Benemérita Universidad Autónoma de Puebla
Facultad de Ciencias de la Computación

SISTEMA DE ADMINISTRACIÓN Y CONSULTA DE LEYES VÍA WEB

TESIS

Que para obtener el Título de

LICENCIADO EN CIENCIAS DE LA COMPUTACIÓN

Presenta

Boleaga Rios Ana Gabriela

Índice de Figuras

Índice de Tablas

Introducción

Todos los aspectos de la vida del hombre se han visto transformados en las últimas décadas por los avances de la ciencia y la tecnología. Los sistemas de información son cada día más potentes. El acceso al conocimiento ha dado un gran salto, con la informática por ejemplo, pues hoy es posible en tiempo real, tener información de todo el mundo en segundos. Debido a esto, es necesario utilizar una herramienta para eliminar uno de los principales problemas que se presentan en la sociedad, dicho problema es la falta de conocimiento sobre las leyes que nos rigen.

La aplicación que se pretende desarrollar es un sistema Web basado en componentes reutilizables, con la finalidad de lograr un modelo de desarrollo efectivo en tiempo y recursos. Esta aplicación facilitará al usuario la obtención de información fidedigna de las leyes a través de pasos sencillos que lo guiarán en su búsqueda.

El proceso de desarrollo que se utilizará será el Proceso Unificado de Rational (Rational Unified Process en inglés, habitualmente resumido como RUP) que junto con el Lenguaje Unificado de Modelado UML, constituyen la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos.

Java será el lenguaje de desarrollo y Struts será el framework ó marco de trabajo [9] que se utilizará en la aplicación, estos elementos facilitarán la creación de este proyecto y brindarán las herramientas necesarias para crear componentes reutilizables.

Las leyes serán ingresadas a través de un módulo especial de captura que será desarrollado, dicha información se almacenará en una base de datos creada en MySQL. Por cada ley o norma almacenada se permitirá agregar libros, títulos, artículos, transitorios y archivos según sea necesario para que los usuarios de esta aplicación puedan realizar una búsqueda posterior en dicha información.

Organización de la Tesis

La tesis se encuentra organizada en 4 capítulos, cuyo contenido se describe a continuación.

En el capítulo 1 (Marco Teórico) se presentan los fundamentos teóricos que se utilizarán en esta tesis. Se tratan como puntos principales el lenguaje de programación Java, la programación Web y la base de datos MySQL.

En el capítulo 2 (Análisis) se realiza el modelo de análisis que contiene el documento de requerimientos del usuario y el documento de especificación de requerimientos de software.

En el capítulo 3 (Diseño) es especifican los casos de uso identificados en la aplicación, así como el diagrama de clases, paquetes y el modelo relacional de la base de datos. En cada caso de uso se describen los actores, flujo de eventos, flujo básico, diagrama de actividades, diagrama de secuencia y diagrama de colaboración, según las especificaciones del proceso de desarrollo RUP.

En el capítulo 4 (Implementación) se muestran los prototipos de interfaces y se explica la forma en la que se realizó la aplicación explicando cada uno de los módulo con los que cuenta.

1. Marco Teórico

1.1 Java

Java es un lenguaje de programación orientado a objetos, es el primer lenguaje que tiene la virtud de ser compilado e interpretado de forma simultánea. A diferencia de los lenguajes de programación convencionales, que generalmente están diseñados para ser compilados a código nativo, el compilador de Java genera bytecodes[1] que son ejecutados (usando normalmente un compilador Just-In-Time (JIT)[2]), por una Máquina Virtual Java (JVM).

Hasta ahora, la única forma de realizar una página web con contenido interactivo, era mediante la interfaz CGI[3] (Common Gateway Interface), que permite pasar parámetros entre formularios definidos en lenguaje HTML y programas escritos en Perl o en C. Esta interfaz resulta muy incómoda de programar y es pobre en sus posibilidades.

Utilizando Java, se pueden eliminar los inconvenientes de la interfaz CGI y también se pueden añadir aplicaciones que vayan desde sitios interactivos de propósito educativo a juegos o aplicaciones especializadas. Además, con Java podemos estar seguros de que el código no contiene ningún trozo de código malicioso que dañe al sistema.

Java proporciona una nueva forma de acceder a las aplicaciones. El software viaja transparente a través de la red. No hay necesidad de instalar las aplicaciones, ellas mismas vienen cuando se necesitan.

1.1.1 Máquina virtual Java

La tecnología Java está compuesta básicamente por 2 elementos: el lenguaje Java y su plataforma. Con plataforma nos referimos a la máquina virtual de Java.

La Máquina Virtual Java (JVM) es el entorno en el que se ejecutan los programas Java, su misión principal es la de garantizar la portabilidad de las aplicaciones Java. Define esencialmente un ordenador abstracto y especifica las instrucciones (bytecodes) que este ordenador puede ejecutar.

En cada plataforma (Unix, Linux, Windows 95/NT, Macintosh, etc.) existe una máquina virtual específica. De este modo, cuando el mismo bytecode llega a diferentes plataformas, éste se ejecutará de forma correcta, pues en cada una de esas plataformas existirá la máquina virtual adecuada. Con este mecanismo se consigue la famosa multiplataforma de Java, que con sólo codificar una vez, podemos ejecutar en varias plataformas. La independencia de plataforma es una de las razones por las que Java es interesante para Internet, ya que muchas personas deben tener acceso con ordenadores distintos. Pero no se queda ahí, Java está desarrollándose incluso para distintos tipos de dispositivos además del ordenador como móviles, agendas y en general para cualquier cosa que se le ocurra a la industria.

En realidad la máquina virtual desempeña otras funciones, como la de aislar los programas Java al entorno de la máquina virtual, consiguiendo una gran seguridad, siendo uno de los principales propósitos de Java, crear software altamente fiable. Para ello proporciona numerosas comprobaciones en compilación y en tiempo de ejecución. Sus características de memoria liberan a los programadores de una familia entera de errores (la aritmética de punteros), ya que se ha prescindido por completo los punteros, y la recolección de basura elimina la necesidad de liberación explícita de memoria.

---

[1] Bytecode es un código intermedio más abstracto que el código máquina, recibe su nombre porque generalmente cada código de operación tiene una longitud de un byte, si bien la longitud del código de las instrucciones varía. Se trata de una forma de salida utilizada por los implementadores de lenguajes para reducir la dependencia respecto del hardware.

[2] El compilador JIT convierte bytecodes en código máquina nativo. Este proceso de compilación se realiza sólo vez y un vínculo se crea entre el código de bytes y el código compilado correspondiente. El propósito de este proceso es aumentar la velocidad en la que se ejecutan programas Java. Puede realizar un aumento de velocidad de cuándo se realizan algunas funciones matemáticamente intensivas más de una 50 vez.

[3] CGI especifica un estándar para transferir datos entre el cliente y el programa.

1.1.2 Conexión con Bases de Datos mediante Java

1.1.2.1 Java DataBase Connectivity

En la actualidad surge la posibilidad de utilizar aplicaciones que permitan acceder a información de forma dinámica, tal como a bases de datos, con contenidos y formatos muy diversos.

Una de las ventajas de utilizar el Web para este fin, es que no hay restricciones en el sistema operativo que se debe usar, permitiendo la conexión entre sí, de las páginas Web desplegadas en un navegador que funciona en una plataforma, con servidores de bases de datos alojados en otra plataforma. Además, no hay necesidad de cambiar el formato o estructura de la información dentro de las bases de datos.

Para realizar una requisición de acceso desde el Web hasta una base de datos no sólo se necesita de un navegador Web y de un Servidor Web, sino también de un software de procesamiento (aplicación CGI), el cual es el programa que es llamado directamente desde un documento HTML en el cliente. Dicho programa lee la entrada de datos desde que provienen del cliente y toma cierta información de variables de ambiente.

JDBC (Conectividad de Base de Datos) es una Interfaz[4] de Programación de Aplicaciones (API[5]) que permite la ejecución de operaciones sobre bases de datos desde el lenguaje de programación Java independientemente del sistema de operación donde se ejecute o de la base de datos a la cual se accede utilizando el dialecto SQL del modelo de base de datos que se utilice.

Java no implementa Bases de Datos, ya que solo es un lenguaje de programación, pero implementa funciones que permiten al programador realizar conexiones entre la interfaz de usuario y el Gestor de Base de Datos.

Java permite conectarse por medio de puentes JDBC o a través de Drivers a programas gestores de bases de datos, su independencia entre ambos permite al usuario mantener siempre un enfoque, separando el diseño de la Base de Datos y el de la interfaz en dos mundos de pensamientos diferente, el mundo de los datos y el mundo de las interfaces.

1.1.2.2 Tipos de conectores (drivers) JDBC

Los conectores o drivers JDBC, se pueden dividir en cuatro tipos principalmente:

- **JDBC-ODBC (BRIDGE).** Permite el acceso a Base de Datos JDBC mediante un driver ODBC. Cada máquina cliente que use el puente, debe tener librerías clientes de ODBC.

- **Driver Java parciales (NATIVE).** Traducen las llamadas al API de JDBC Java en llamadas propias del motor de Base de Datos. Al igual que el tipo anterior, exige en las máquinas clientes código binario propio del cliente de la Base de datos específica y del sistema operativo.

- **Driver JDBC a través de Middleware (NETWORK).** Traduce las llamadas al API JDBC en llamadas propias del protocolo específico del broker. Éste se encargará de traducirlas de nuevo en sentencias propias del motor de Base de Datos de cada caso.

