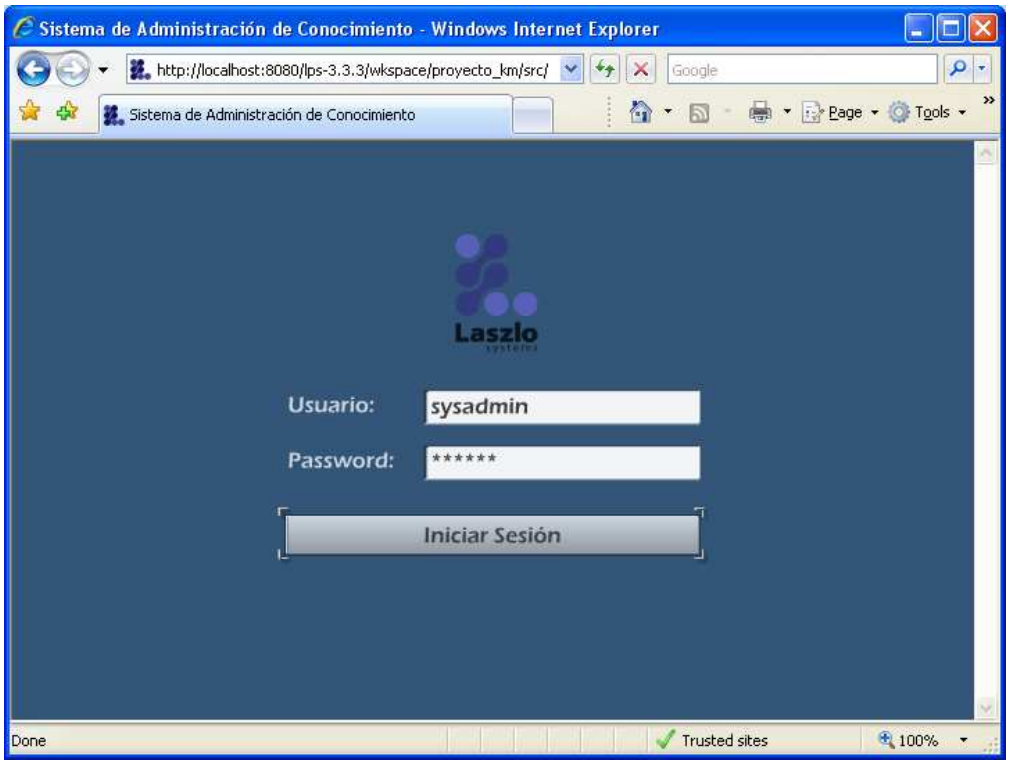


Figura 5.2
Ejecución del sistema y despliegue de la página de identificación de usuario

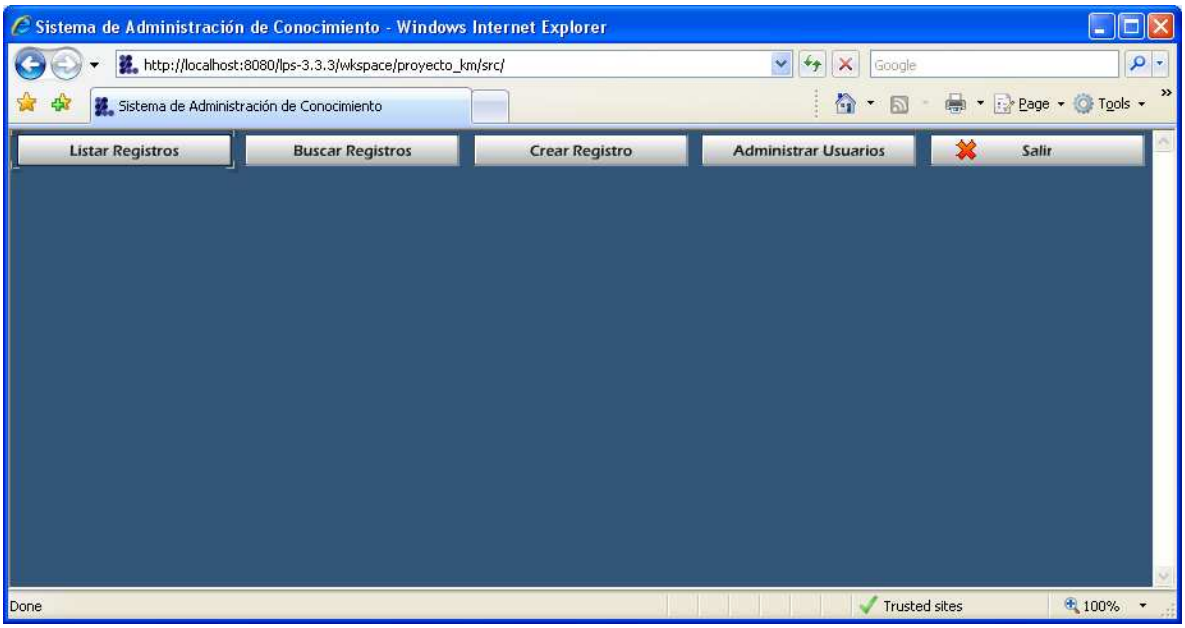


Figura 5.3
Funciones disponibles para el Administrador del sistema

La Figura 5.3 ilustra las funciones que el Administrador del sistema puede realizar en el transcurso de su sesión en la Rich Internet Application por medio de botones en la parte superior de la pantalla.

Además, cuando se ingresa información inválida en el sistema para tratar de acceder al repositorio de conocimiento (usuario o password incorrectos, información sensible a mayúsculas y minúsculas), se niega el ingreso al usuario, como se muestra en la Figura 5.4.

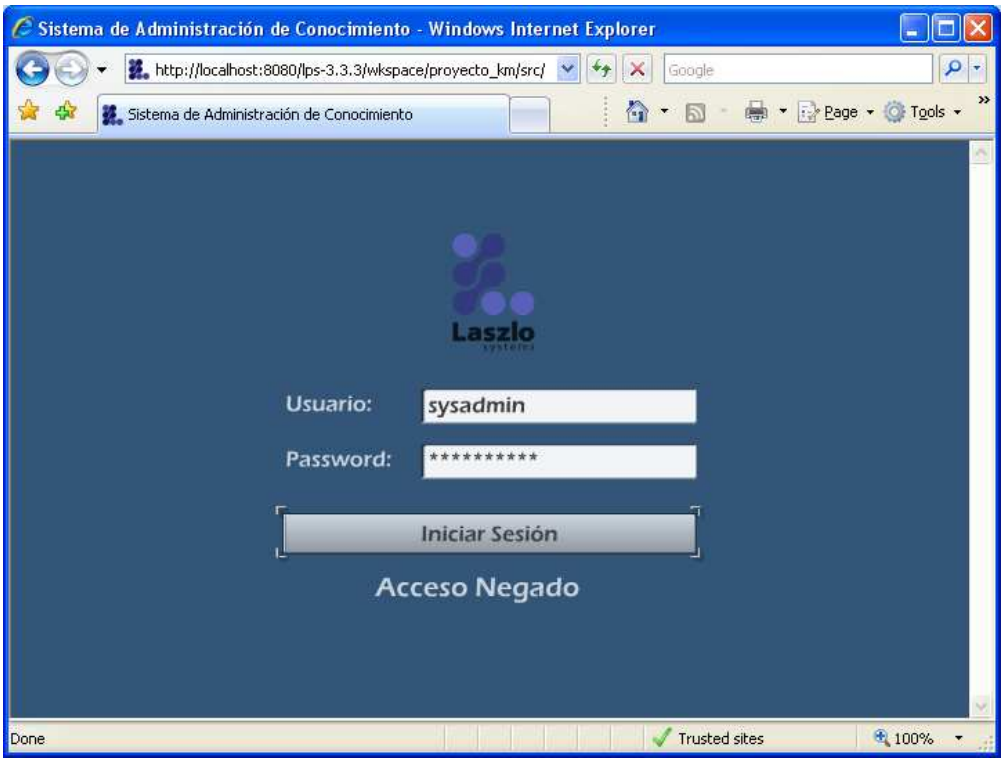


Figura 5.4
Negativa de acceso al introducir datos inválidos

### 5.2.2. Listar registros de conocimiento

De las funciones disponibles para cualquier tipo de usuario se encuentra el listado de registros, que despliega en una sola pantalla toda la información almacenada en el sistema, mostrando sólo algunos datos relevantes por cada registro de conocimiento. La finalidad es que el usuario tenga de primera mano una vista general de ellos y pueda acceder a más detalle seleccionando el que necesite.

Cabe mencionar que para el usuario de tipo Lector solo le será permitido en esa pantalla hacer consultas. Pero al resto de los usuarios le desplegará a un lado de cada registro las opciones de Borrar o Anexar, que se describen en la sección 5.2.5.

La Figura 5.5 ilustra el resultado de elegir el listado de registros como un usuario de tipo Administrador del sistema.

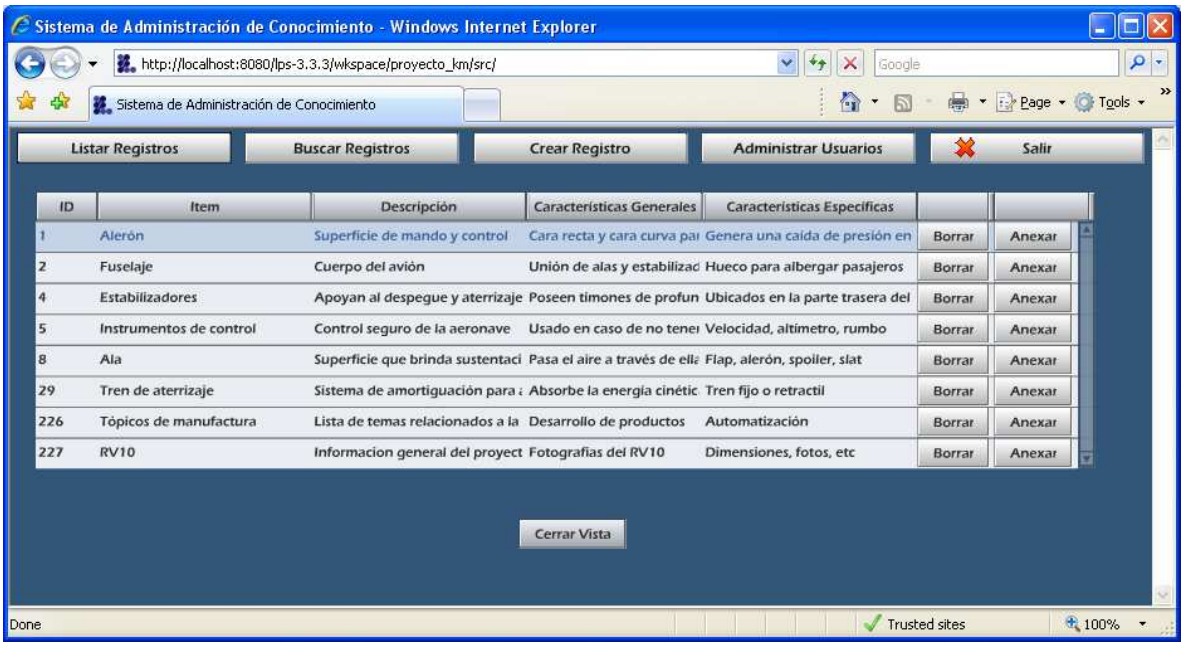


Figura 5.5
Listado de registros para el Administrador del sistema

La Figura 5.5 muestra cómo se despliegan los resultados en la aplicación por medio de una tabla dinámica en donde es posible ordenar los registros dependiendo de encabezado que se seleccione. Es un funcionamiento similar al que ofrece una hoja de cálculo cuando se necesita ordenar numéricamente o alfabéticamente. Esta funcionalidad que brinda generalmente una aplicación de escritorio también lo hace una Rich Internet Application sin mayores contratiempos y sin recarga de información, lo cual es muy atractivo y útil para el usuario final.

### 5.2.3. Buscar registros de conocimiento

La búsqueda de registros también es una funcionalidad disponible para todos los tipos de usuario en donde el criterio de búsqueda de información se define por descripción del registro, esto es con la finalidad de poder localizar registros de conocimiento relacionados.

Inicialmente se elige la opción de buscar registros y se muestra una pantalla que solicita la palabra o frase que se desee ubicar en el sistema. Ésta no es sensible a mayúsculas o minúsculas, lo cual ofrece la ventaja de poder buscar información sin importar si la persona que agregó la información lo hizo empleando un formato específico. Posteriormente se inicia la búsqueda y en la misma pantalla se devuelve

el resultado por medio de una tabla dinámica con la misma funcionalidad explicada en la sección anterior.

De igual forma, el Lector solo puede acceder al registro de conocimiento, mientras que el resto de los tipos de usuario pueden Borrar o Anexar información.

Todo lo anterior se resume en la Figura 5.6.