- **Driver java puro (THIN).** Convierte o traduce las llamadas al API JDBC en llamadas al protocolo de red usado por el motor de bases de datos, lo que en realidad es una invocación directa al motor de bases de datos. Totalmente Java. Un controlador Java puro que no requiere de código sistema operativo-dependiente, lo que hace al controlador 100% portable.

1.1.2.3 Arquitecturas típicas JDBC

La arquitectura básica de JDBC es simple. Una clase llamada DriverManager provee un mecanismo para controlar un conjunto de drivers JDBC. Esta clase intenta cargar los drivers especificados en la propiedad del sistema jdbc.drivers.

---

[4] Una interfaz especifica una lista de operaciones que tienen que proporcionar todo aquello que cumpla la interfaz [10].

[5] Una API (Application Programming Interface - Interfaz de Programación de Aplicaciones) es el conjunto de funciones y procedimientos que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción, representa un interfaz de comunicación entre componentes software.

- **Arquitecturas JDBC en dos capas.** La aplicación que accede a la base de datos reside en el mismo lugar que el driver de la base de datos. En este caso, será el driver el encargado de manejar la comunicación a través de la red, ver figura 1.1.

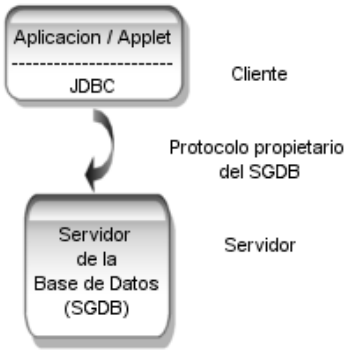

Figura 1.1. Arquitectura JDBC en dos capas

- **Arquitecturas JDBC en tres capas.** En el modelo de tres capas (three-tier), los comandos son enviados a una capa intermedia (middle-tier) de servicios, la cual enviará sentencias SQL a la base de datos. La base de datos procesa las sentencias y devuelve los resultados a la capa intermedia que se los enviará al usuario. Este modelo es bastante interesante, ya que la aplicación intermedia no poseerá las restricciones de seguridad de los applets y dejará más libertad al programador; otra ventaja del uso del modelo en tres capas, es que el usuario puede utilizar una API de más alto nivel, y por lo tanto más sencilla de manejar, que será traducida por la capa intermedia a las llamadas apropiadas, en este caso utilizando el API de JDBC, ver figura 1.2.

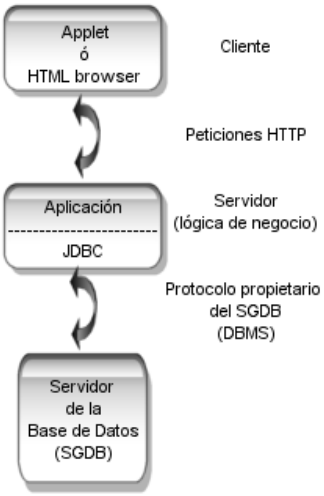

Figura 1.2. Arquitectura JDBC en tres capas

1.2 Programación Web

El desarrollo de aplicaciones web involucra decisiones no triviales de diseño e implementación que inevitablemente influyen en todo el proceso de desarrollo, afectando la división de tareas.

Existen en la actualidad tecnologías complejas que demoran el proceso de desarrollo e incrementan los costos, pero en ocasiones permite adecuarse a metodologías de diseño más fácilmente. Tal es el caso de las tecnologías orientadas a objetos, las cuales tienden a demorar el desarrollo en etapas tempranas pero la adopción de éstas hace que el mantenimiento se transforme en una actividad

más simple, la división en capas sea tarea natural del desarrollo y el tiempo invertido en el diseño facilite el trabajo necesario para el resto de las actividades.

1.2.1 Programación por capas

La estrategia tradicional de utilizar aplicaciones compactas causa gran cantidad de problemas de integración en sistemas software complejos como pueden ser los sistemas de gestión de una empresa o los sistemas de información integrados consistentes en más de una aplicación. Estas aplicaciones suelen encontrarse con importantes problemas de escalabilidad, disponibilidad, seguridad, integración, etc. Para solventar estos problemas se ha generalizado la división de las aplicaciones en capas que normalmente serán tres: una capa que servirá para guardar los datos (base de datos), una capa para centralizar la lógica de negocio (modelo) y por último una interfaz gráfica que facilite al usuario el uso del sistema, ver figura1.3.

Figura 1.3. Arquitectura tres capas

1.2.1.1 Capa de Presentación

El objetivo de la capa de presentación es encargarse de la representación de la información, de manera que aunque distintos equipos puedan tener diferentes representaciones internas de caracteres (ASCII, Unicode, etc.), números, sonido o imágenes, los datos lleguen de manera reconocible.

Esta capa es la primera en trabajar más el contenido de la comunicación que cómo se establece la misma. En ella se tratan aspectos tales como la semántica y la sintaxis de los datos transmitidos, ya que distintas computadoras pueden tener diferentes formas de manejarlas.

Por lo tanto, podemos resumir definiendo a esta capa como la encargada de manejar las estructuras de datos abstractas y realizar las conversiones de representación de datos necesarias para la correcta interpretación de los mismos, al mismo tiempo permite cifrar los datos y comprimirlos.

1.2.1.2 Capa de Negocio

Se denomina capa de negocio (e incluso de lógica del negocio) donde residen los programas que se ejecutan, recibiendo las peticiones del usuario y enviando las respuestas tras el proceso. Es donde se establecen todas las reglas que deben cumplirse. Esta capa se comunica con la capa de presentación, para recibir las solicitudes y presentar los resultados, y con la capa de datos, para solicitar al gestor de base de datos almacenar o recuperar datos de él.

1.2.1.3 Capa de Datos

La capa de datos está formada por uno o más gestores de bases de datos que realizan todo el almacenamiento de datos, recibe solicitudes de almacenamiento o recuperación de información desde la capa de negocio.

Normalmente un componente de la capa de datos implementa las operaciones de creación, lectura, actualización y eliminación de registros. Además de estas operaciones, la capa de datos también implementa otros métodos que implementan la lógica que esté muy cercana a los datos (por ejemplo filtros sobre los datos, cálculos agregados, etc.).

El interés por esta capa de datos es cada vez mayor, puesto que las empresas tratan de tener un mejor conocimiento de los elementos de una transacción.

1.2.2 Struts

Los Servlets[6] de Java fueron una revolución y ya constituían una gran mejora frente a las CGI. Pero implicaba una gran interacción con el navegador. Debido a esto nacieron las Java Server Pages[7], que permiten tener servlets dentro de ellas. De esta manera se conseguía mezclar muy fácilmente el código Java con HTML. Pero este modelo no era suficiente ya que por ejemplo no resolvía problemas de control de flujo. Así que se necesitaba un nuevo modelo.

Struts es un framework (marco de trabajo) [9] de código abierto creado para facilitar la construcción de sitios web basados en Java Servlets y Java Server Pages (JSP). Struts es uno de los muy conocidos proyectos Apache Jackarta. El objetivo general del proyecto Jackarta es proveer soluciones de calidad comercial basadas en la plataforma Java manteniendo el estilo abierto y cooperativo. Como todo framework intenta simplificar notablemente la implementación de una arquitectura según el patrón Modelo Vista Controlador (MVC), donde los servlets se encargarían del control de flujo y los JSPs se encargarían de negociar con el navegador.

1.2.2.1 El patrón Modelo – Vista – Controlador

Modelo Vista Controlador (MVC) es un patrón de arquitectura de software que se ve frecuentemente en aplicaciones web, separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos, donde la vista es la página HTML y el código que provee de datos dinámicos a la página; el controlador es el Sistema de Gestión de Base de Datos y el modelo es el modelo de datos.

El patrón MVC se divide en tres partes:

- El Modelo contiene los datos y la funcionalidad de la aplicación. Es independiente de la representación de los datos.

- Las Vistas muestran la información al usuario de una cierta forma.

- El Controlador recibe como entrada las peticiones de los usuarios y determina la vista a usar para presentar los resultados.

En el patrón de diseño Modelo-Vista-Controlador, el flujo de la aplicación está dirigido por un Controlador central. El Controlador delega solicitudes (en nuestro caso, solicitudes HTTP) a un manejador apropiado. Los manejadores están unidos a un Modelo, y cada manejador actúa como un adaptador entre la solicitud y el Modelo. El Modelo representa, o encapsula, un estado o lógica de negocio de la aplicación. Luego el control es devuelto a través del Controlador hacia la Vista apropiada. El reenvío puede determinarse consultando los conjuntos de mapeos, cargados desde un fichero de configuración en XML. Esto proporciona un acoplamiento cercano entre la Vista y el Modelo, que hace que nuestra aplicación sea más fácil de crear y de mantener, ver figura 1.4.

La utilización de esta metodología en el diseño de una aplicación presenta la gran ventaja de producir código modular en el que la modificación de uno de sus componentes, por ejemplo la interfaz de usuario (la forma en la que el usuario navega por el web o la forma en la que se presentan los datos) no requiere modificar ninguno de los elementos de las otras capas. Asimismo, si por cualquier razón se migrase la información y se alojase en una base de datos diferente, únicamente sería necesario modificar la capa de datos.

---

[6] Un *servlet* es una clase java que puede ser llamada dinámicamente y que puede ser ejecutada en un servidor web especial [8] y que recibe y responde a las peticiones de uno o más clientes.

[7] JavaServer Pages es una tecnología que nos permite mezclar HTML estático con HTML generado dinámicamente al momento de la petición por parte del usuario, el Servidor de Aplicaciones interpreta el código contenido en la página JSP, para construir un Servlet, cuya salida será un documento estático (típicamente HTML).

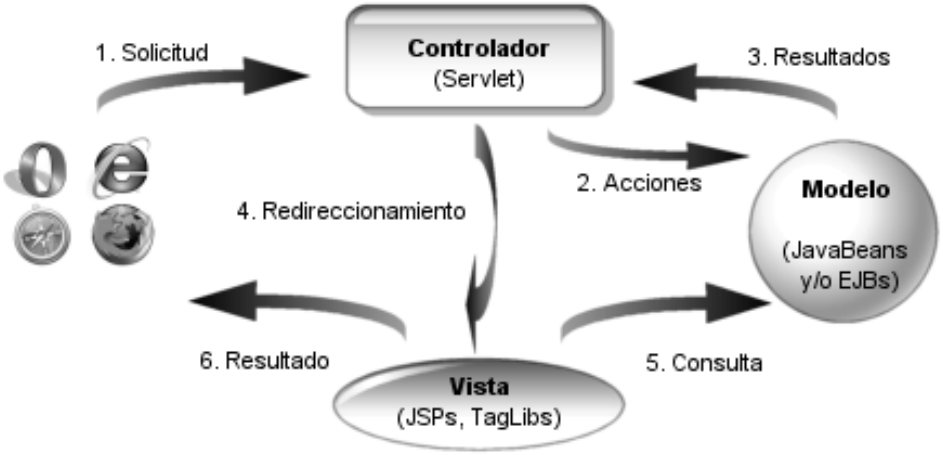

Figura 1.4. Modelo – Vista – Controlador

1.2.2.2 Struts, una implementación del patrón MVC

En el ámbito del desarrollo web se siguen pautas que tratan más o menos de conseguir un desarrollo estructurado de las aplicaciones, donde la verificación de sesión se centraliza y cada caso de uso[8] se distingue claramente. Utilizando Struts ese "más o menos" se convierte en una arquitectura completamente estructurada que divide perfectamente lógica de negocio (Modelo), presentación (Vista) y control de flujo de aplicaciones (Controlador). En muchos desarrollos web se diseña consciente o inconscientemente siguiendo este patrón, por tanto la adopción del modelo Struts no debiera suponer problema alguno.

Utilizando el patrón de diseño Modelo-Vista-Controlador, las aplicaciones Struts tiene tres componentes principales: un servlet controlador, que está proporcionado por el propio Struts, páginas JSP (la "vista"), y la lógica de negocio de la aplicación (o el "modelo"). Por tanto, los Struts nos proporcionan un marco para implementar el Modelo-Vista-Controlador y nosotros nos encargamos de generar las páginas JSP que se corresponderán con la lógica de negocio.

El controlador ya se encuentra implementado por Struts, aunque si fuera necesario se puede heredar y ampliar o modificar, y el workflow de la aplicación se puede programar desde un archivo XML. Las acciones que se ejecutarán sobre el modelo de objetos de negocio se implementan basándose en clases predefinidas por el framework.

1.2.2.3 Funcionamiento en aplicaciones WEB

El navegador genera una solicitud que es atendida por el Controlador (un Servlet especializado). El mismo se encarga de analizar la solicitud, seguir la configuración que se le ha programado en su XML y llamar al Action correspondiente pasándole los parámetros enviados. El Action instanciará y/o utilizará los objetos de negocio para concretar la tarea. Según el resultado que retorne el Action, el Controlador derivará la generación de interfaz a una o más JSPs, las cuales podrán consultar los objetos del Modelo para mostrar información de los mismos.

Este gráfico (Figura 1.5) nos provee una visión más detallada del funcionamiento de Struts

[8] Los casos de uso documentan el comportamiento del sistema desde el punto de vista del usuario [10].

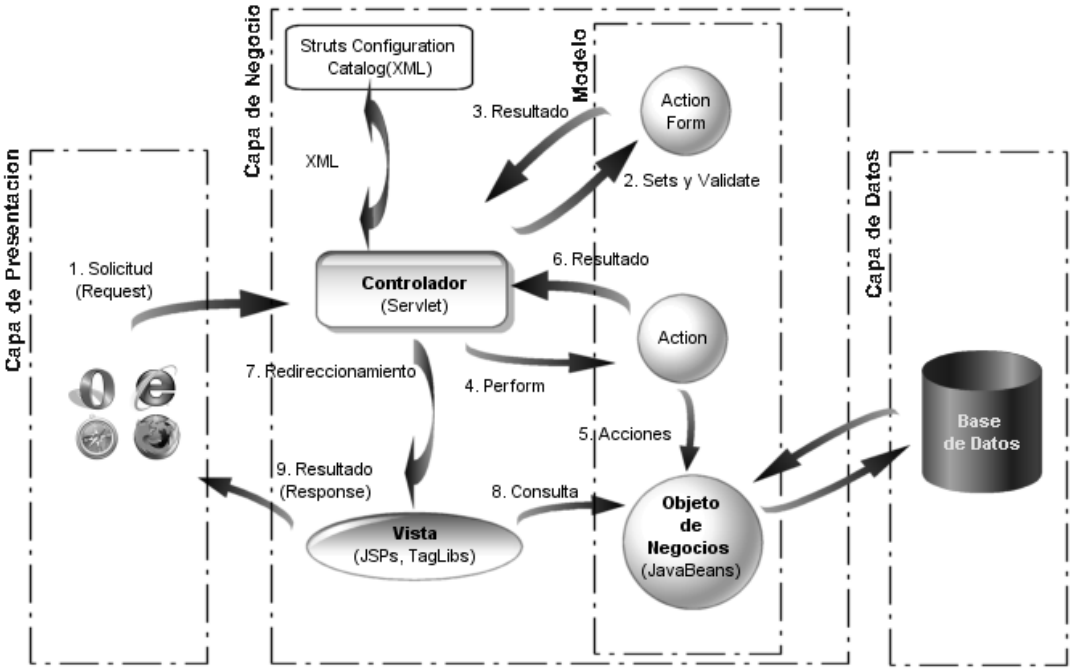

Figura 1.5. Struts en aplicaciones WEB

1.2.2.4 Estructura

Para contar con una aplicación bien estructurada de un formulario sencillo que recoge datos y los manda al servidor para su proceso, necesitaríamos implementar unos 6 ficheros (2 de ellos Jps).

Utilizando Struts nunca se llega a una página de la capa de presentación directamente. Esto es, en la URL nunca se llega a una página jsp o html a través de su nombre. De eso se trata el MVC, la presentación está separada en otra capa.

En este entorno, se debe invocar una acción o aplicación que debe estar mapeada en Struts: una acción se corresponderá con una clase Java (heredera de la clase Action de Struts). El mapeo de acciones y clases se especifica en un fichero de importancia vital: **struts-config.xml**. Ahí se especifican todas las relaciones entre acciones y clases, formularios y clases, acciones y jsps de presentación, que globalmente conforman el "mapa" de la aplicación.

Si quisiéramos crear la aplicación Struts más simple posible, por ejemplo una página con un saludo, debiéramos hacer lo siguiente:

- Una página JSP (la presentación)
- Una clase Action (componente del controlador)
- La clase Action debe definirse en el struts-config.xml correctamente

1.2.2.5 Diagrama de clases

En el caso más simple este sería el conjunto de clases utilizadas y sus relaciones.

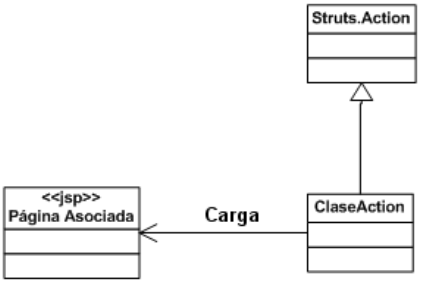

Figura 1.6. Diagrama de clases

Como se puede ver en la figura 1.6, no tendríamos más que dos ficheros, una clase que hereda de Struts.Action y una JSP con la presentación, siendo una clase el modelo, patrón o plantilla a partir del que se crea un objeto [7].

La clase Action se configura en el strtus-config.xml y se convierte en parte del controlador. Cuando la aplicación recibe una petición, Struts decidirá que debe cargar esa clase y a través de ella cargará el JSP.

1.2.2.6 Diagrama de colaboración

Como se muestra en la figura 1.7 sería la manera de interactuar entre las clases y Struts:

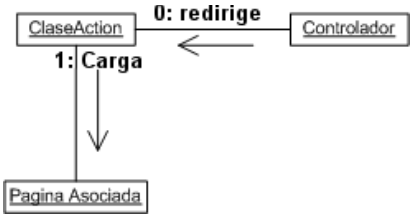

Figura 1.7. Diagrama de colaboración

Una petición llega a struts (una URL); éste mira en su "mapa" (el fichero struts-config.xml), y deduce que tiene que cargar la ClaseAction. Esta clase está configurada para que cargue una página JSP.

Como se puede observar, no se carga el JSP directamente, hay que pasar por el controlador.

1.2.2.7 Diagrama de secuencia

En la figura 1.8 se muestran los pasos seguidos por la aplicación en el plano temporal.