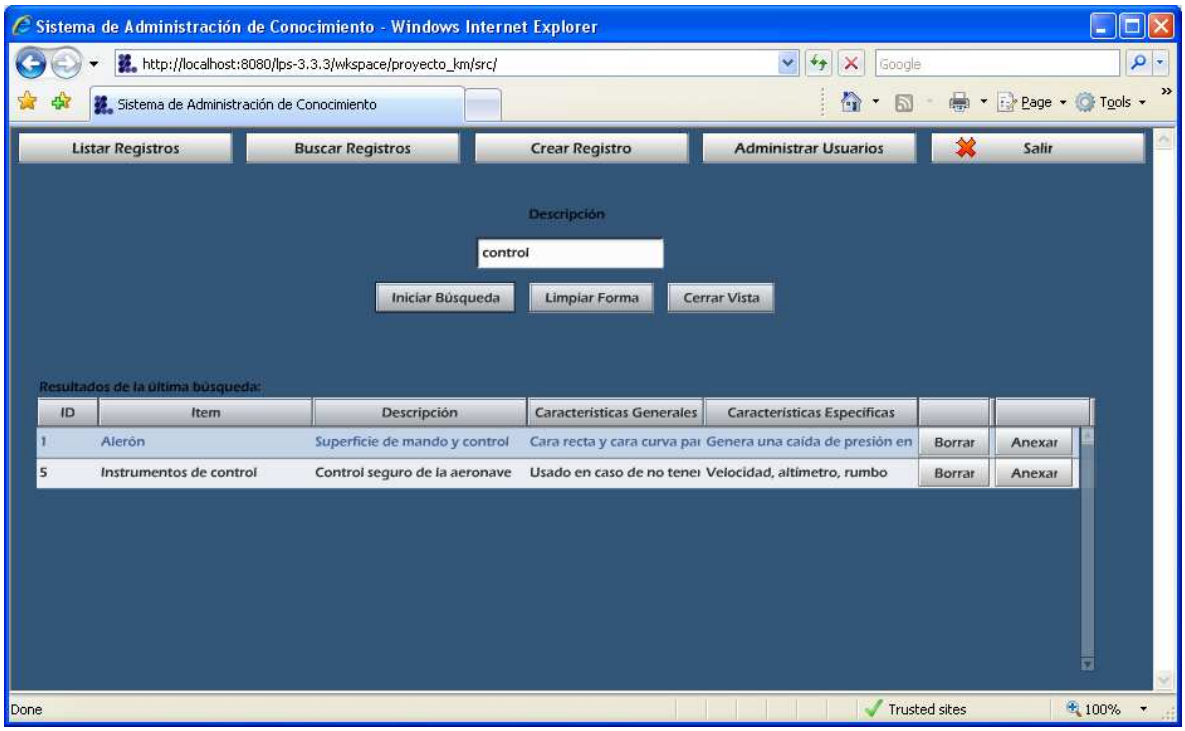


Figura 5.6
Búsqueda de registros para el Administrador del sistema

### 5.2.4. Crear registro de conocimiento

El módulo de creación de registro de conocimiento permite dar de alta un tópico en el sistema, en donde se solicita información textual para ser almacenada en la base de datos y también es posible agregar archivos anexos relacionados con ese registro en particular, haciéndolo dependiente de él.

Los tipos de archivo que se pueden almacenar no tienen restricción, pues la función del sistema es recopilar toda la información relevante para la empresa acerca de lo que consideren útil y pertinente a ser almacenado y distribuido.

La Figura 5.7 nos muestra la pantalla de creación de registro.

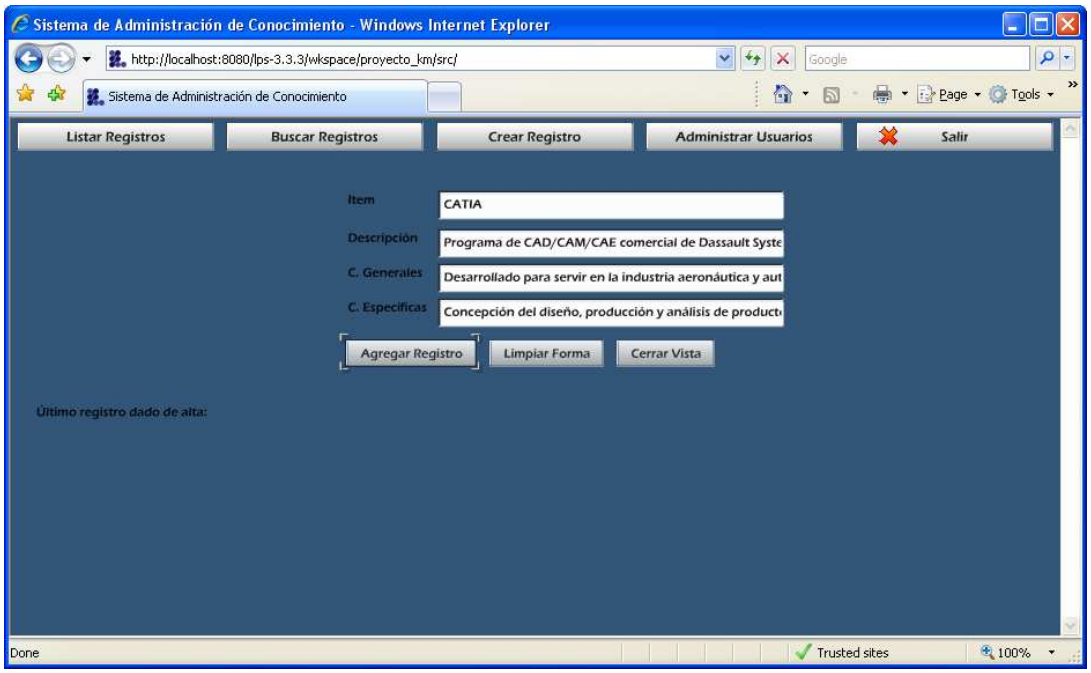


Figura 5.7
Creando un registro de conocimiento como Administrador del sistema

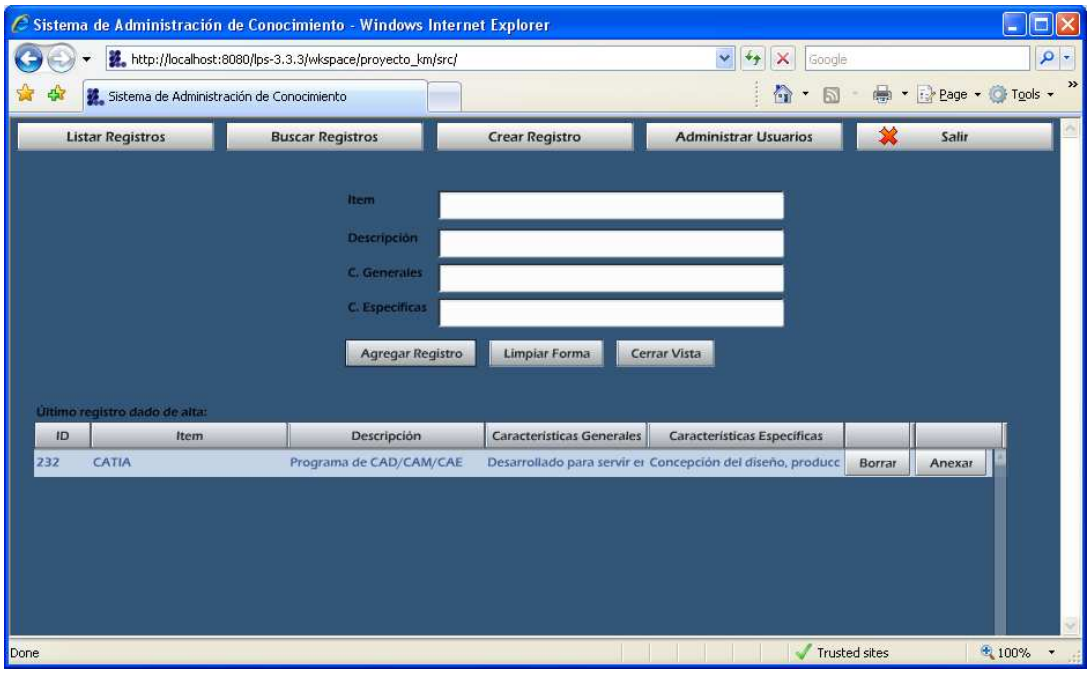


Figura 5.8
Registro de conocimiento creado como Administrador del sistema

La Figura 5.8 muestra la respuesta del sistema cuando se agrega el registro de conocimiento, permitiendo de forma inmediata la modificación del mismo cuando se selecciona. También en posible anexarle un archivo directamente a ese registro o definitivamente eliminarlo del repositorio de conocimiento.

Es importante mencionar que la creación de registros de conocimiento sólo está disponible para los niveles administrativos (de conocimiento y de sistema).

Debido a que el procedimiento de modificación de registro incluye al de agregar archivos anexos, no se explica en esta sección, pero esto se detalla en la sección 5.2.5.

### 5.2.5. Modificar registro de conocimiento

La relevancia del sistema de administración de conocimiento radica en mantener actualizada la información contenida en él, por eso es muy importante la modificación del registro, misma que no está definida con un botón específico dentro del sistema, sino que sólo está disponible para los usuarios administrativos cuando tratan de acceder al detalle de un registro en particular.

Esta operación solamente puede darse cuando se ha realizado una búsqueda específica de un registro o se ha hecho un listado de todos los registros de conocimiento y se ha efectuado la consulta mediante la función de doble clic en él para mostrar su detalle y archivos anexos (si es que los tiene).

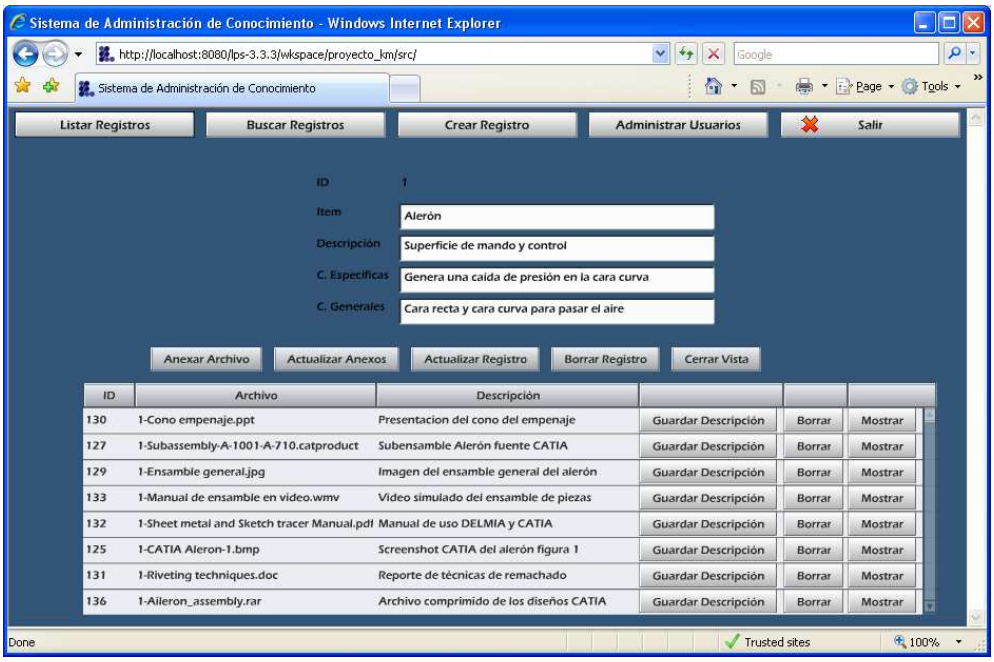


Figura 5.9
Registro de conocimiento consultado como Administrador del sistema

La Figura 5.9 muestra el detalle de un registro de conocimiento incluyendo la lista de archivos anexos asociados a él. Como puede observarse, hay botones en medio de la tabla dinámica que permiten hacer operaciones sobre el registro.

Para anexar un archivo, basta con seleccionar el botón correspondiente y se abrirá una nueva ventana de diálogo que especifica la ruta donde se encuentra el recurso a guardar (documento, imagen, video, etc.) Esto se muestra en la figura 5.10.


Figura 5.10
Cuadro de diálogo para Anexar un archivo

Al seleccionar el botón Anexar de esa misma pantalla se establece la conexión con el servidor de aplicación y se almacena dicho archivo en el sistema de administración de conocimiento, quedando ligado ese anexo con el registro seleccionado. El formato del nombre con el que se almacena el archivo anexo es controlado de forma automática por el sistema. Un ejemplo de la respuesta del sistema tras esta operación es la que se muestra en la Figura 5.11.

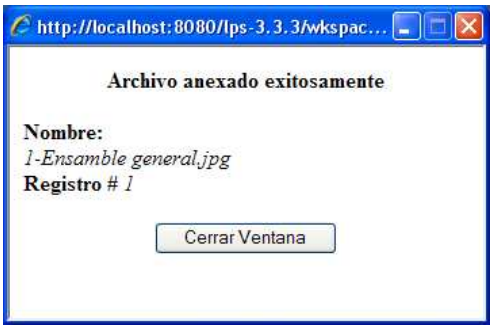


Figura 5.11
Confirmación de la operación Anexar archivo

Posteriormente, es necesario seleccionar el botón Actualizar anexos de la pantalla del detalle de registro para que la tabla dinámica recargue la información.