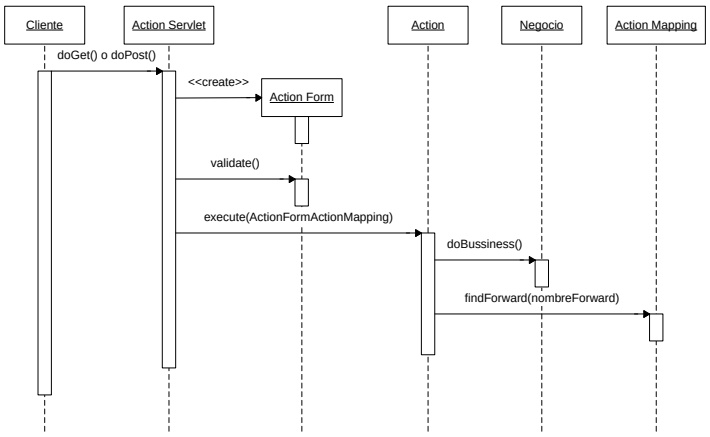

Figura 1.8. Diagrama de secuencia

1.2.2.8 Vista

La vista proporcionará una serie de páginas web dinámicamente al cliente, siendo para él simples páginas HTML. Existen múltiples frameworks que generan estas páginas web a partir de distintos formatos, siendo el más extendido el de páginas JSP, que mediante un conjunto de tags XML proporcionan un interfaz sencillo y adecuado a clases Java y objetos proporcionados por el servidor de aplicaciones. Esto permite que sean sencillas de desarrollar por personas con conocimientos de HTML. Entre estos tags tienen mención especial la librería estándar JSTL (*JavaServer Pages Standard Tag Library*) que proporciona una gran funcionalidad y versatilidad.

1.2.2.8.1 Tags

Los elementos que deben representar los JSP a veces requieren de mucho código, por lo que debemos utilizar otro tipo de componentes más orientados a presentación para ayudarnos a simplificar las páginas, e incluso, proporcionar librerías de funciones para simplificar la labor a otros programadores menos avanzados. Estos componentes son las denominadas librerías de etiquetas o TagLibs, con el objetivo de encapsular lógica o tareas.

Las TagLibs o librería de etiquetas nos permiten crear nuevas etiquetas, que podemos incluir en nuestros JSPs.

Este tipo de alternativa para incluir contenido dinámico y llamadas a lógica tiene las siguientes propiedades:

- Mejora la separación de la lógica y presentación
- Mejora la reusabilidad
- Encapsula la lógica de negocio
- Mejora la sintaxis para que los diseñadores Web incorporen fácilmente

Ya existe un conjunto de tagLibs definidos por organizaciones o particulares y otro estándar para poder utilizar. Entre estos destacamos:

- JSTL (Tags para obtener y establecer atributos, flujos de control, iteraciones, etc.)
- Struts, está formado por cuatro librerías diferentes:
    - Bean: Tags para acceso a los bean y sus propiedades.
    - HTML: Formulario puente entre las JSPs y otros componentes.
    - Logic: Con estos tag se puede eliminar el uso de scriptlets.
    - Nested: Tags que permiten relacionarse con otros tags anidados.

1.2.2.8.2 Controlador

El controlador en la plataforma Java 2 Enterprise Edition (J2EE[9]) se desarrolla mediante servlets, que hacen de intermediarios entre la vista y el modelo, más versátiles que los JSP para esta función al estar escritos como clases Java normales, evitando mezclar código visual (HTML, XML...) con código Java. Para facilitar la implementación de estos servlets también existe una serie de frameworks que proporcionan soporte a los desarrolladores.

Este componente recibe todas las peticiones realizadas a la aplicación. Cada petición se identifica mediante un parámetro. En base a esta identificación, el Controlador decide qué objeto u objetos de negocio (Modelo) debe ejecutar para resolver la petición.

Tras la ejecución de los objetos de negocio, y en función del resultado devuelto por estos, el Controlador determina qué JSP se usará para visualizar el resultado, generando una redirección que concluirá con la generación del código HTML de la página.

1.2.2.8.3 La clase Action

La clase Action define dos métodos que podrían ser ejecutados dependiendo de nuestro entorno servlet:

public ActionForward perform(ActionMapping mapping, ActionForm form, ServletRequest request, ServletResponse response) throws IOException, ServletException;

public ActionForward perform(ActionMapping mapping, ActionForm form, HttpServletRequest request, HttpServletResponse response) throws IOException, ServletException;

La mayoría de los proyectos sólo usarán la versión "HttpServletRequest".

El objetivo de una clase Action es procesar una solicitud, mediante su método perform(), y devolver un objeto ActionForward que identifica dónde se debería reenviar el control (por ejemplo a una JSP) para proporcionar la respuesta apropiada. En el patrón de diseño MVC, una clase Action típica implementará una lógica en su método perform() realizando las siguientes operaciones:

- Validar el estado actual de la sesión del usuario.

[9] La Plataforma J2EE define un conjunto de estándares para el desarrollo de aplicaciones empresariales en múltiples capas.

- Si la validación no se ha completado, valida las propiedades del formulario bean según sea necesario. Si se encuentra un problema, almacena las claves de los mensajes de error apropiados como un atributo de la petición, y reenvía el control de vuelta al formulario de entrada para que se puedan corregir los errores.
- Realizar el procesamiento requerido para tratar con esta solicitud (como grabar una fila de la base de datos). Esto se puede hacer mediante código lógico embebido dentro de la propia clase Action, pero generalmente debería realizarse llamando a un método apropiado del bean de lógica de negocio.
- Actualizar los objetos del lado del servidor que serán usados para crear la siguiente página del interface de usuario (normalmente beans del ámbito de solicitud o de sesión, dependiendo de cuánto tiempo necesitemos mantener estos ítems disponibles).
- Devolver un objeto ActionForward apropiado que identifica la página JSP usada para generar esta respuesta, basada en los beans actualizados recientemente. Típicamente adquiriremos una referencia a dicho objeto llamando a findForward() o al objeto ActionMapping que recibimos (si estamos usando un nombre lógico normal para este mapeo), o en el propio servlet controlador (si estamos usando un nombre lógico global para la aplicación).

1.2.2.8.4 El archivo struts-config.xml

El descriptor struts-config.xml configura el marco de trabajo Struts para formularios, acciones, controladores, mensajes y plug-ins, como se muestra a continuación:

- **<form-beans>**

```
<form-beans>
  <form-bean  name="logonForm"     type="paq1. paq2. paqN.LogonForm"/>
</form-beans>
```

La primer sección del descriptor se encuentra compuesta por los elementos **<form-beans>**, aquí son definidos todos aquellos Java Beans[10] que serán utilizados en la aplicación, la definición se encuentra compuesta por el nombre de la Clase Java así como un nombre corto que recibirá el Bean.

El elemento **<form-beans>** tiene los siguientes atributos importantes:

- **name**: Un identificador único para este bean, que será usado para referenciarlo en los correspondientes mapeos de acciones. Normalmente, es también el nombre del atributo de solicitud o sesión bajo el que se almacena este bean de formulario.
- **type**: El nombre totalmente cualificado de la clase Java de nuestro bean de formulario.

- **<global-forwards>**

```
<global-forwards>
  <forward   name="exito"         path="/beanstruts/archivo1.jsp"/>
  <forward   name="rechazo"       path="/beanstruts/archivo2.jsp"/>
</global-forwards>
```

La sección **<global-forwards>** como su nombre lo indica, es una manera de re-direccionar solicitudes dentro de toda la estructura del proyecto, esto es, basado en los parámetros anteriores, si en alguna sección de Struts se invoca el nombre éxito la solicitud será redireccionada al archivo jsp correspondiente.

- **<action-mappings>**

```
<action-mappings>
    <action    path="/logon"  type="paq1. paq2. paqN.LogonAction"  name="logonForm">
    </action>
</action-mappings>
```

[10] Un Java Bean es una manera de modularizar el uso datos en una aplicación con JSP's/Servlets a través de una Clase, su característica primordial es el uso de los métodos get y set los cuales permiten el acceso a los valores del Bean.

Esta sección contiene nuestras definiciones de acciones. Usamos un elemento <action> por cada una de nuestras acciones que queramos definir. Cada elemento action requiere que se definan los siguientes atributos:

- **path:** El path a la clase action en relación al contexto de la aplicación.
- **type:** El nombre totalmente cualificado de la clase Java de nuestra clase Action.
- **name:** El nombre de nuestro elemento <form-bean> para usar con esta action.

- **<controller>**

```
<controller>
   <set-property property="inputForward" value="true"/>
 </controller>
```

Después se encuentra la sección <controller> empleada para definir parámetros específicos del Controlador (org.apache.struts.action.ActionServlet); en este caso la declaración únicamente toma sus valores "default".

- **<message-resources>**

```
<message-resources  parameter="com.osmosislatina.struts.ApplicationResources"/>
```

Finalmente son definidos los ApplicationResources de la aplicación, es decir, los ficheros que contienen las claves y los mensajes de la aplicación web, los cuales contienen valores estáticos que pueden ser utilizados a lo largo de una aplicación Struts.

1.2.2.8.5 ActionMapping

Para poder operar satisfactoriamente, el servlet controlador Struts necesita conocer varias cosas sobre cómo se debería mapear toda URI solicitada a una clase Action apropiada. El conocimiento requerido ha sido encapsulado en un interface Java, llamado ActionMapping, estas son las propiedades más importantes:

- type - nombre totalmente cualificado de la clase Java que implementa la clase Action usada por este mapeo.
- name - El nombre del bean de formulario definido en el fichero de configuración que usará este action.
- path - El path de la URI solicitada que corresponden con la selección de este mapeo.
- unknown - Seleccionado a true si este action debería ser configurado como por defecto para esta aplicación, para manejar todas las solicitudes no manejadas por otros action. Sólo un Action puede estar definido como por defecto dentro de una sola aplicación.
- validate - Seleccionado a true si se debería llamar al método validate () de la action asociada con este mapeo.
- forward - El path de la URI solicitada a la que se pasa el control cuando se ha invocado su mapeo. Esto es una alternativa a declarar una propiedad type.

1.2.2.9 Modelo

El modelo, conteniendo lógica de negocio, es modelado por un conjunto de clases Java, existiendo dos claras alternativas de implementación, utilizando objetos java tradicionales llamados POJOs (Plain Old Java Objects) o bien utilizando EJB (Enterprise JavaBeans[11]) en sistemas con unas mayores necesidades de concurrencia o distribución.

[11] Los EJBs proporcionan un modelo de componentes distribuido estándar del lado del servidor. El objetivo de los EJBs es dotar al programador de un modelo que le permita abstraerse de los problemas generales de una aplicación empresarial.

1.2.2.9.1 Action Forms

Una de las tareas que durante el desarrollo de una aplicación insume mucho trabajo es la interacción con formularios, ya sea para editar u obtener nueva información. Las comprobaciones, la gestión de errores, el volver a presentarle el mismo form al usuario con los valores que puso y los mensajes de error y un largo etcétera están soportadas por Struts.

La idea es la siguiente: todo el trabajo de comprobaciones y generación de mensajes de error se implementa en los ActionForm y todo el trabajo de generación de interfaz en la(s) JSP.

1.2.2.9.2 Validación automática en formularios

Struts ofrece una facilidad adicional para validar los campos de entrada que ha recibido. Para utilizar esta característica, sobrescribimos el siguiente método en nuestra clase ActionForm:

public ActionErrors validate(ActionMapping mapping, HttpServletRequest request);

El método validate () es llamado por el servlet controlador después de que se hayan rellenado las propiedades del bean, pero antes de que se llame al método perform () correspondiente de la clase Action. El método validate () tiene las siguientes opciones:

- Realiza las validaciones apropiadas y no encuentra problemas, por lo tanto devuelve null o ejemplares de ActionErrors de longitud cero, y el servlet controlador procederá a llamar al método perform () de la clase Action apropiada.
- Realiza las validaciones apropiadas y encuentra problemas, devolviendo un ejemplar de ActionErrors conteniendo ActionError's, que son clases que contienen las claves del mensaje de error que se deberían mostrar. El servlet controlador almacena este arreglo como un atributo de la solicitud disponible para usarse por la etiqueta <html:errors>, y devolverá el control al formulario de entrada.

Como se mencionó anteriormente, esta característica es totalmente opcional. La implementación por defecto de validate() devuelve null, y el servlet controlador asumirá que no se requiere que se haga ninguna validación por parte de la clase Action.

Una aproximación común es realizar validaciones iniciales usando el método validate(), y luego manejar la validación de la "lógica de negocio" desde el objeto Action.

1.3 MYSQL

MySQL en los últimos años ha tenido un crecimiento vertiginoso. Es la base de datos de código abierto más popular del mundo. Código abierto significa que todo el mundo puede acceder al código fuente, es decir, al código de programación de MySQL, esto significa que también todos pueden contribuir con ideas, elementos, mejoras o sugerir optimizaciones. Y así es que MySQL ha pasado de ser una pequeña base de datos a una completa herramienta. Su rápido desarrollo se debe en gran medida a la contribución de mucha gente al proyecto, así como la dedicación del equipo de MySQL.

A diferencia de los proyectos propietarios, en los que el código fuente es protegido y desarrollado por un número reducido de personas, los proyectos de código abierto no excluyen a nadie interesado en aportar ideas, si disponen de los conocimientos necesarios.

Lo que en un tiempo se consideró como un sencillo juguete para uso en sitios Web, se ha convertido en la actualidad en una solución viable y de misión crítica para la administración de datos.

MySQL es un sistema de administración de bases de datos relacional (RDBMS). Se trata de un programa capaz de almacenar una enorme cantidad de datos de gran variedad y de distribuirlos para cubrir las necesidades de cualquier tipo de organización, desde pequeños establecimientos comerciales a grandes empresas y organismos administrativos. MySQL es la base de datos de código libre más popular, es un competidor cada vez más directo de los gigantes en esta materia como Oracle o SQL Server de Microsoft[11].

MySQL incluye todos los elementos necesarios para instalar el programa, preparar diferentes niveles de acceso de usuario, administrar el sistema y proteger los datos.

Puede desarrollar sus propias aplicaciones de bases de datos en la mayor parte de lenguajes de programación utilizados en la actualidad y ejecutarlos en casi todos los sistemas operativos. MySQL utiliza el lenguaje de consulta estructurado SQL.

Antes MySQL se consideraba como la opción ideal de sitios web; sin embargo, ahora incorpora muchas de las funciones necesarias para otros entornos y conserva su gran velocidad. MySQL es una base de datos robusta que se la puede comparar con una base de datos comercial, es

incluso más veloz en el procesamiento de las transacciones y dispone de un sistema de permisos elegante y potente, y ahora, además, incluye un motor de almacenamiento InnoDb[12] compatible con ACID[13], además dispone de procedimientos almacenados, triggers, vistas.

Existen muchos tipos de bases de datos, desde un simple archivo hasta sistemas relacionales orientados a objetos. MySQL, como base de datos relacional, utiliza múltiples tablas para almacenar y organizar la información.

MySQL es rápido, y una solución accesible para administrar correctamente los datos de una empresa. MySQLAB es la compañía responsable del desarrollo de MySQL, dispone de un sistema de asistencia eficiente y a un precio razonable, y, como ocurre con la mayor parte de las comunidades de código abierto, se puede encontrar una gran cantidad de ayuda en la Web.

1.3.1 Almacenamiento de Datos

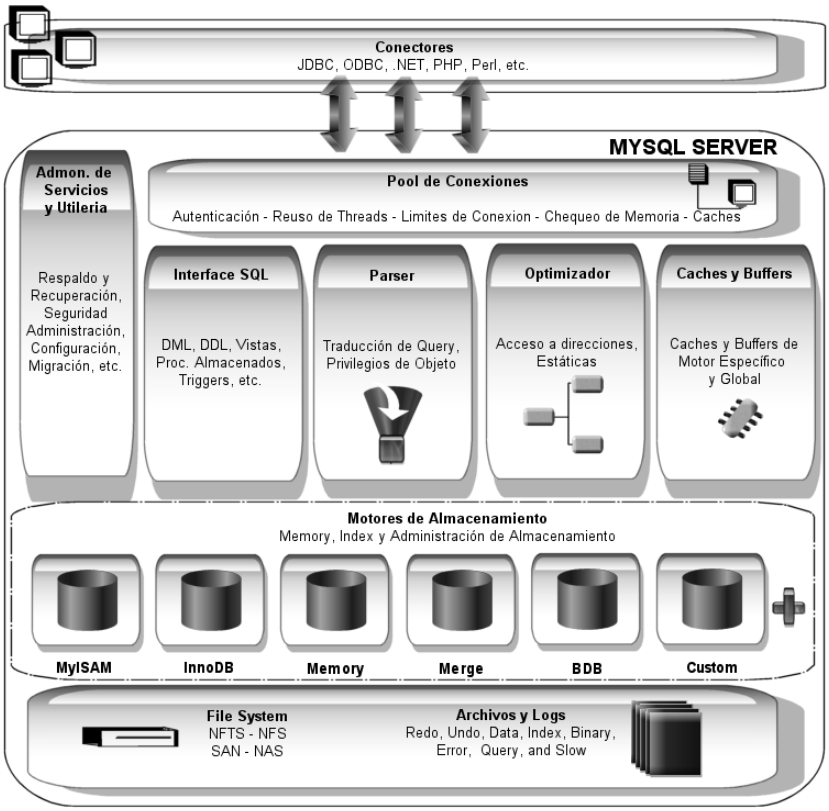

Figura 1.9. Almacenamiento de Datos MYSQL

Esta eficiente arquitectura (véase figura 1.9) provee beneficios para el tipo de aplicación que se necesite como data warehousing[14], procesamiento de transacciones, situaciones de alta disponibilidad, etc. Todo para que el motor de la base de datos funcione eficientemente. Esto va de la mano con las ventajas de utilizar un set de interfaces y servicios.

El programador de aplicaciones y el administrador de base de datos pueden interactuar con la base de datos a través de los conectores Api's y capas de servicios que están disponibles para los tipos de almacenamiento. Es posible cambiar entre los tipos de almacenamiento sin necesidad de modificar mucho código o procesos durante el cambio.

---

[12] InnoDB es un tipo de tabla de Mysql que permite trabajar con transacciones y definir reglas de integridad referencial.

[13] ACID son las propiedades que una base de datos debe cumplir para que el Sistema administrador de base de datos (DBMS) maneje correctamente la transaccionalidad, el término ACID viene de Atomicidad, Consistencia, Aislamiento y Durabilidad.

[14] Data warehousing es un proceso, no un producto. Es una técnica para consolidar y administrar datos de variadas fuentes con el propósito de responder preguntas de negocios y tomar decisiones, de una forma que no era posible hasta ahora.

1.3.1.1 Ingeniería de almacenamiento MySQL (Pluggable Storange Engine)

Es el componente de MySQL responsable de administrar las operaciones de entrada/salida de la base de datos, así como de habilitar y reforzar ciertas características de acuerdo a la particular aplicación, esto hace más eficiente y alto al performance de la base de datos. Esta es una de las razones por las que MySQL es ahora conocido por tener alto performance comparado con las bases de datos propietarias.