Este mismo procedimiento es el que se realiza cuando directamente, sin entrar al detalle de un registro, se quiere almacenar un archivo anexo desde la pantalla que resulta al listar todos los registros de conocimiento o al buscar alguno en concreto (Secciones 5.2.2 y 5.2.3).

La descripción del archivo anexo se almacena introduciendo de forma directa el texto en el campo correspondiente dentro de la tabla dinámica y seleccionando el botón de ese mismo renglón titulado Guardar descripción. Esto hace que se efectúe la operación y se ilustra en la Figura 5.12.

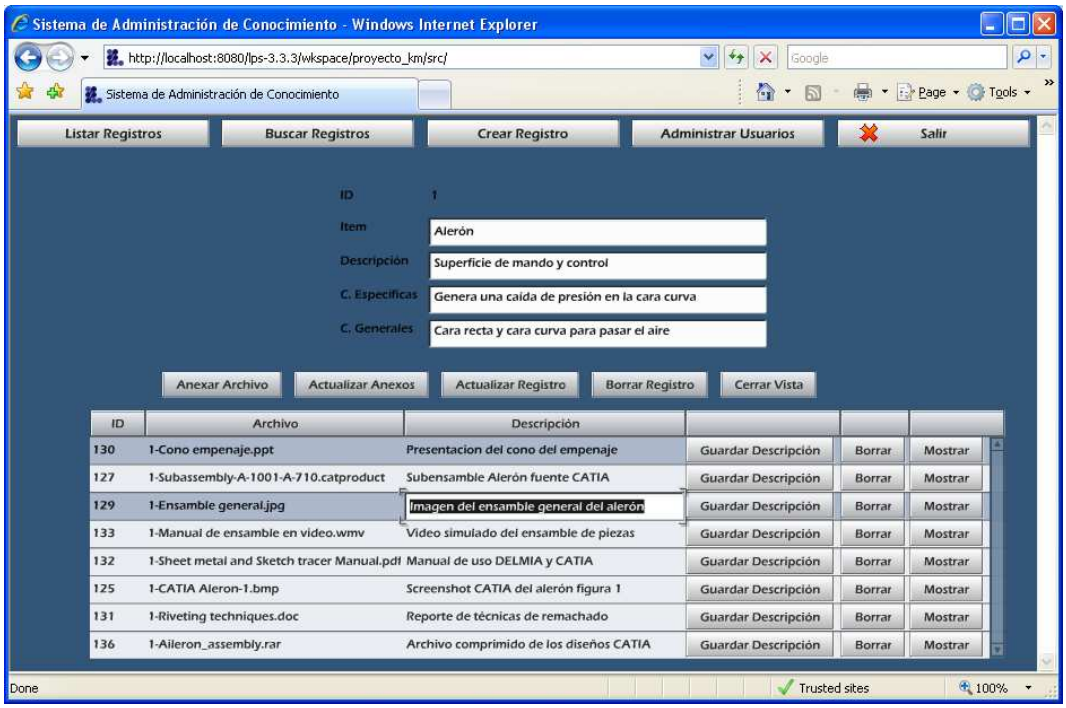


Figura 5.12
Actualizando la descripción de un archivo anexo como Administrador del sistema

La actualización de un registro respecto a la información capturada como texto (ítem, descripción, características generales y específicas) se realiza sobre escribiendo la información nueva en el campo correspondiente y sólo es necesario seleccionar el botón Actualizar Registro para que se almacenen los nuevos datos, como se ilustra en la Figura 5.13.

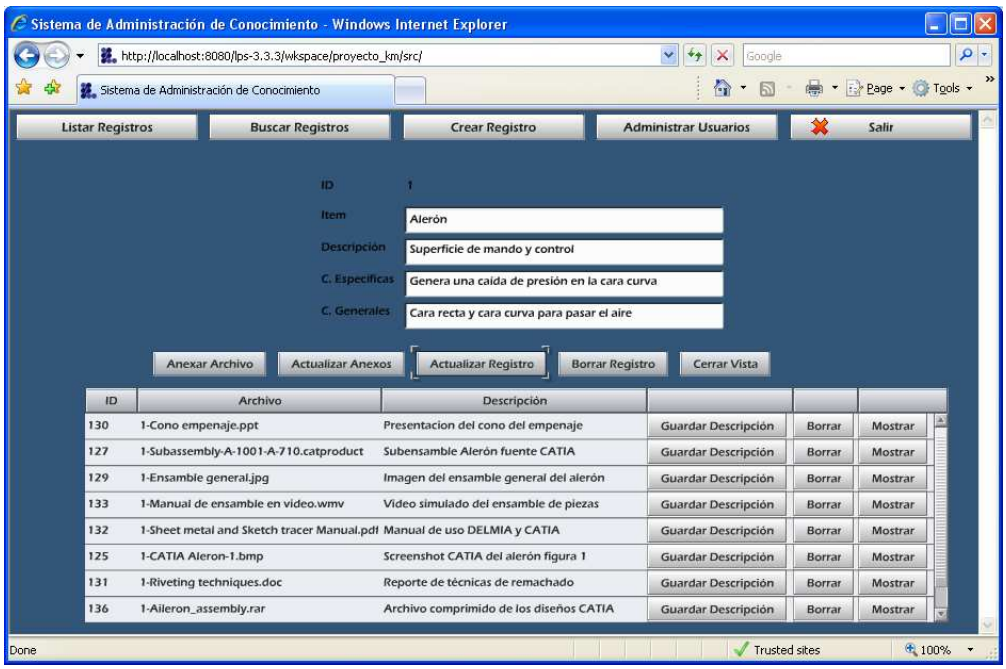


Figura 5.13
Actualizando un registro de conocimiento como Administrador del sistema

El borrado del registro de conocimiento significa que se elimina toda la información relacionada en él, incluyendo los archivos anexos. Esta operación se efectúa haciendo clic en el botón Borrar Registro. Dicha opción es la misma que se encuentra disponible en la tabla dinámica cuando se listan todos los registros del sistema o se busca un registro particular. (Secciones 5.2.2 y 5.2.3)

Un ejemplo de la pantalla de salida cuando se borra un registro de conocimiento es la que se muestra en la Figura 5.14.


Figura 5.14
Confirmación de la operación Borrar Registro y Borrar Anexo

El borrado de anexos de un registro de conocimiento permite eliminar solo un archivo a la vez, lo cual puede aplicarse a los anexos obsoletos y considerados innecesarios o irrelevantes. Esta operación se realiza seleccionando el botón Borrar que aparece en el renglón correspondiente al archivo anexo en cuestión. La pantalla de salida cuando se ejecuta esta acción es la misma que se muestra en la Figura 5.14.

### 5.2.6. Reproducción de archivos anexos

La reproducción de los archivos anexos de un registro de conocimiento se realiza seleccionando el botón Mostrar que aparece en cada anexo disponible cuando se está en la pantalla del detalle de un registro específico. Es necesario hacer hincapié en que esta operación está disponible para todos los niveles de usuario, ya que solo es una operación de consulta.

Esto permite que dependiendo el tipo de archivo se muestre de forma automática su contenido (según los criterios explicados en la sección 4.8) o se muestre un vínculo con el cual sea posible descargarlo en el equipo en uso para abrirlo posteriormente con software específico.

Por ejemplo, basándonos en la Figura 5.9 y eligiendo un archivo de video, el sistema lo reproduce automáticamente, incluyendo la descripción almacenada, tal como lo muestra la Figura 5.15.

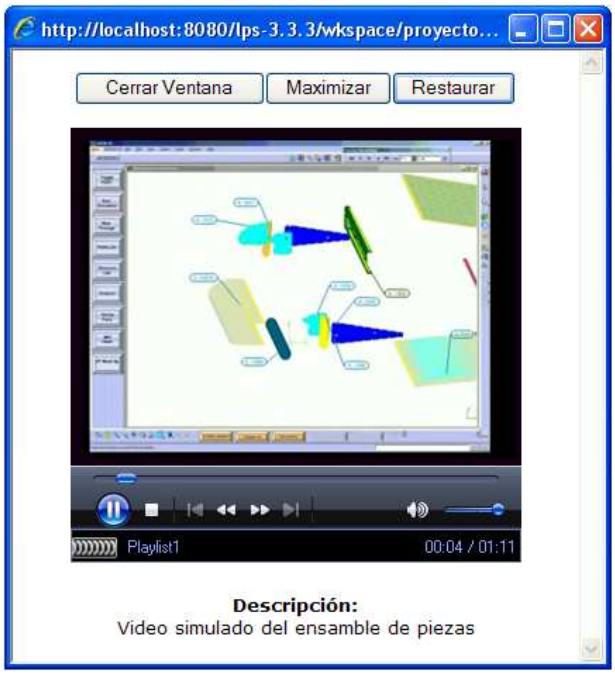


Figura 5.15
Reproducción de un archivo de video dentro del sistema

La figura anterior muestra cómo se reproduce un archivo de video dentro del sistema de administración de conocimiento y también permite regular la reproducción gracias a la integración de los controles de video de la aplicación instalada por defecto (Windows Media Player). Las dimensiones de la ventana de reproducción también pueden ser controladas con los botones que aparecen en la parte superior del video.

La reproducción de los archivos de audio es exactamente igual al proceso explicado previamente.

Cuando se intenta acceder a un archivo de imagen (Sección 4.8), entonces el sistema reconoce el tipo de archivo y lo despliega como se ilustra en la Figura 5.16.

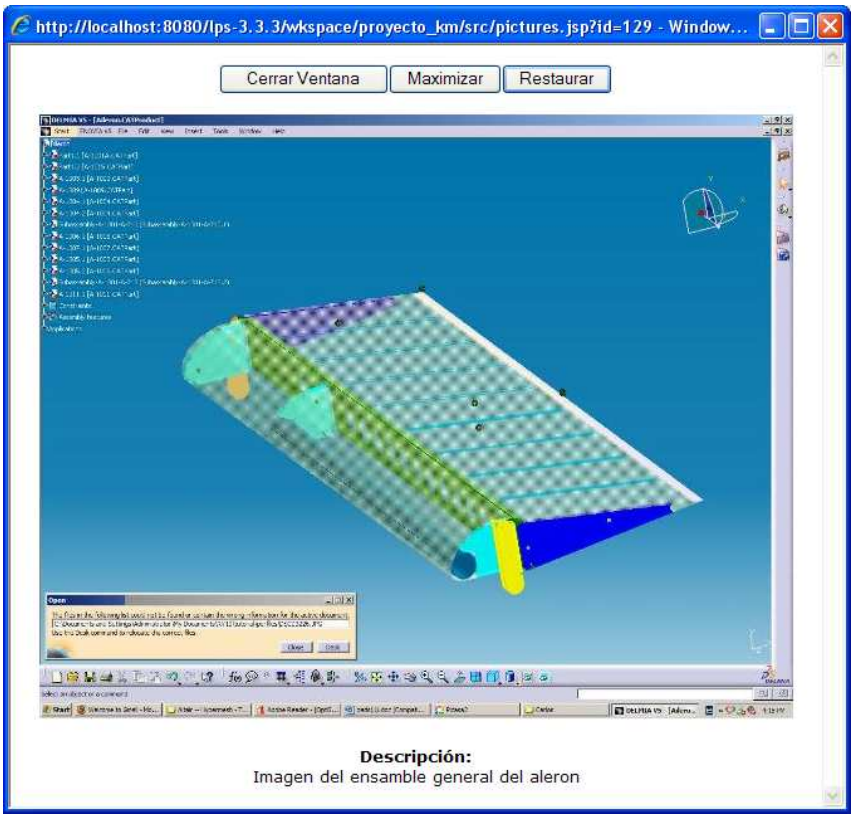


Figura 5.16
Reproducción de un archivo de imagen dentro del sistema

En la Figura 5.16 podemos observar que al tratarse de un objeto de contenido fijo (imagen), no hay controles disponibles, salvo los que actúan sobre las dimensiones de la ventana (Cerrar, Maximizar y Restaurar).

En el caso de tratar de acceder a un tipo de archivo no reconocido por el sistema como reproducible de manera automática (Sección 4.8), entonces se muestra una pantalla de salida como la Figura 5.17.

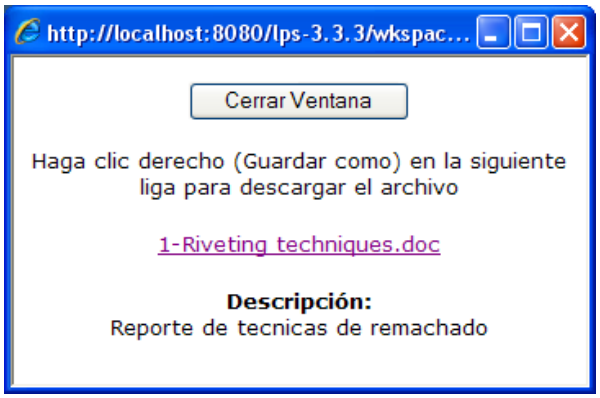


Figura 5.17
Acceso a un archivo no reproducible dentro del sistema

La Figura 5.17 demuestra que el sistema de administración de conocimiento al no reconocer el tipo de archivo seleccionado, genera una pantalla diferente para que el usuario pueda descargarlo y abrirlo con la aplicación necesaria. En este caso es un archivo con extensión .doc (documento de procesador de palabras) que puede ser consultado con Microsoft Word.

### 5.2.7. Administrar usuarios

El módulo de administración de usuarios sólo está disponible para el administrador del sistema y en él se puede agregar un nuevo usuario, modificar la información y permisos de uno existente o borrarlo definitivamente del sistema.

Para acceder a este módulo es necesario seleccionar el botón Administrar Usuarios y deberá mostrarse una tabla dinámica que contenga a todos los usuarios con permisos para ingresar al sistema de administración de conocimiento.

La Figura 5.18 que se muestra más adelante resume lo anterior.

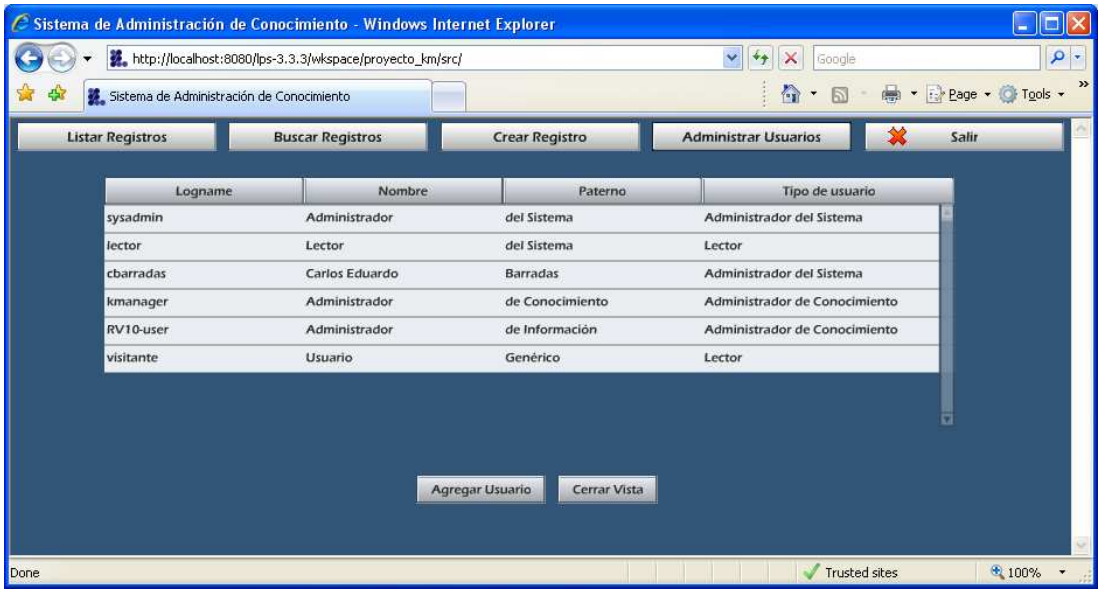


Figura 5.18
Módulo de administración de usuarios

Para la creación de un usuario nuevo, es necesario seleccionar el botón Agregar Usuario. Posteriormente se debe introducir la información requerida por el sistema para agregar a la persona indicada con el tipo de usuario seleccionado (Lector, Administrador del sistema o Administrador de conocimiento) como se muestra en la Figura 5.19.

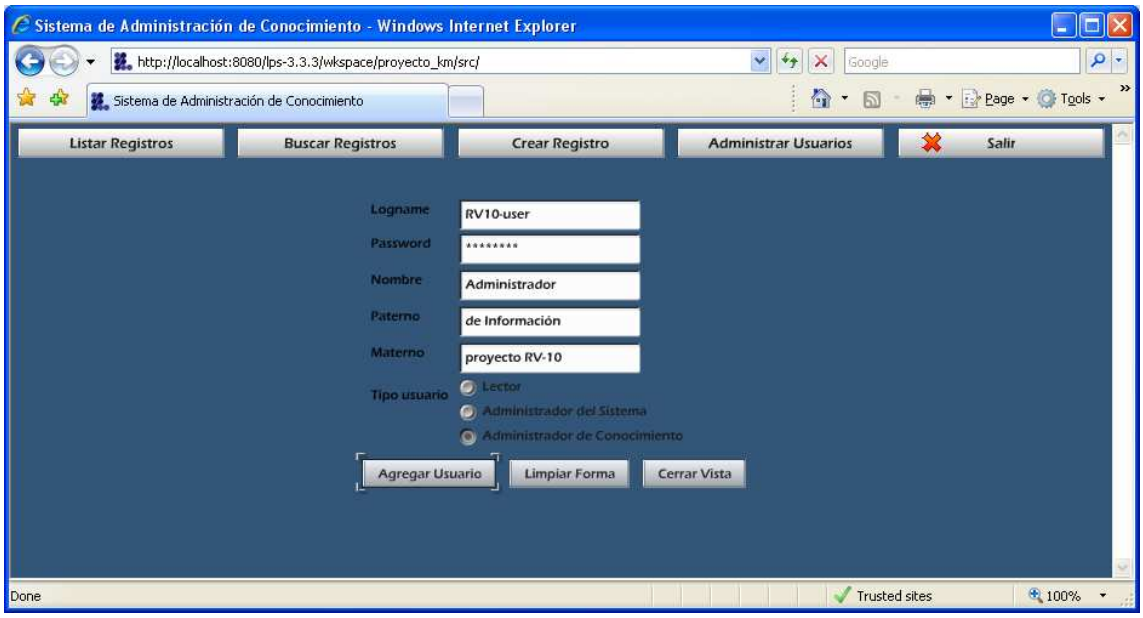


Figura 5.19
Creación de un usuario para que acceda al sistema

Al seleccionar el botón Agregar Usuario, se genera el acceso en el sistema y a partir de ese momento ya es posible ingresar autentificándose con dichas credenciales.

Para modificar la información de un usuario existente o borrarlo es necesario hacer doble clic sobre él dentro de la tabla dinámica y posteriormente se abre el detalle de su registro, como se muestra en la Figura 5.20.

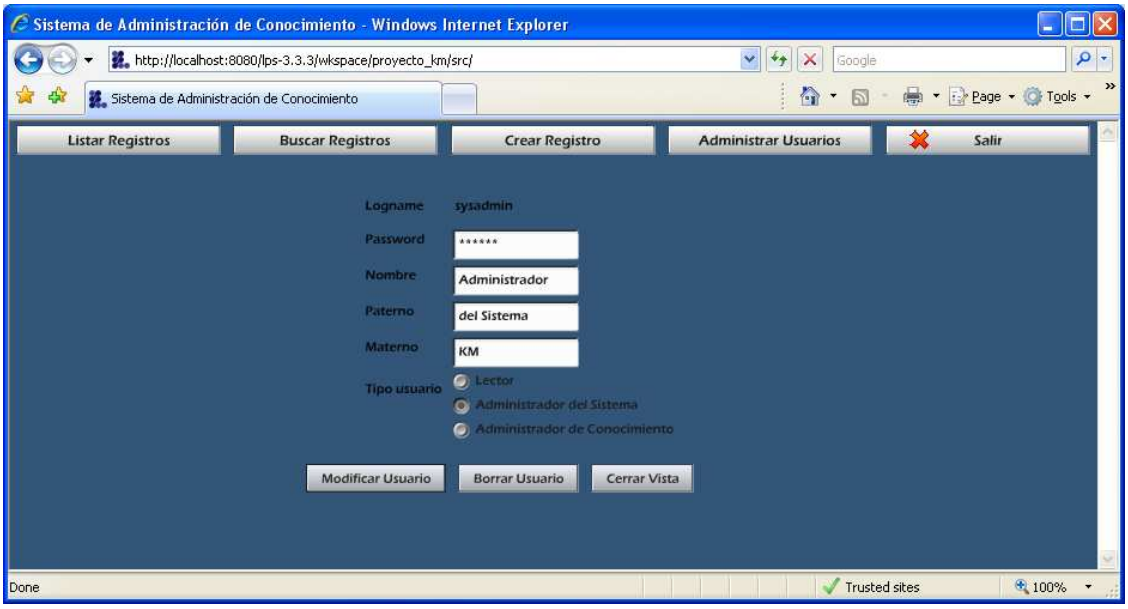


Figura 5.20
Modificación de información de un usuario en el sistema

La modificación de los datos del usuario también consiste en sobre escribir la información necesaria en los cuadros de texto correspondientes, inclusive, es posible cambiar el nivel de permisos de acceso. Los cambios se realizan al momento de seleccionar el botón Modificar Usuario, que se ilustra también en la Figura 5.20.

Para el borrado de un usuario del sistema, el procedimiento consiste en abrir el detalle del usuario dentro de la tabla dinámica y posteriormente seleccionar el botón Borrar Usuario.

Una pantalla de salida parecida a la que se muestra en la Figura 5.21 es la que se obtiene como respuesta de esta operación.


Figura 5.21
Confirmación de la operación Borrar Usuario

## 5.3. Comparativa de las pantallas mostradas dependiendo el tipo de usuario que accede al sistema

En los ejemplos anteriores se mostró la funcionalidad completa del sistema utilizando al usuario Administrador del sistema, quien no tiene restricciones de uso en la aplicación. Por lo que en esta sección se muestran las diferentes opciones disponibles para el Administrador de conocimiento y para el Lector.

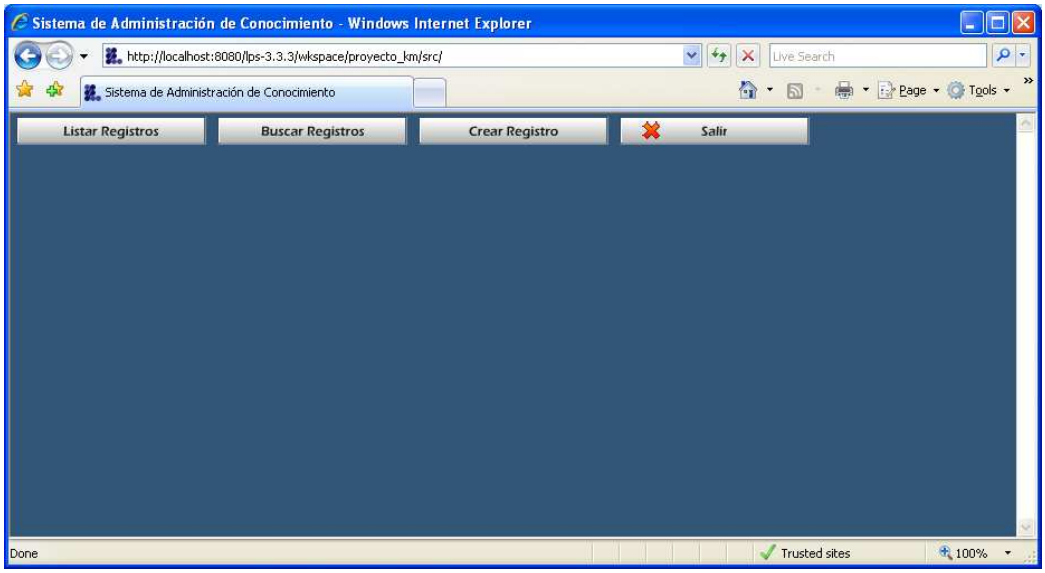


Figura 5.22
Funciones disponibles para el Administrador de conocimiento

La Figura 5.22 muestra las opciones disponibles para un usuario que tiene permisos de Administrador de conocimiento, en donde se puede observar que a diferencia de la Figura 5.3, no aparece el botón de Administrar Usuarios. Cabe

destacar que este estilo de menú, con los botones en la parte superior de la pantalla, siempre está visible y no cambia durante la sesión del usuario. Por lo que las figuras siguientes están dedicadas exclusivamente al despliegue de las pantallas para el usuario de Tipo Lector, quien es el que tiene más restricciones en el sistema.