Desde una perspectiva técnica, los componentes de almacenamiento son:

- **Concurrencia.** Algunas aplicaciones requieren de estrategias de bloqueos más rígidas que otros (por ejemplo, bloqueo a nivel de fila). Seleccionar la correcta estrategia de bloqueo puede reducir en gran medida la saturación del sistema y ayuda en mucho al performance de la base de datos.
- **Soporte de transacciones.** No todas las aplicaciones necesitan transaccionalidad, pero para las que lo usan, está muy bien definido la característica como ACID.
- **Integridad referencial.** La necesidad de tener en un servidor de base de datos relacional integridad referencial a través de claves foráneas y del lenguaje de definición de datos (DDL[15]).
- **Almacenamiento Físico**, esto abarca todo lo que tiene que ver con el tamaño de página para tablas e índices, así como el formato usado para guardar los datos físicamente en disco.
- **Soporte de indexamiento.** Los diferentes escenarios en las aplicaciones, se benefician de las diferentes estrategias de índices, y cada uno de los tipos de almacenamiento tiene generalmente sus propios métodos de indexamiento.
- **Caché de Memoria.** Las diferentes aplicaciones responde mejor de acuerdo a la estrategia de almacenamiento en caché que utilizan.

1.3.2 Tipos de tablas

Existen dos tipos de tablas de transacción segura (InnoDB y BDB). El resto (ISAM; MyISAM, MERGE y HEAP) no son de transacción segura. La elección del tipo de tabla adecuado puede afectar enormemente al rendimiento.

1.3.2.1 Tablas ISAM

ISAM son siglas de Indexed Sequential Access Method (Método de Acceso Secuencial Indexado), se trata de un método para almacenar información a la que se pueda acceder rápidamente. ISAM fue desarrollado originalmente por IBM y en la actualidad forma parte del almacenamiento básico de muchos sistemas de bases de datos, tanto relacionales como de otros modelos.

En un sistema ISAM, la información se organiza en registros compuestos por campos de tamaño fijo. Los registros se almacenan secuencialmente, inicialmente para acelerar el acceso en sistemas de cinta. Un conjunto secundario de ficheros dispersos (tablas hash[16]) conocidos como índices contienen «punteros» a los registros que permiten acceder a los registros individuales sin tener que buscar en todo el fichero. Este es el punto de partida para todos los modernos sistemas de bases de datos navegacionales, en los cuales los punteros que dirigen hacia otra información fueron almacenados dentro de los propios registros. El avance clave que posee ISAM es que los índices son pequeños y pueden ser buscados rápidamente, permitiendo a la base de datos acceder sólo a los registros que necesita. Modificaciones adicionales a la información no requieren de cambios a otra información, sólo a la tabla y los índices.

Las bases de datos relacionales pueden fácilmente ser construidas en una red ISAM con la adición de lógica para mantener la validez de los enlaces entre las tablas. Típicamente el campo usado como enlace, será indexado para su lectura rápida. Si bien es cierto que esto es más lento que simplemente almacenar el puntero relacionado a la información directamente en los registros, esto

[15] Un lenguaje de definición de datos (DDL, por sus siglas en inglés) es un lenguaje proporcionado por el sistema de gestión de base de datos que permite a los usuarios de la misma llevar a cabo las tareas de definición de las estructuras que almacenarán los datos así como de los procedimientos o funciones que permitan consultarlos.

[16] Las tablas hash, son una de las aplicaciones más extendidas de las funciones de hash, aceleran el proceso de búsqueda de un registro de información según una clave.

también significa que los cambios al orden físico de la información no requiere ninguna actualización de punteros, entonces éstos siguen siendo válidos.

ISAM es muy fácil de entender e implementar, puesto que básicamente consiste en el acceso directo y secuencial a un archivo en una base de datos. También es muy barato. El truco está en que cada máquina cliente debe administrar su propia conexión a cada archivo que accesa. Esto, a su tiempo, presenta la posibilidad de inserciones conflictivas a esos archivos, que a su vez causa una base de datos inconsistente. Esto es típicamente solucionado con la adición de una red cliente-servidor, que supervisa las solicitudes del cliente y se mantiene ordenando. Este es el concepto básico detrás de SQL, en el cual hay una capa de "clientes" sobre la subyacente capa de almacenamiento de datos.

Las tablas ISAM eran el estándar antiguo de MySQL. Estas fueron sustituidas por las tablas MyISAM en la versión 3.23. Por lo tanto, es probable que solo se encuentre este tipo de tablas si está tratando con bases de datos antiguas. La principal diferencia entre las dos, es que el índice de las tablas MyISAM es mucho más pequeño que el de las tablas ISAM, de manera que un SELECT con un índice sobre una tabla MyISAM utilizará mucho menos recursos del sistema, además las tablas ISAM presentaban limitaciones importantes como la no exportación de ficheros entre máquinas de distintas arquitecturas o que no podían usar más de 4 Gigabytes

1.3.2.2 Tablas MyISAM

Optimizada para sistemas operativos de 64 bits, permite ficheros de tamaños mayores que las ISAM. Los datos se almacenan en un formato independiente, lo que permite pasar tablas entre distintas plataformas. Los índices se almacenan en un archivo con la extensión".MYI" y los datos en otro archivo con extensión ".MYD". Ofrece la posibilidad de indexar campos BLOB[17] y TEXT. Además este tipo de tablas soportan el tipo de dato VARCHAR.

1.3.2.2.1 Tablas Estáticas

Se usan cuando la tabla no contiene columnas de longitud variable (VARCHAR, BLOB, o TEXT). Cada registro se almacena usando un número de bytes fijo.

De los tres formatos de almacenamiento MyISAM, el formato estático es el más simple y seguro (menos sujeto a corrupción). También es el más rápido de los formatos sobre disco. La velocidad proviene de la facilidad con que se encuentran los registros en el fichero de datos en disco: cuando se busca un registro basándose en un número de registro del índice, multiplica el número de registro por la longitud de registro. También, al escanear una tabla, es muy fácil leer un número constante de registro con cada operación de lectura de disco.

La seguridad se evidencia si su máquina falla mientras el servidor MySQL está escribiendo en un fichero MyISAM de formato fijo. En este caso, **myisamchk** puede determinar fácilmente dónde comienza cada registro y dónde acaba, así que usualmente puede recuperar todos los registros excepto los parcialmente escritos. Tenga en cuenta que los índices de tabla MyISAM siempre pueden reconstruirse basados en los registros de datos.

1.3.2.2.2 Tablas Dinámicas

El formato de almacenamiento dinámico se usa si una tabla MyISAM contiene alguna columna de longitud variable (VARCHAR, BLOB, o TEXT), o si la tabla se crea con la opción ROW_FORMAT=DYNAMIC.

Este formato es un poco más complejo ya que cada columna tiene una cabecera que indica la longitud. Un registro puede acabar en más de una localización cuando es largo como por ejemplo el resultado de una actualización.

Puede usar OPTIMIZE TABLE o **myisamchk** para desfragmentar una tabla. Si tiene columnas de longitud fija a las que accede o cambia frecuentemente en una tabla que también contenga alguna columna de longitud variable, puede ser buena idea mover las columnas de longitud variable a otras tablas para evitar fragmentación.

Características generales de tablas de formato dinámico:

[17] Los BLOB (Binary Large OBjects, grandes objetos binarios) son elementos utilizados en las bases de datos para almacenar datos de gran tamaño que cambian de forma dinámica.

- Todas las columnas de cadenas de carácteres son dinámicas excepto aquéllas con longitud menor a cuatro.
- Cada registro viene precedido por un bitmap que indica qué columnas contienen la cadena vacía (para columnas de cadenas) o cero (para columnas numéricas). Hay que tener en cuenta que esto no incluye columnas que contienen valores NULL. Si una columna de cadena de carácteres tiene una longitud de cero tras eliminar los espacios en blanco finales, o una columna numérica tiene un valor de cero, se marca en el bitmap y no se guarda en disco. Las cadenas no vacías se guardan como un byte de longitud más al de los contenidos de la cadena.
- Para tablas de longitud fija normalmente se necesita mucho menos espacio de disco.
- Cada registro usa sólo tanto espacio como necesita. Sin embargo, si un registro crece, se divide en tantos trozos como haga falta, resultando en una fragmentación de registro. Por ejemplo, si actualiza un registro con información que alarga la longitud del registro, el registro se fragmenta. En este caso, puede que tenga que ejecutar OPTIMIZE TABLE o **myisamchk -r** de vez en cuando para mejorar el rendimiento. Use **myisamchk -ei** para obtener estadísticas de tabla.
- Más difícil de reconstruir tras un fallo que las tablas de formato estático, ya que los registros pueden fragmentarse en varios trozos y puede faltar algún enlace ó fragmento.

1.3.2.2.3 Tablas Comprimidas

El formato de almacenamiento comprimido es de sólo lectura generado con la herramienta **myisampack**.

Todas las distribuciones MySQL incluyen por defecto **myisampack**. Los escaneos de tablas comprimidas son descomprimidos por **myisamchk**.

Las tablas comprimidas tienen las siguientes características:

- Las tablas comprimidas ocupan muy poco espacio. Esto minimiza el uso de disco, lo que es útil al usar discos lentos como CD-ROMs.
- Cada registro se comprime por separado, así que hay poca sobrecarga de acceso. La cabecera de un registro ocupa de 1 a 3 bytes en función del registro más grande en la tabla. Cada columna está comprimida de forma distinta. Usualmente hay un árbol de Huffman[18] para cada columna. Algunos de los tipos de compresión son:
    - Compresión espacial de sufijo.
    - Compresión espacial de prefijo.
    - Números con valor de cero se almacenan usando un bit.
    - Si los valores de una columna entera tienen un rango pequeño, la columna se almacena usando el tipo menor posible. Por ejemplo, una columna BIGINT (ocho bytes) puede almacenarse como columna TINYINT (un byte) si todos los valores están en el rango de -128 a 127.
    - Si una columna tiene sólo un pequeño conjunto de valores posibles, el tipo de columna se convierte a ENUM[19].
    - Una columna puede usar cualquier combinación de los tipos de compresión precedentes.
- Pueden tratar registros de longitud fija o variable.

1.3.2.3 Tablas Merge

El motor de almacenamiento MERGE, también conocido como MRG_MyISAM, es una colección de tablas MyISAM idénticas que pueden usarse como una. “Idéntica” significa que todas las tablas tienen información de columna e índice idéntica. No puede mezclar tablas en que las columnas se listen en orden distinto, no tengan exactamente las mismas columnas, o tengan los índices en orden distinto. Sin embargo, alguna o todas las tablas pueden comprimirse con **myisampack**.

Cuando crea una tabla MERGE, MySQL crea dos ficheros en disco. Los ficheros tienen nombres que comienzan con el nombre de la tabla y tienen una extensión para indicar el tipo de

[18] El árbol de Huffman es una de las técnicas más básicas en la compresión de datos.

[19] ENUM es un campo que puede tener un único valor de una lista que se especifica. El tipo Enum acepta hasta 65535 valores distintos.

fichero. Un fichero .FRM almacena la definición de tabla, y un fichero .MRG contiene los nombres de las tablas que deben usarse como una. Las tablas no tienen que estar en la misma base de datos que la tabla MERGE misma.

Puede usar SELECT, DELETE, UPDATE, e INSERT en la colección de tablas, debe tener permisos de SELECT, UPDATE, y DELETE en las tablas que mapea a una tabla MERGE.

Si se hace un DROP de la tabla MERGE, sólo se borra la especificación MERGE. Las tablas subyacentes no se ven afectadas.

Cuando se crea una tabla MERGE, se debe especificar una cláusula UNION=(list-of-tables) que indica qué tablas quiere usar como una. Se puede especificar opcionalmente una opción INSERT_METHOD si quiere que las inserciones en la tabla MERGE se realicen en la primera o última tabla de la lista UNION. Use un valor de FIRST o LAST para hacer que las inserciones se hagan en la primera o última tabla, respectivamente. Si no especifica una opción INSERT_METHOD o si la especifica con un valor de NO, los intentos de insertar registros en la tabla MERGE producen un error.

1.3.2.4 Tablas MEMORY

El motor de almacenamiento MEMORY crea tablas con contenidos que se almacenan en memoria. Éstas se conocían previamente como HEAP. En MySQL 5.0, MEMORY es el término preferido, aunque HEAP se soporta para compatibilidad con versiones anteriores.

Cada tabla MEMORY está asociada con un fichero de disco. El nombre de fichero comienza con el nombre de la tabla y tiene una extensión de .FRM para indicar que almacena la definición de la tabla.

Como indica su nombre, las tablas MEMORY se almacenan en memoria y usan índices hash[20] por defecto. Esto las hace muy rápidas, y muy útiles para crear tablas temporales. Sin embargo, cuando se apaga el servidor, todos los datos almacenados en las tablas MEMORY se pierden. Las tablas por sí mismas continúan existiendo ya que sus definiciones se almacenan en ficheros .frm en disco, pero están vacías cuando reinicia el servidor.

1.3.2.5 Tablas InnoDB

InnoDB dota a MySQL de un motor de almacenamiento transaccional [11] (conforme a ACID) con capacidades de commit (confirmación), rollback (cancelación) y recuperación de fallas. InnoDB realiza bloqueos a nivel de fila y también proporciona funciones de lectura consistente sin bloqueo al estilo Oracle en sentencias SELECT. Estas características incrementan el rendimiento y la capacidad de gestionar múltiples usuarios simultáneos. No se necesita un bloqueo escalado en InnoDB porque los bloqueos a nivel de fila ocupan muy poco espacio. InnoDB también soporta restricciones FOREIGN KEY. En consultas SQL, aún dentro de la misma consulta, pueden incluirse libremente tablas del tipo InnoDB con tablas de otros tipos.

InnoDB se diseñó para obtener el máximo rendimiento al procesar grandes volúmenes de datos. Probablemente ningún otro motor de bases de datos relacionales en disco iguale su eficiencia en el uso de CPU.

A pesar de estar totalmente integrado con el servidor MySQL, el motor de almacenamiento InnoDB mantiene su propio pool de almacenamiento intermedio para tener un cache de datos e índices en la memoria principal. InnoDB almacena sus tablas e índices en un espacio de tablas, el cual puede consistir de varios ficheros (o particiones disco). Esto difiere de, por ejemplo, el motor MyISAM, donde cada tabla se almacena empleando ficheros separados. Las tablas InnoDB pueden ser de cualquier tamaño, aún en sistemas operativos donde el tamaño de los ficheros se limita a 2GB.

En MySQL 5.0, InnoDB viene incluido por defecto en las distribuciones binarias. El instalador Windows Essentials configura a InnoDB como el tipo de base de datos MySQL por defecto en Windows.

InnoDB se utiliza en muchos grandes sitios de bases de datos que necesitan alto rendimiento. El famoso sitio de noticias de Internet Slashdot.org corre sobre InnoDB. Mytrix, Inc. almacena más de 1TB de datos en InnoDB, y otros sitios manejan una carga promedio de 800 inserciones y actualizaciones por segundo en InnoDB.

---

[20] Una función de hash es una función para resumir o identificar probabilísticamente un gran conjunto de información, dando como resultado un conjunto imagen finito generalmente menor.

1.3.2.6 Tablas BDB

Las tablas BDB procesan las transacciones de forma ligeramente diferente a las tablas Innodb. Si una persona está realizando una transacción sobre una tabla x, si esta transacción no está completa, ninguna persona podrá consultar los datos de esta tabla mientras la transacción no finalice.

El periodo de tiempo que puede significar al llevar a cabo esta consulta es demasiado. El hecho de que no se trate de una consulta de selección "rápida" en las tablas BDB significa que todas las transacciones que se pospongan pueden dar lugar a graves problemas de rendimiento.

Como en el caso de las tablas innodb, el modo predeterminado de AUTOCOMMIT=1. Esto significa que a menos que coloque sus cambios dentro de una transacción (comenzando con BEGIN), se completarán inmediatamente.

2. Análisis

2.1. Introducción

Una de las principales tareas que realizan en su desempeño diario los trabajadores de los departamentos jurídicos en los distintos ámbitos del gobierno, es la consulta de leyes, con la idea de fundamentar resoluciones y/o realizar alegatos sustentados en el marco que la ley les proporciona. Para realizar esta tarea actualmente es necesario contar con personal con un profundo conocimiento de las leyes y diversas herramientas electrónicas tales como distintas compilaciones de leyes, cd's interactivos, etcétera; además de contar con referencias bibliográficas actualizadas que describan las normas jurídicas en cuestión.

Con este panorama, es pertinente considerar otras herramientas que posibiliten obtener mejores condiciones de servicio de los departamentos jurídicos, al mismo tiempo se deben considerar las ventajas de poner el conocimiento de los mismos departamentos jurídicos a disposición de la sociedad para alcanzar mejores condiciones de vida de la población, teniendo como prioridad la seguridad de sus habitantes y que se cumpla con la normatividad que establece la ley.

La gran ventaja de los sitios Web es que tenemos una gran cantidad de información disponible pero el servicio de consultas de información legal, se ofrece actualmente a efectos informativos y tiene de forma expresa efectos limitados, ya que no se revisan documentos que puedan ser relevantes o no se dispone de información completa en ningún caso. Por defecto se busca en los boletines, convenios colectivos, o jurisprudencia dentro de un plazo previamente establecido por el mismo sitio.

Cuando se genera una búsqueda en los sitios existentes se despliega un listado de normas donde aparece la palabra deseada y se puede consultar el detalle de la misma sin ninguna ayuda en la navegación entre la información que fue resultado de la búsqueda, para una posterior comparación.

De igual forma, no se necesita un portal para abogados y profesionales del derecho que conozcan los títulos de toda norma para poder consultarla, se requiere información confiable, cuyos movimientos y actualizaciones sean registrados.

El producto a desarrollar será un instrumento tecnológico que facilite la búsqueda, consulta, organización, actualización y modificación de códigos, decretos, leyes, reglamentos, acuerdos, circulares y en general cualquier ordenamiento o disposición que se publique oficialmente.

Las ventajas que presenta este sistema se resumen en las siguientes:

- Aplicación Web – Por lo que cualquier persona con una computadora y acceso a Internet, así como con los permisos necesarios, tendrá la oportunidad de consultar las leyes o reglamentos que necesite.
- Centralización de información – Al hacer uso de un solo banco de datos se evita redundancia en la información y se genera una base de conocimientos.
- Acercamiento con la sociedad – Al satisfacer una necesidad de la misma (conocer leyes y reglamentos aplicables a sus personas).
- Mecanismos de auditoría – De manera que se pueda saber quien realiza cambios en las leyes y reglamentos dentro del sistema.