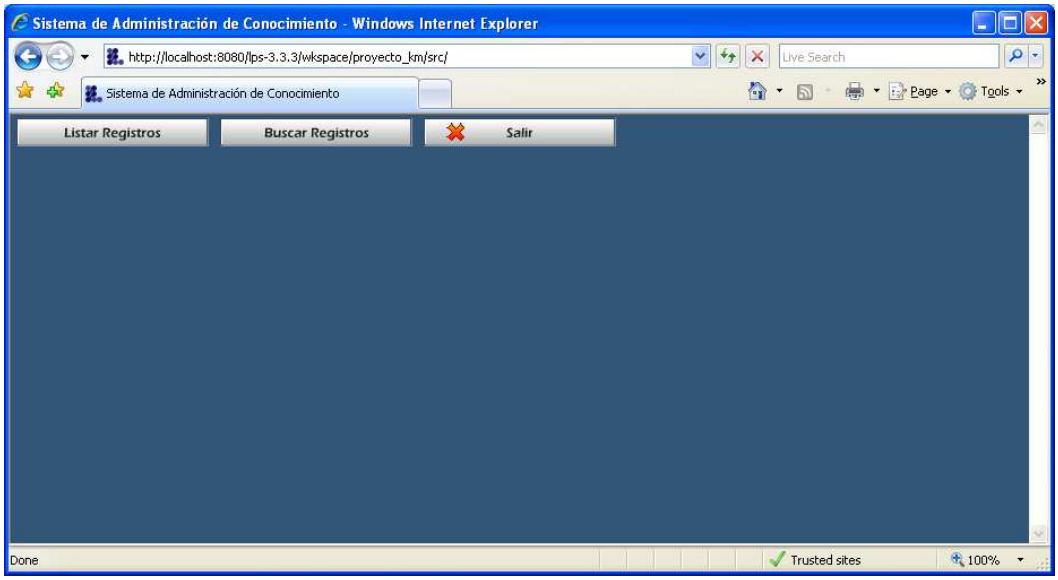


Figura 5.23
Funciones disponibles para el Lector

En la Figura 5.23 se muestran las opciones disponibles para un usuario de tipo Lector, el cual sólo puede realizar consultas en el sistema.

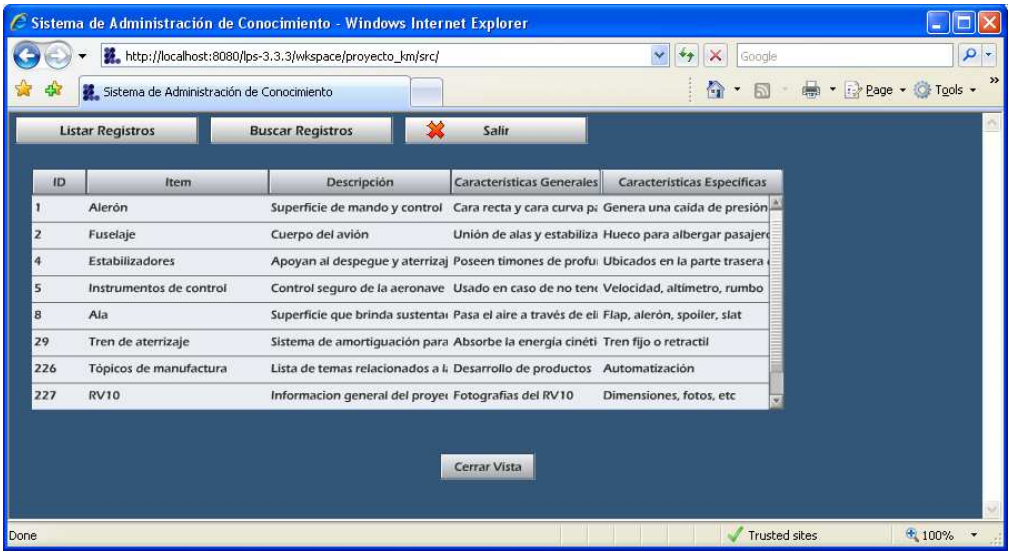


Figura 5.24
Listado de registros para el Lector

La Figura 5.24 indica que el listado de registros de conocimiento para un usuario de tipo lector no ofrece los botones de Borrar y Anexar que se muestran en la Figura 5.5, reduciendo el permiso solo para hacer doble clic en el registro deseado y mostrar su detalle.

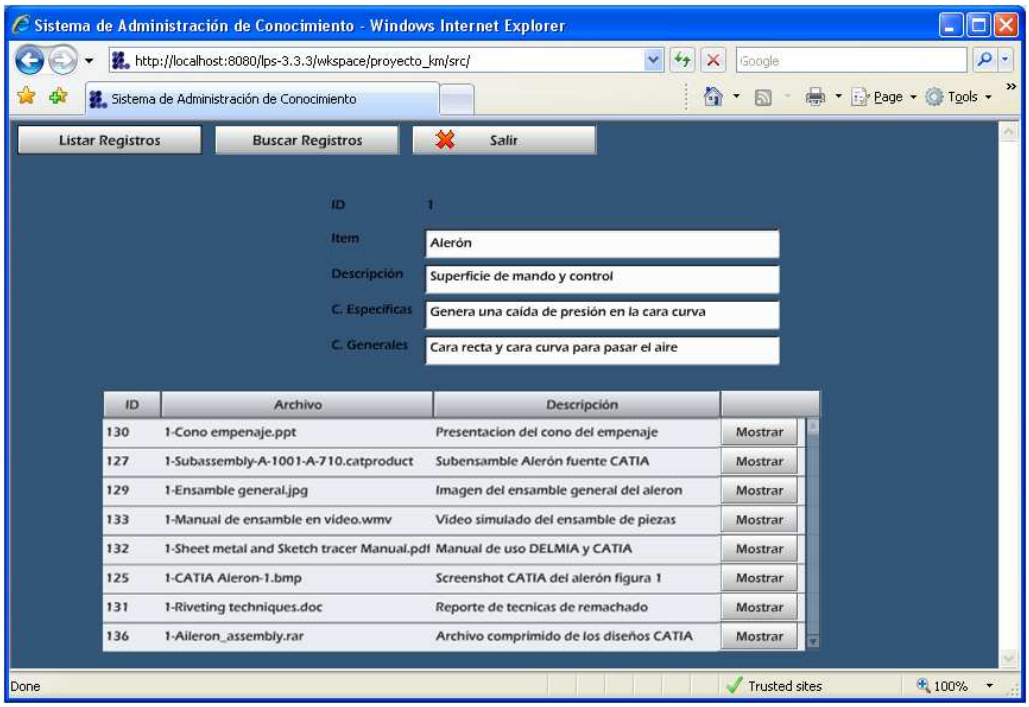


Figura 5.25
Registro de conocimiento consultado como Lector

Podemos observar en la Figura 5.25 que únicamente es posible mostrar el contenido del archivo anexo cuando se está en la vista de detalle del registro de conocimiento. La reproducción de los anexos es la misma que se describe en la sección 5.2.6. Al mismo tiempo, se puede apreciar que a diferencia de la Figura 5.9, no se muestran más botones relativos a la administración del registro de conocimiento (Anexar Archivo, Actualizar Registro, etc.)

Respecto a la búsqueda de registros para el usuario Lector, la manera de mostrar la pantalla es similar a la ilustrada en la Figura 5.6, sin embargo, la tabla dinámica resultante no ofrece opciones para Borrar o Anexar; todo esto es resumido en la Figura 5.26.

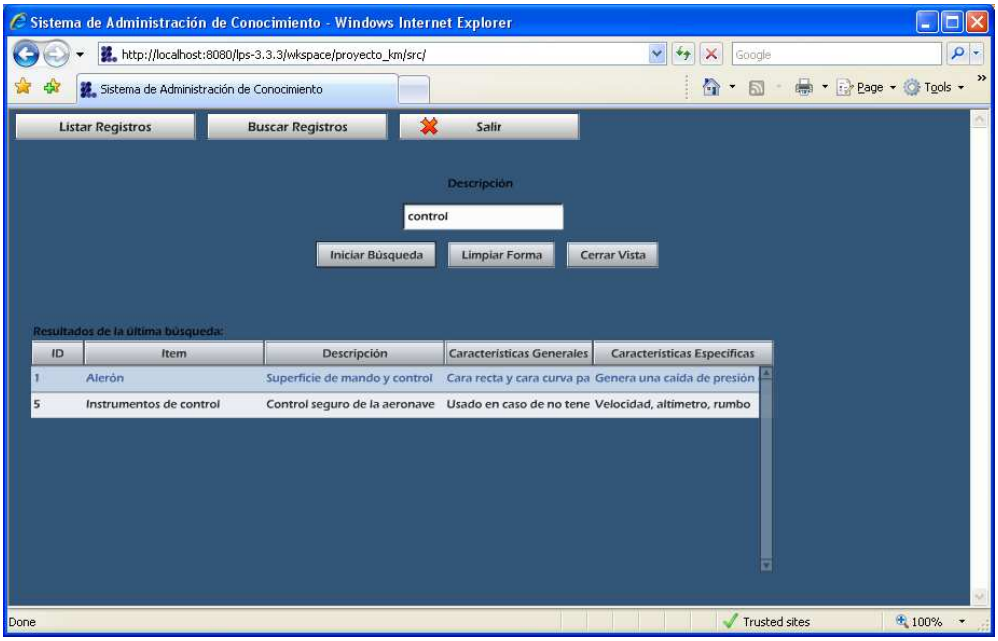


Figura 5.26
Búsqueda de registros para el Lector

## 5.4. Estructura de la información almacenada en el sistema

Para ejemplificar claramente la forma de operar el sistema de administración de conocimiento desarrollado y plantear una manera de clasificar la información del proyecto RV-10, fue necesario almacenar datos generales de todas las etapas que involucran este proyecto de ensamble de acuerdo al modo de clasificación descrito en la sección 4.5.

Debido a que el proyecto RV-10 se encuentra en proceso, los archivos anexos almacenados en el sistema están orientados únicamente al ensamble de alerón (aleta giratoria ubicada en la parte posterior de las alas). Para ello se recopiló y estructuró información suficiente con el objetivo de evidenciar las funcionalidades descritas en la sección 5.2.

Por lo anterior, la Figura 5.27 que se muestra a continuación, resume la estructura de la información utilizada, en donde se puede observar la dependencia de los archivos anexos a un registro de conocimiento particular (alerón), siendo posible seguir agregando anexos a ese registro conforme la información del proyecto se va generando, manteniéndola actualizada.

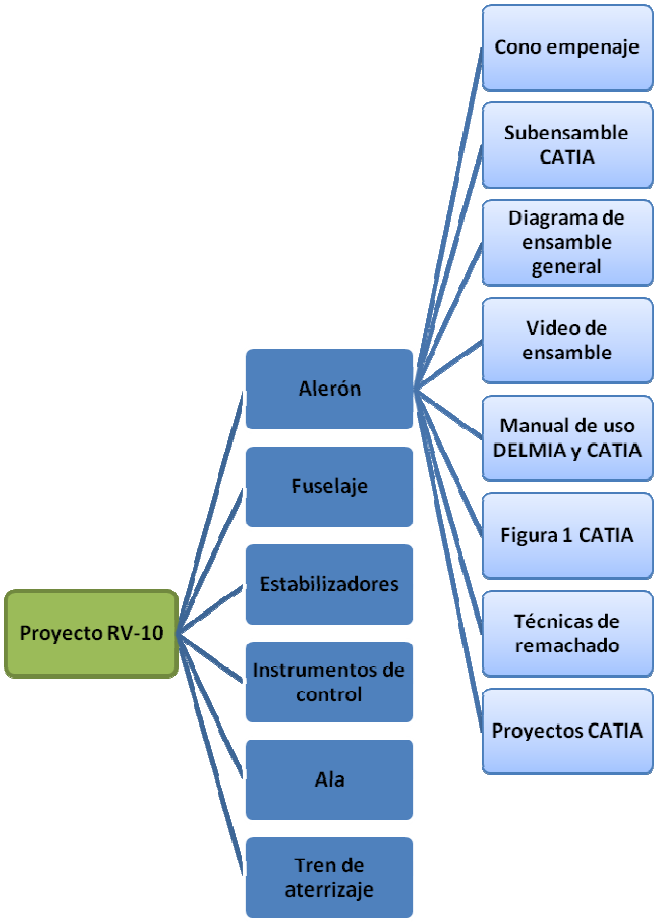


Figura 5.27
Estructura de la información almacenada

## 5.5. Resumen del capítulo

En este capítulo se describió a detalle el caso de estudio utilizado para validar la funcionalidad del sistema de administración de conocimiento desarrollado, iniciando por la definición del proyecto de ensamble RV-10 y posteriormente explicando la manera de utilizar la aplicación, además de mostrar las respuestas que se obtienen al realizar cada operación.

Al mismo tiempo, se ilustró la diferencia de los niveles de usuario definidos en el sistema gracias a las pantallas de salida que se obtienen al autentificarse con cierto nivel de permisos.

Se agruparon las funcionalidades por tipo de usuario, haciendo un flujo de operación ordenado donde se detalló el resultado de seleccionar botones específicos para interactuar con la aplicación. La reproducción de contenidos multimedia

también se orientó a un caso general de despliegue de información para clarificar las funcionalidades descritas en el capítulo 4.

Al final, se ilustró la manera en que se estructuró la información capturada en el sistema, dependiendo del tipo de clasificación utilizado y el avance del proyecto de ensamble hasta el momento.

# Capítulo 6. Conclusiones y trabajo futuro

En este capítulo se resumen los resultados obtenidos de la propuesta presentada en el capítulo 4 y ejemplificada en el capítulo 5. A su vez, se detallan las experiencias aprendidas y el trabajo futuro relacionado.

## 6.1. Resumen

Este trabajo surge por el interés de ofrecer una alternativa de aplicaciones para administrar conocimiento basadas en Web, orientadas hacia una emergente clasificación de software conocida como Rich Internet Application, aprovechando además las herramientas de desarrollo Open Source con la finalidad de promover la mejora continua de las aplicaciones sin necesidad de utilizar software propietario.

Para lograr lo anterior, fue necesario realizar una investigación acerca de las implicaciones de administrar conocimiento y sus ventajas, además de abordar los temas relacionados con tecnología de información en cuanto al desarrollo de aplicaciones en Web y su orientación hacia la colaboración (Web 2.0).

Asimismo, se investigó al respecto de las aplicaciones enfocadas al término AJAX, lo cual permite reproducir contenidos en Web de una forma dinámica y con una experiencia visual enriquecedora, dejando atrás a las aplicaciones de la Web 1.0. Es aquí donde se elige al framework de desarrollo llamado OpenLaszlo para ser utilizado como motor visual de la aplicación a desarrollar.

Por lo tanto, se realizó una propuesta de arquitectura de sistema, incluyendo las herramientas tecnológicas utilizadas, generación de casos de uso para la aplicación, diccionario de datos, diagramas de robustez y tipos de clasificación de la información para permitir su almacenamiento de tal modo que no fuera muy complejo el proceso de organización. La relevancia del sistema radica en la reproducción de contenidos multimedia dentro de la propia aplicación, permitiendo que usuarios no experimentados puedan emplear el sistema sin mayores contratiempos.

Al final, la propuesta tecnológica se comprobó mediante un caso de estudio dirigido a un proyecto de ensamble, donde se demuestra la funcionalidad del sistema gracias a la clasificación de la información y los permisos de usuario definidos en el desarrollo.

## 6.2. Experiencias aprendidas

Gracias al proceso de investigación desarrollado para la elaboración de este trabajo de tesis, se pueden resaltar las siguientes experiencias:

- Existen en la actualidad múltiples aplicaciones para administrar conocimiento que por lo general se concretan demasiado en la clasificación de la información, convirtiéndose en herramientas bastante complicadas de utilizar y aprovechar, ya que están orientadas a usuarios expertos. Además, sus interfaces visuales se basan en múltiples ventanas que pueden confundir al usuario.
- Durante la etapa de desarrollo de la aplicación con OpenLaszlo fue necesario invertir muchas horas de pruebas de código para verificar el funcionamiento de los componentes visuales y su interacción con orígenes de datos externos, ya que la conectividad con bases de datos el control de archivos anexos debe realizarse de manera independiente en la aplicación.
- Aunque existe mucha información relacionada con el desarrollo de aplicaciones con OpenLaszlo en Internet, no existe una lista de mejores prácticas o recomendaciones oficiales de implementaciones con bases de datos, ya que este framework es de capa de aplicación y gran parte de la información publicada asume que la información a desplegar ya se procesó de alguna manera y está lista para ser interpretada por OpenLaszlo. Por lo que gran parte del funcionamiento del sistema de administración de conocimiento está soportado por el desarrollo independiente que se realizó en JSP gracias a la estructuración de los resultados por medio de XML.
- La interactividad que puede ofrecer OpenLaszlo en su versión 3.3.3 depende en gran medida del cumplimiento de los requisitos mínimos del cliente, pues aunque las aplicaciones desarrolladas pretenden ser independientes del sistema operativo y navegador Web en uso, es muy importante tener las versiones más recientes de los plugins (Macromedia Flash, Reproductor de Windows Media), aunque en las versiones posteriores de OpenLaszlo que han surgido después del desarrollo de este sistema de administración de conocimiento ya ofrecen mejores beneficios.
- Las Rich Internet Applications no sólo ofrecen una nueva manera de desplegar contenidos en el navegador Web, sino que también cambian la perspectiva del desarrollador en cuanto a la manera de estructurar una aplicación. Ya que la correcta y oportuna identificación de las capas que conforman al sistema determinan los límites y operaciones de cada componente. Permitiendo la interoperabilidad con otros sistemas que pueden complementar a la aplicación original, esto se traduce en mayor complejidad en el desarrollo.

## 6.3. Conclusiones

La administración de conocimiento es un tema muy amplio que se resume en convertir algo que se sabe por experiencia, en conocimiento "tangible", es decir, convertirlo en algo documentado y replicable. La importancia de la administración de conocimiento va más allá de solo tenerlo por escrito, sino ponerlo a disposición de la gente que puede sacar provecho de la información. De ahí la relevancia de este tema en el presente documento de tesis, que busca integrarlo en una solución tecnológica de reciente generación, resumida en un sistema de administración de conocimiento.

La Web está evolucionando y ahora las soluciones de software para las organizaciones buscan trasladar sus sistemas hacia ambientes distribuidos y en línea, motivados por los retos de la disponibilidad, accesibilidad y flexibilidad de la información. Este trabajo de investigación muestra una visión general de los crecientes avances de las herramientas tecnológicas e infraestructura orientadas a ofrecer servicios Web de nueva generación, llamada Web 2.0.

La Web 2.0 es mucho más que una tecnología, es un cambio cultural de cómo usar el software para hacer nuevas cosas con nuevas herramientas, esto también afecta en la manera de crear el software. En la Web 2.0 se involucra el término Rich Internet Application que también persigue el mismo objetivo, tratando de mejorar las aplicaciones en cuanto a su diseño e interactividad.

La principal aportación de este trabajo de investigación es haber desarrollado una Rich Internet Application utilizando herramientas Open Source, demostrando que la reproducción de contenidos se puede realizar de manera dinámica en contraste con una aplicación de la generación Web 1.0. Además de orientarse hacia el tema de administración de conocimiento desde un enfoque práctico y con un proyecto real de investigación.

Las aplicaciones Web 1.0 ofrecen de manera muy limitada, pero aún funcional, información almacenada en una base de datos u orígenes externos de información. Sin embargo, una Rich Internet Application ofrece la ventaja de no hacer tantas recargas de datos. Por ejemplo, si se necesita hacer un ordenamiento específico de un listado de registros, basta con seleccionar la zona dinámica correspondiente para realizar el ordenamiento y los datos mostrados simplemente se actualizan sin necesidad de esperar una nueva recarga de información que implica una demora en la respuesta del sistema. (Figuras 5.5 y 5.9)

De acuerdo a lo planteado al inicio del trabajo de tesis, este sistema permite la transferencia de conocimiento a los diferentes integrantes de una organización en cualquier momento. Esto se logra a través de las bondades que ofrece un sistema en Web, ya que es accesible desde cualquier computadora con acceso a la red

corporativa al menos y con un navegador Web (las consideraciones de seguridad de redes en las organizaciones quedan fuera del alcance de este trabajo de investigación). En donde los usuarios dependiendo el nivel de permisos que le haya asignado el Administrador del sistema podrán consultar y aportar conocimiento para mantener activa la información en él. (Secciones 5.2 y 5.3)

A su vez, la mejora en la toma de decisiones y la generación de nuevas alternativas para mejorar la productividad también pueden ser cubiertas al utilizar este sistema de administración de conocimiento dado que la publicación en línea de procedimientos, técnicas y toda la información relevante de las operaciones de una organización permitirán que por ejemplo, un operador consulte de manera directa en el sistema los registros relacionados con una actividad en particular, apoyándose en los videos, imágenes o reportes que estén almacenados en la aplicación, lo cual le ayudará a cumplir de mejor manera sus actividades laborales y las de la organización. (Sección 5.2.6)

Además, se demuestra que con la arquitectura planteada orientada a una Rich Internet Application y la clasificación de información propuesta, un sistema de administración de conocimiento puede ser construido e implementado. Ofreciendo la particularidad del acceso y reproducción de la información sin la necesidad de utilizar una computadora de escritorio con gran poder de cómputo como puede ser una estación de trabajo con enormes capacidades gráficas (workstation, en inglés). (Capítulos 4 y 5)

Por último, el resultado de esta investigación permitió que los usuarios finales, al momento de observar el funcionamiento del sistema desarrollado, tuvieran una idea clara de cómo se abordaba el tema de la administración de conocimiento mediante un desarrollo en Web 2.0. Poniendo en consideración su futura implementación, después de la realización de ajustes en el sistema con el objetivo de cubrir necesidades específicas que quedan fuera del alcance de este proyecto.

## 6.4. Trabajo futuro

Posibles mejoras que son aplicables al sistema propuesto en este trabajo son:

- Actualizar el código hacia la versión de OpenLaszlo 4.0.6 (última versión disponible hasta el momento), donde se facilita la capacidad de convertir y reproducir los archivos de video utilizando el formato FLV (Flash Video File), evitando la dependencia de los archivos de video hacia un reproductor como el Windows Media Player y orientándose al despliegue de contenidos al estilo de sitios como YouTube [43].
- Añadir categorías o metadatos a los registros de conocimiento, para poder hacer listados de registros comunes basados en estos criterios.

- Ampliar los criterios de búsqueda de registros, incluyendo la localización de archivos anexos específicos basados en su descripción.
- Considerar la posibilidad de implementar una base de datos XML para tratar que OpenLaszlo controle toda la operación del sistema incluyendo la conectividad con la base de datos. Esto significa ajustar el código fuente para ofrecer una funcionalidad extra perceptible solo para el desarrollador y no para el usuario final.
- Permitir operaciones de "drag and drop" (arrastrar y soltar) para facilitar aún más la interacción con el usuario final, de tal manera que el usuario tenga una cesta de registros favoritos o de uso frecuente para su rápida consulta.
- Crear un módulo de revisión o control de versiones de los cambios de un registro de conocimiento. Esto es con la finalidad de que exista un moderador que autorice o niegue un cambio en algún registro de conocimiento dependiendo el nivel de permisos del usuario que hace la solicitud.
- Publicar los cambios asociados a un registro de conocimiento por medio de RSS (Really Simple Syndication, en inglés). Término que también es aplicable en la Web 2.0 donde es solamente necesario tener un lector de feeds (suministros de información, en inglés) para estar suscrito y mantenerse al día en relación a los cambios en la información capturada en el sistema.
- Investigar la manera de obtener información en OpenLaszlo con Web Services o middleware (software que conecta dos diferentes aplicaciones por separado) como CORBA, esto permitirá la posibilidad de crear una arquitectura orientada a servicios (SOA, por sus siglas en inglés).

## Referencias bibliográficas


1. Garibaldi, C., *Fundamentos de la administración del conocimiento*. 2003. p. 24-34.
2. Hahn, J. and M.R. Subramani. *A framework of knowledge management systems: issues and challenges for theory and practice*. in *ICIS '00: Proceedings of the twenty first international conference on Information systems*. 2000: Association for Information Systems.
3. Van der Vlist, E., et al., *Professional Web 2.0 Programming*. 2007, Indianapolis, IN: Wiley Publishing, Inc. 522.
4. Gregory, A., *Collaborate to accumulate*. Manufacturing Computer Solutions, 2000. 6(3): p. 7.
5. Rowley, J., *What is knowledge management?* Library Management, 1999. 20(8): p. 416-419.
6. King, W.R., P.V. Marks, and S. McCoy, *The most important issues in knowledge management*. Communications of the ACM, 2002. 45(9): p. 93-97.
7. Winch, G., *Knowledge management and competitive manufacturing*. Engineering Management Journal, 2000: p. 130-134.
8. Agostini, A., et al. *Stimulating knowledge discovery and sharing*. in *GROUP '03: Proceedings of the 2003 international ACM SIGGROUP conference on Supporting group work*. 2003: ACM Press.
9. O'Reilly, T. *What Is Web 2.0*. 2005 [cited 2007]; Available from: http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html?page=1.
10. Cortés, O., *Web 2.0 Sé parte de ella*, in *Software Guru, Conocimiento en práctica*. 2007.
11. Soto, L.D., *Web 2.x y Web 3.x*, in *Software Guru, Conocimiento en práctica*. 2007.
12. Hinchcliffe, D. *How Web 2.0 Works*. 2005 [cited 2007]; Available from: http://web2.socialcomputingmagazine.com/howweb2works.htm.
13. World Wide Web Consortium. *Extensible Markup Language (XML) 1.0*. 2006 [cited 2007]; Available from: http://www.w3.org/TR/2006/REC-xml-20060816/#sec-terminology.
14. World Wide Web Consortium. *XML en 10 puntos*. 2001 [cited 2007]; Available from: http://www.w3.org/XML/1999/XML-in-10-points.es.html.
15. Moore, D., R. Budd, and E. Benson, *Professional Rich Internet Applications: AJAX and Beyond*. 1 ed. 2007, Indianapolis, IN: Wiley Publishing, Inc. 565.
16. Eichorn, J., *Understanding AJAX: Using JavaScript to Create Rich Internet Applications*. 2006: Prentice Hall. 352.
17. Klein, N., M. Carlson, and G. MacEwen, *Laszlo in Action*. 1 ed. 2007: Manning Publications. 500.
18. Information Society Directorate General of the European Commission (2000) *Free Software / Open Source: Information Society Opportunities for Europe?* Information Society Technologies Volume, 42

19. Laszlo Systems. *OpenLaszlo. The premier open-source platform for rich internet applications*. 2006 [cited 2007]; Available from: http://www.openlaszlo.org/.
20. Ramparany, F., M. Aben, and L. Solorzano, *An integrated Knowledge Engineering Workbench*. Proceedings of the IEEE/ACM International Conference on Developing and Managing Expert System Programs, 1991.
21. Open Text Corporation. *Livelink ECM - Knowledge Management - Open Text corporation*. 2007 [cited 2007]; Available from: http://www.opentext.com/corporate/contact_us.html.
22. Convera Inc. *Convera, The Vertical Search Company - Solutions: RetrievalWare*. 2007 [cited 2007]; Available from: http://www.convera.com/solutions/retrievalware/Default.aspx.
23. Institute for Manufacturing. *Tool for Action Plans Selection (TAPS)*. 2007 [cited 2007]; Available from: http://www.ifm.eng.cam.ac.uk/csp/news/04april/1.html.
24. Tan, K.H. and K. Platts, *A connectance-based approach for managing manufacturing knowledge*. Industrial Management + Data Systems, 2004. 104(1/2): p. 158.
25. Digital Quality Systems. *Zavanta® Knowledgebase Software - "Everyone knows what to do"*. 2007 [cited 2007]; Available from: http://www.dqsnet.com/knowledgemanagement.html.
26. Mesbah, A. and A. van Deursen. *An Architectural Style for Ajax*. in *Conference on Software Architecture (WICSA'07)*. 2007.
27. Dailey Paulson, L., *Building Rich Web Applications with Ajax*, in *Computer*. 2005. p. 4.
28. Google Inc. *Google LABS Suggest*. 2007 [cited 2007]; Available from: http://www.google.com/webhp?complete=1&hl=en.
29. Laszlo Systems (2006) *OpenLaszlo An Open Architecture Framework for Advanced Ajax Applications*. Volume, 19
30. Coremans, C., *AJAX and Flash Development with OpenLaszlo: A Tutorial*. First ed. 2006: Brainy Software. 318.
31. Adobe Systems Inc. *Adobe - Flash Player Version Penetration*. 2007 [cited 2007]; Available from: http://www.adobe.com/products/player_census/flashplayer/version_penetration.html#ft4.
32. Apache Friends, *apache friends - xampp*. 2007.
33. Sun microsystems, *Java SE Downloads - Previous Release - JDK 5*. 2007.
34. MySQL AB, *MySQL AB :: Download Connector/J 5.0*. 2007.
35. Eclipse Foundation, *Eclipse - an open development platform*. 2007.
36. Bruno Leroux - EclipseTotale, *EclipseTotale - Sysdeo/SQLI Eclipse Tomcat Launcher Plugin*. 2007.
37. Eclipse Foundation, *Web Tools Platform (WTP) Project*. 2007.
38. IBM, *IDE4Laszlo - OpenLaszlo*. 2007.
39. Microsoft Corporation, *Microsoft Office System*. 2007.

40. Adobe Systems Inc, *Adobe Reader*. 2007.
41. Proyecto RV10 ITESM. *Proyecto de ensamble RV 10*. 2007 [cited 2007]; Available from: http://rv10.wikispaces.com.
42. Van's Aircraft Inc. *Van's Aircraft - Aircraft Models: RV-10 Introduction*. 2007 [cited 2007]; Available from: http://www.vansaircraft.com/public/rv-10int.htm.
43. Google Inc. *YouTube - Broadcast Yourself*. 2007 [cited 2007]; Available from: http://www.youtube.com/.

## Apéndice. Integrando JSP con OpenLaszlo

OpenLaszlo reconoce a XML como su único origen de datos, lo cual significa que no hay una implementación específica para las operaciones de extracción de información en una base de datos. Esto permite que se pueda desarrollar algún método para recuperar la información dependiendo las necesidades y experiencia del programador.

En este trabajo se desarrollaron archivos JSP específicos para operar sobre la base de datos y almacenar en un directorio particular los archivos anexos. Es decir, esta tarea no la realiza OpenLaszlo, sino que es lanzada por el motor de aplicación y quien la ejecuta es Java. Este apéndice se enfoca a detallar la manera en que se realizó la implementación e integración de JSP con OpenLaszlo.

Por ejemplo, para el caso del login o identificación del usuario se desarrolló el siguiente código en OpenLaszlo:

```
<!-- Clase para hacer el login -->
<class name="myLogin" height="120">
       <text fgcolor="white" y="10">Usuario:</text>
       <edittext name="username" x="100" y="10" width="200" />
       <text fgcolor="white" y="50">Password:</text>
       <edittext name="password" x="100" y="50" width="200" password="true" />
       <method name="sendData" args="action">
              var d=canvas.datasets.DSenvialogin; var p=new LzParam();
              p.addValue("action", action, true); p.addValue("username",
              username.getText(), true); p.addValue("password",
              password.getText(), true); d.setQueryString(p); d.doRequest();
       </method>
</class>
```

En el código anterior se respeta la sintaxis del formato LZX y se emplean objetos específicos de OpenLaszlo, pero resalta entre ellos el dataset (o conjunto de datos, en inglés) llamado DSenvialogin, mismo que se define en los encabezados de la aplicación como sigue:

```
<!-- Datasets con jsp's -->
       <dataset name="DSenvialogin" type="http" src="search_login.jsp" />
```

Como se puede observar, se hace referencia a un archivo JSP titulado search_login, el cual es realmente quien recibe los datos del usuario, establece la conexión a la base de datos y regresa el resultado en formato XML hacia OpenLaszlo. El siguiente código ejecuta lo anterior:

```
<%@ page import="java.sql.*"%>
<%@ page import="java.io.*" %>
<response>
<%
    Connection connection = null;
    try {
        Class.forName("com.mysql.jdbc.Driver");
        connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/laszlo?user=root&password=admin");
```

```
        Statement stmt = connection.createStatement();
        String logName = request.getParameter("username");
        String password = request.getParameter("password");
        String actioN = request.getParameter("action");
        FileWriter fichero = null;
        fichero = new FileWriter("e:\\Tesis\\ol_final\\Server\\lps-
               3.3.3\\wkspace\\proyecto_km\\queries\\queries.xml");
        String texto="<response>";
        fichero.write(texto + "\r\n");
        ResultSet rs2 = stmt.executeQuery("select * from users where logname ='"+ logName
               +"' and password ='"+ password +"'");
        int count = 0;
        while (rs2.next()) {
               count++;
        }
        ResultSet rs = stmt.executeQuery("select * from users where logname ='"+ logName +"'
               and password ='"+ password +"'");
        while (rs.next()) {
       texto="<knowledge logname=\""+rs.getString("logname")+"\"\r\n "
               +"password=\""+rs.getString("password")+"\"\r\n "
               +"permission=\""+rs.getString("permission")+"\"\r\n "
               +"resultado=\"Acceso Permitido\"/>";
               fichero.write(texto + "\r\n");
%>
       <knowledge logname="<%= rs.getString("logname")%>"
               permission="<%= rs.getString("permission")%>"
               password="<%= rs.getString("password")%>"/>
<%
        }
        if (count!=1) {
               texto="<knowledge resultado=\"Acceso Negado\"/>";
               %>
                       <knowledge resultado="Acceso Negado"/>
               <%
        }
        texto="</response>";
        fichero.write(texto);
        fichero.close();
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        try {
            connection.close();
        } catch (SQLException e) {
        }
    }
%>
</response>
```

Este archivo JSP recibe como parámetros el usuario y contraseña introducidos en la interfaz de OpenLaszlo y define un archivo de salida para efectos de debugging (depuración de código, en inglés) llamado queries.xml que contiene la respuesta en formato XML, misma que se envía a la aplicación OpenLaszlo para su interpretación y despliegue en pantalla.

Todo el código descrito anteriormente aplica exclusivamente al funcionamiento explicado en la sección 5.2.1.

El siguiente código corresponde a una vista completa en OpenLaszlo llamada agregarView, en donde se controla el módulo para crear un registro de conocimiento, descrito en la sección 5.2.4.

```
<!-- Vista para crear registro de conocimiento-->
<view name="agregarView" width="100%" height="100%" x="20" y="50"
        visible="false" font="verdana">
        <form x="${(canvas.width / 3.5)}" name="forma_crea">
                <submit name="contact2" data="${DSnuevoreg}" />
                <view name="crea_formato">
                        <text x="0" y="2">Item</text>
                        <edittext x="80" y="2" width="300" name="item" />
                        <text x="390" y="2" visible="false" name="obligatorio">
                                *Dato requerido
                        </text>
                        <text x="0" y="35">Descripción</text>
                        <edittext x="80" y="35" width="300" name="details" />
                        <text x="0" y="65">C. Generales</text>
                        <edittext x="80" y="65" width="300" name="general" />
                        <text x="0" y="95">C. Específicas</text>
                        <edittext x="80" y="95" width="300" name="concrete" />
                </view>
                <view>
                        <simplelayout axis="x" spacing="10" />
                        <button text="Agregar Registro" isdefault="true">
                                <method event="onclick">
                                        if (
                                        agregarView.forma_crea.crea_formato.item.getText()
                                        == '' ) {
                                agregarView.forma_crea.crea_formato.obligatorio.setAttribute(
                                        'visible', true ); } else {
                                agregarView.forma_crea.crea_formato.obligatorio.setAttribute(
                                        'visible', false ); parent.parent.contact2.submit();
                                        datapath.updateData();
                                        canvas.DSultimaalta.doRequest();
                                        canvas.DSultimaalta.doRequest();
                                        datapath.updateData();
                                        agregarView.forma_crea.crea_formato.item.setText("");
                                agregarView.forma_crea.crea_formato.details.setText("");
                                agregarView.forma_crea.crea_formato.general.setText("");
                                agregarView.forma_crea.crea_formato.concrete.setText("");
                                        }
                                </method>
                        </button>
                        <button name="limpiarBtn">
                                Limpiar Forma
                                <handler name="onclick">
                                        <![CDATA[
        agregarView.forma_crea.crea_formato.obligatorio.setAttribute( 'visible', false);
        agregarView.forma_crea.crea_formato.item.setText("");
        agregarView.forma_crea.crea_formato.details.setText("");
        agregarView.forma_crea.crea_formato.general.setText("");
        agregarView.forma_crea.crea_formato.concrete.setText(""); ]]>
                                </handler>
                        </button>
                        <button name="salidaBtn">
                                Cerrar Vista
                                <handler name="onclick">
                                        <![CDATA[ canvas.agregarView.hide(); ]]>
                                </handler>
                        </button>
                </view>
        </form>
        <simplelayout axis="y" spacing="20" />
        <view>
                <simplelayout axis="y" />
                <text>Último registro dado de alta:</text>
```

```
                <mygrid datapath="DSultimaalta:/response/"
                        contentdatapath="knowledge" shownitems="1" showhlines="true">
                        <gridtext width="50" datapath="@id" editable="false"
                                sortable="true" resizable="false">
                                ID
                        </gridtext>
                        <gridtext width="${(canvas.width / 5) - 10}"
                                datapath="@item" editable="false" sortable="true"
                                resizable="false">
                                Item
                        </gridtext>
                        <gridtext width="${(canvas.width / 5) - 10}"
                                datapath="@details" editable="false" sortable="true"
                                resizable="false">
                                Descripción
                        </gridtext>
                        <gridtext width="${(canvas.width / 6) - 10}"
                                datapath="@general" editable="false" sortable="true"
                                resizable="false">
                                Características Generales
                        </gridtext>
                        <gridtext width="${(canvas.width / 5) - 10}"
                                datapath="@concrete" editable="false" sortable="true"
                                resizable="false">
                                Características Específicas
                        </gridtext>
                        <gridcolumn width="62">
                                <view datapath="position()">
                                        <button text="Borrar">
                                                <method event="onclick">
                        parent.datapath.getDataset();
                        Debug.write(parent.datapath.getNodeAttribute('id'));
                        res_cadena=parent.datapath.getNodeAttribute('id');
                        entero = parseInt(res_cadena);
                        LzBrowser.loadJS("window.open('http://localhost:8080/lps-
                        3.3.3/wkspace/proyecto_km/src/borrar.jsp?id="+
                        entero +"', '_blank',
                        'left=300,top=200,height=200,width=350,toolbar=no,location=no'),
                        '_blank'"); parent.datapath.deleteNode();
                                                </method>
                                        </button>
                                </view>
                        </gridcolumn>
                        <gridcolumn width="81">
                                <view datapath="position()">
                                        <button text="Anexar">
                                                <method event="onclick">
                        parent.datapath.getDataset();
                        Debug.write(parent.datapath.getNodeAttribute('id'));
                        res_cadena=parent.datapath.getNodeAttribute('id');
                        entero = parseInt(res_cadena);
                        LzBrowser.loadJS("window.open('http://localhost:8080/lps-
                        3.3.3/wkspace/proyecto_km/src/upload.jsp?registro="+
                        entero +"', '_blank',
                        'left=300,top=200,height=200,width=350,toolbar=no,location=no'),
                        '_blank'");
                                                </method>
                                        </button>
                                </view>
                        </gridcolumn>
                </mygrid>
        </view>
        <method name="display">
                if(this.visible == false) { this.setVisible(true);
                setOpacity(0.9); }
        </method>
        <method name="hide">
                if(this.visible == true) { this.setVisible(false); }
        </method>
</view>
```

De igual manera en este caso se define un dataset específico llamado DSultimaalta que es el que se ejecuta cuando el usuario selecciona el botón Agregar registro, después de haber introducido los datos solicitados. Este dataset también se encuentra en los encabezados de la aplicación y apunta al archivo altareg.jsp como se muestra en el siguiente código.

```
<!-- Datasets con jsp's -->
       <dataset name="DSultimaalta" type="http" src="altareg.jsp" />
```

Finalmente, el código del respectivo JSP contiene las instrucciones necesarias para recibir los parámetros enviados desde la interfaz de OpenLaszlo y almacenar en la base de datos la información, contestando nuevamente un objeto XML para que Openlaszlo lo interprete:

```
<%@ page import="java.sql.*"%>
<%@ page import="java.io.*" %>
<response>
<%
    Connection connection = null;
    try {
        Class.forName("com.mysql.jdbc.Driver");
        connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/laszlo?user=root&password=admin");
        Statement stmt = connection.createStatement();
        FileWriter fichero = null;
        fichero = new FileWriter("e:\\Tesis\\ol_final\\Server\\lps-
               3.3.3\\wkspace\\proyecto_km\\queries\\queries.xml");
        String texto="<response>";
        fichero.write(texto + "\r\n");
        ResultSet rs = stmt.executeQuery("select * from knowledge where id = (select max(id)
               from knowledge)");
        while (rs.next()) {
               texto="<knowledge id=\""+rs.getString("id")+"\"\r\n "
               +"item=\""+rs.getString("item")+"\"\r\n "
               +"details=\""+rs.getString("details")+"\"\r\n "
               +"general=\""+rs.getString("general")+"\"\r\n "
               +"concrete=\""+rs.getString("concrete")+"\"/>";
               fichero.write(texto + "\r\n");
%>
            <knowledgeid="<%= rs.getString("id")%>"
                      item="<%= rs.getString("item")%>"
                      details="<%= rs.getString("details")%>"
                      general="<%= rs.getString("general")%>"
                      concrete="<%= rs.getString("concrete")%>"/>
<%
        }
        texto="</response>";
        fichero.write(texto);
        fichero.close();
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        try {
            connection.close();
        } catch (SQLException e) {
        }
    }
%>
</response>
```

Esto significa que las vistas de la aplicación son las que definen cuándo se invocan los archivos JSP para que se efectúe una operación sobre la base de datos, en donde el modo de comunicación es mediante objetos XML.

El caso de los archivos anexos tiene una peculiaridad extra, ya que en él se incluye código JavaScript para la manipulación de los archivos multimedia y su correcta reproducción en el navegador Web. El siguiente es un fragmento de código de la vista que controla el detalle de un registro de conocimiento, permitiendo acceder a los archivos anexos relacionados con él. Debido a que esta vista es la más compleja, solo se muestra la parte correspondiente a la columna de la tabla dinámica que despliega el botón de Mostrar ilustrada en las Figuras 5.9, 5.12, 5.13 y 5.25.

```
<gridcolumn width="85" resizable="false">
        <view datapath="position()">
                <button text="Mostrar">
                        <method event="onclick">
                                parent.datapath.getDataset();
                                Debug.write(parent.datapath.getNodeAttribute('id'));
                                res_cadena_id=parent.datapath.getNodeAttribute('id');
                                contacto=parent.datapath.getNodeAttribute('id_kw');
                                num_id = parseInt(contacto);
                                Debug.write(parent.datapath.getNodeAttribute('anexo'));
                                res_cadena_x=parent.datapath.getNodeAttribute('anexo');
                                entero = parseInt(res_cadena_id);
extension=res_cadena_x.substring(res_cadena_x.lastIndexOf("."),res_cadena_x.length());
                                if (extension==".mp3" || extension==".avi" ||
                                extension==".mpg" || extension==".wmv" ||
                                extension==".wav") {
                                LzBrowser.loadJS("window.open('http://localhost:8080/lps-
                                        3.3.3/wkspace/proyecto_km/src/multimedia.jsp?id="+
                                        entero +"', '_blank',
                                        'left=100,top=100,height=480,width=640,toolbar=no,locat
                                        ion=no,resizable=yes,scrollbars=yes'),
                                        '_blank'"); } else { if (extension==".png" ||
                                        extension==".jpg" || extension==".gif" ||
                                        extension==".bmp") {
                                LzBrowser.loadJS("window.open('http://localhost:8080/lps-
                                        3.3.3/wkspace/proyecto_km/src/pictures.jsp?id="+
                                        entero +"', '_blank',
                                        'left=100,top=100,height=480,width=640,toolbar=no,locat
                                        ion=no,resizable=yes,scrollbars=yes'),
                                        '_blank'"); } else {
                                LzBrowser.loadJS("window.open('http://localhost:8080/lps-
                                        3.3.3/wkspace/proyecto_km/src/otro_anexo.jsp?id="+
                                        entero +"', '_blank',
        'left=200,top=200,height=200,width=350,toolbar=no,location=no,resizable=yes'),
                                '_blank'"); } }
                        </method>
                </button>
        </view>
</gridcolumn>
```

En este fragmento de código se define el archivo JSP que se invocará para que abra el archivo anexo dependiendo de su extensión.

Por ejemplo, cuando OpenLaszlo lanza el código para validar el tipo de archivo anexo y éste resulta ser de audio o video, se ejecuta el archivo multimedia.jsp. Su código es el siguiente:

```
<%@ page import="java.sql.*"%>
<%@ page import="java.io.*"%>
<head>
	<script language="JavaScript">
		function maximizar()
		{
			self.moveTo(0,0)
			self.resizeTo(screen.availwidth,screen.availheight)
		}
		function restaurar()
		{
			self.resizeTo(screen.availwidth*.7,screen.availheight*.7)
			self.moveTo(150,100)
		}
	</script>
</head>
<body>
<font face="Verdana" size="2">
	<center>
		<BUTTON onclick="window.close();">Cerrar Ventana</BUTTON>
		<button onclick="javascript:maximizar();">Maximizar</button>
		<button onclick="javascript:restaurar();">Restaurar</button>
	</center>
	<br>
<response>
<%
    Connection connection = null;
    try {
        Class.forName("com.mysql.jdbc.Driver");
        connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/laszlo?user=root&password=admin");
        Statement stmt = connection.createStatement();
        String iD = request.getParameter("id");
        FileWriter fichero = null;
        fichero = new FileWriter("e:\\Tesis\\ol_final\\Server\\lps-
		3.3.3\\wkspace\\proyecto_km\\queries\\anexos.xml");
        String texto="<response>";
        String cadena="";
        String desc="";
        String cadena_completa="../anexos/";
        fichero.write(texto + "\r\n");
        ResultSet rs = stmt.executeQuery("select * from annexed where id ='"+ iD +"'");
        while (rs.next()) {
		texto="<knowledge id=\""+rs.getString("id")+"\"\r\n "
			+"id_kw=\""+rs.getString("id_kw")+"\"\r\n "
			+"anexo=\""+rs.getString("attachment")+"\"\r\n "
			+"description=\""+rs.getString("description")+"\"/>";
			fichero.write(texto + "\r\n");
			cadena = rs.getString("attachment");
			desc = rs.getString("description");
%>
	<knowledge id="<%= rs.getString("id")%>"
	id_kw="<%= rs.getString("id_kw")%>" anexo="<%=cadena%>"
	description="<%= rs.getString("description")%>" />
<%
        }
        texto="</response>";
        fichero.write(texto);
        fichero.close();
%>
<center>
	<embed src="<%=cadena_completa+cadena%>" showstatusbar="1"
		autostart="true" loop="false" volume="100" />
```

```
        <p><b>Descripci&oacute;n:</b><br>
        <%=desc%>
</center>
<%
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        try {
            connection.close();
        } catch (SQLException e) {
        }
    }
%>
</response>
```

Aquí es donde se definen los controles de reproducción del archivo y los controles de la ventana en ejecución ilustrados en la Figura 5.15.

Para el caso de reproducir un archivo de imagen (Figura 5.16), entonces se invoca el código del archivo pictures.jsp, que contiene lo siguiente:

```
<%@ page import="java.sql.*"%>
<%@ page import="java.io.*"%>
<head>
        <script language="JavaScript">
                function maximizar()
                {
                        self.moveTo(0,0)
                        self.resizeTo(screen.availwidth,screen.availheight)
                }
                function restaurar()
                {
                        self.resizeTo(screen.availwidth*.7,screen.availheight*.7)
                        self.moveTo(150,100)
                }
        </script>
</head>
<body>
<font face="Verdana" size="2">
        <center>
                <BUTTON onclick="window.close();">Cerrar Ventana</BUTTON>
                <button onclick="javascript:maximizar();">Maximizar</button>
                <button onclick="javascript:restaurar();">Restaurar</button>
        </center>
        <br>
<response>
<%
    Connection connection = null;
    try {
        Class.forName("com.mysql.jdbc.Driver");
        connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/laszlo?user=root&password=admin");
        Statement stmt = connection.createStatement();
        String iD = request.getParameter("id");
        FileWriter fichero = null;
        fichero = new FileWriter("e:\\Tesis\\ol_final\\Server\\lps-
                3.3.3\\wkspace\\proyecto_km\\queries\\anexos.xml");
        String texto="<response>";
        String cadena="";
        String desc="";
        String cadena_completa="../anexos/";
        fichero.write(texto + "\r\n");
        ResultSet rs = stmt.executeQuery("select * from annexed where id ='"+ iD +"'");
        while (rs.next()) {
                texto="<knowledge id=\""+rs.getString("id")+"\"\r\n "
                        +"id_kw=\""+rs.getString("id_kw")+"\"\r\n "
```

```
                        +"anexo=\""+rs.getString("attachment")+"\"\r\n "
                        +"description=\""+rs.getString("description")+"\"/>";
                        fichero.write(texto + "\r\n");
                        cadena = rs.getString("attachment");
                        desc = rs.getString("description");
%>
            <knowledge id="<%= rs.getString("id")%>"
                        id_kw="<%= rs.getString("id_kw")%>"
                        anexo="<%=cadena%>"
                        description="<%= rs.getString("description")%>"/>
<%
        }
        texto="</response>";
        fichero.write(texto);
        fichero.close();
%>
<center>
        <img src="<%=cadena_completa+cadena%>">
        <p><b>Descripci&oacute;n:</b><br><%=desc%>
</center>
<%
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        try {
            connection.close();
        } catch (SQLException e) {
        }
    }
%>
</response>
</body>
```

Por último, en caso de ser un tipo de archivo no reconocido por el sistema (Figura 5.17), entonces se ejecuta el código del archivo otro_anexo.jsp que se muestra a continuación:

```
<%@ page import="java.sql.*"%>
<%@ page import="java.io.*" %>
<response>
<%
        Connection connection = null;
    try {
        Class.forName("com.mysql.jdbc.Driver");
        connection =
DriverManager.getConnection("jdbc:mysql://localhost:3306/laszlo?user=root&password=admin");
        Statement stmt = connection.createStatement();
        String iD = request.getParameter("id");
        FileWriter fichero = null;
        fichero = new FileWriter("e:\\Tesis\\ol_final\\Server\\lps-
               3.3.3\\wkspace\\proyecto_km\\queries\\anexos.xml");
        String texto="<response>";
        String cadena="";
        String desc="";
        String cadena_completa="../anexos/";
        fichero.write(texto + "\r\n");
        ResultSet rs = stmt.executeQuery("select * from annexed where id ='"+ iD +"'");
        while (rs.next()) {
               texto="<knowledge id=\""+rs.getString("id")+"\"\r\n "
                        +"id_kw=\""+rs.getString("id_kw")+"\"\r\n "
                        +"anexo=\""+rs.getString("attachment")+"\"\r\n "
                        +"description=\""+rs.getString("description")+"\"/>";
                        fichero.write(texto + "\r\n");
                        cadena = rs.getString("attachment");
                        desc = rs.getString("description");
%>
            <knowledge id="<%= rs.getString("id")%>"
```

```
                        id_kw="<%= rs.getString("id_kw")%>"
                        anexo="<%=cadena%>"
                        description="<%= rs.getString("description")%>"/>
<%
        }
        texto="</response>";
        fichero.write(texto);
        fichero.close();
%>
<body>
        <font face="Verdana" size="2">
        <center><button onclick="window.close();">Cerrar Ventana</button></center><br>
        <center>
        Haga clic derecho (Guardar como) en la siguiente liga para descargar el archivo<p>
        <a href="<%=cadena_completa+cadena%>"><%=cadena%></a>
        <p><b>Descripci&oacute;n:</b><br><%=desc%>
        </center>
</body>
    <%
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        try {
            connection.close();
        } catch (SQLException e) {
        }
    }
%>
</response>
```