2.1.1. Objetivos

Desarrollar una Aplicación Web de consulta y administración de leyes basado en la metodología de Rational Unified Process (RUP).

2.1.2. Alcance

Crear un sitio web permitiendo el acceso a un usuario administrador que realice las operaciones de agregar, modificar y eliminar cualquier ordenamiento o disposición que se publique oficialmente. De igual forma, se creará la sección de búsqueda y consulta de acceso general, es decir cualquier usuario común o visitador podrá interactuar con la misma.

2.2. Documento de requerimientos del usuario (DRU)

Este documento da un panorama general de la problemática actual y esboza la estrategia de solución sin abarcar detalles técnicos.

El propósito de éste documento es recoger, analizar y definir las necesidades del cliente en un nivel muy general. El documento se centra en la funcionalidad requerida para los usuarios finales, así como en las tareas y gestiones que se esperan de cada participante en el proyecto.

Los detalles de cómo el sistema cubre los requerimientos se podrán observar en la especificación de los casos de uso, requerimientos e implementación.

El producto a desarrollar es un sistema de consulta y administración de leyes en línea; la finalidad es proporcionar información de dichas leyes y controlar el manejo de información en cuanto a la actualización y publicación de estas.

Los usuarios de este sistema serán la sociedad en general, así como los distintos departamentos jurídicos.

2.2.1. Sentencia del problema

El problema de	No contar con un sistema de software actualizado que permita reorganizar, actualizar o modificar de leyes y reglamentos vigentes en el ámbito jurídico, con información confiable y fidedigna.
afecta a	Las distintas áreas jurídicas.
El impacto es	Tener leyes y reglamentos no actualizados dentro del sistema, lo cual lleva a acuerdos y decretos susceptibles de error, y que por lo tanto no son viables para generar una base de conocimientos.
Una solución exitosa sería	Contar con un sistema automatizado donde se pueda contar con información fidedigna, en un formato estándar para su consulta, y de fácil uso. Además de contar con mecanismos de auditoría de información.

Tabla 2.1 Sentencia del problema

2.2.1.1. Necesidades clave de participantes y usuarios

Necesidad	Prioridad	Causa posible	Solución actual	Solución propuesta
Búsqueda	1	Fundamentar resoluciones de autoridad.	Consulta manual de libros (códigos administrativos. etc.)	Generar un sistema de cómputo que permita buscar en línea los códigos administrativos, etc. En el cuál se pueda realizar consultas desde una palabra hasta una frase.
Versatilidad en la búsqueda en cuanto a establecer parámetros de comparación entre leyes municipales, estatales y federales.	2	Delimitar la competencia de la autoridad municipal con la del estado, o de la federación.	No existe.	Permitir al sistema realizar búsquedas delimitadas por el ámbito de competencia de la clasificación de leyes, códigos, decretos, etc., definidos en el sistema; y a su vez efectuar la comparación de dichos ordenamientos.
Las disposiciones	3	Ahorro de tiempo y	No existe	Establecer vínculos

legales publicadas deberán de permitir establecer vínculos con otros artículos señalados en su propio texto; sean aquellos de la propia ley o de una diversa.		ampliar la visión técnica y legal requerida para encuadrar la resolución en el marco constitucional.		de referencias cruzadas dentro de la información proporcionada por el sitio.
Proporcionar seguridad en el acceso a la administración del sitio mediante nombres de usuarios y contraseñas.	4	La información publicada en el sitio debe de ser confiable; por ello, es necesario asignar responsabilidades en el manejo de la información.	No existe	Un módulo que permita el acceso a los usuarios registrados en el sitio.
Mantener la información disponible para ser publicada, actualizada, modificada o eliminada dentro del sitio.	5	Para asegurar el buen funcionamiento del sitio es necesario administrar la información en él.	No existe	Un modulo que permita administrar la información para agregar nuevos documentos, publicar nuevos documentos, así como eliminar y modificar los mismos.
Auditoria del sitio.	6	Con el fin de asignar responsabilidades de los cambios en la información del sitio	No existe	Un modulo que permita almacenar un historial de las acciones (ingreso, modificación, publicación, etc.) de los usuarios en el sistema.

Tabla 2.2 Necesidades clave de participantes y usuarios

2.3. Especificación de Requerimientos de Software (ERS)

2.3.1. Vista general del proyecto

Para tratar de solventar el problema que presentan los departamentos jurídicos se pretende desarrollar un sistema de búsqueda y consulta de normas jurídicas en línea; además de permitir la administración y la auditoria de la información publicada en el sitio.

Los beneficios que proporcionará este sitio se listan a continuación:

- **Banco Centralizado de Normas Jurídicas.** El sistema permitirá concentrar las diversas normas jurídicas (leyes, decretos, códigos, etcétera) y estas podrán ser consultadas vía web en el momento deseado.
- **Auditable**. El sistema contará con un control de acceso restringido; con lo cual las acciones de los usuarios, tales como: creación, modificación, eliminación de normas; serán almacenadas en un histórico que podrá ser consultado por el administrador del sitio.

- **Actualizable**. El sistema permitirá editar la información publicada; así como el registro de nuevas normas.
- **Fácil Acceso**. Ya que el sistema trabajará vía Web, podrá ser consultado por un gran número de usuarios.

2.3.2. Propósito

El propósito del documento de requerimientos de software es especificar a nuestros diseñadores y a nuestro cliente cómo nuestro sistema "SISTEMA DE ADMINISTRACIÓN Y CONSULTA DE LEYES VÍA WEB (ACLE)" funcionará.

2.3.3. Alcance

Se pretende implementar una herramienta que proporcione en forma centralizada leyes, reglamentos, decretos y demás ordenamientos normativos para su consulta, sirviendo de apoyo a los asuntos que la Dirección Jurídica o algún otro departamento jurídico conozca.

2.3.4. Definiciones, Acrónimos y Abreviaturas

Cliente: Cualquier ciudadano, servidor público o dependencia que solicite el desarrollo de un sistema de información.

Usuario: Cualquier ciudadano o servidor público que tenga acceso al sistema.

ACLE: Sistema de Administración y consulta de Leyes vía Web.

2.3.5. Referencias del Documento

RUP (Rational Unified Process).

2.3.6. Funciones de la Aplicación de Software

- Función Ingresar al sistema: permite a los usuarios del sistema ingresar al mismo (solo aplica para los administradores del sistema).
    - Controlar el acceso vía Usuario/Contraseña.

- Gestión de normas: permite agregar, eliminar, modificar o actualizar las leyes dentro del sistema.
    - Agregar norma.
    - Modificar norma.
    - Eliminar norma.

- Búsquedas: permite a los usuarios consultar por las leyes publicadas en este sitio.
    - Realizar búsquedas personalizadas.

- Control de calidad del sitio: permite a los usuarios realizar comentarios o sugerencias para la mejora del sitio.
    - Crear un buzón de sugerencias.

- Auditoria: permite al administrador del sistema conocer las acciones realizadas por las personas encargadas de agregar, eliminar, actualizar y modificar las leyes en el sitio.
    - Consultar acciones realizadas en la administración de normas.

2.3.7. Especificación de Requerimientos

El programa debe de permitir un total manejo de normas de manera electrónica, incluyendo la generación de consultas personalizadas.

2.3.8. Funcionalidad

- Control de acceso vía Usuario/Contraseña
  - **Introducción:** Se necesita ingresar al sistema de forma segura siempre y cuando sea necesario.
  - **Entradas:** Usuario y contraseña previamente asignados.
  - **Proceso:** Esta función debe, a partir de los datos de entrada, ser capaz de verificar la existencia del usuario, comprobar que la contraseña sea correcta.
  - **Salida:** Ingresar al sistema.

- Agregar norma
  - **Introducción:** El usuario ingresa el archivo respectivo a una norma.
  - **Entradas:** archivo, directorio, nombre del archivo, descripción, nivel, tipo de norma jurídica y estructura de la misma.
  - **Proceso:** Aquí se almacenan los datos de entrada de modo que se genere la estructura apropiada de dicha información.
  - **Salida:** Éxito de la operación.

- Modificar Norma
  - **Introducción:** Existe la necesidad de proporcionar una solución para los posibles errores de captura o actualizaciones.
  - **Entradas:** Norma en cuestión.
  - **Proceso:** Esta función debe, asociar la información correspondiente a la norma referida y tener opción de modificación.
  - **Salida:** Éxito de la operación.

- Eliminar Norma
  - **Introducción:** Una de las necesidades, es la eliminación por distintos motivos de una norma y su información asociada.
  - **Entradas:** Servicio en cuestión.
  - **Proceso:** Esta función debe, borrar la solicitud demandada y la información asociada a la misma.
  - **Salida:** Eliminación de la norma.

- Búsquedas personalizadas
  - **Introducción:** El sistema deberá ser capaz de realizar consultas personalizadas de las normas almacenadas.
  - **Entradas:** Consulta personalizada.
  - **Proceso:** Se ejecuta la consulta.
  - **Salida:** Información resultado de ejecutar la consulta.

- Buzón de sugerencias
  - **Introducción:** Se necesita contar con un buzón electrónico para que las personas que ingresen al sitio puedan retroalimentar el mismo.
  - **Entradas:** Comentario o sugerencia sobre el sitio.
  - **Proceso:** Los datos de entrada se almacenarán en la base de datos para ser consultados por los usuarios que administran el sistema.
  - **Salida:** Éxito de la operación.

- Administración de Usuarios
  - **Introducción:** Permite la creación, modificación y eliminación de un usuario para el acceso al sistema.
  - **Entradas:** Recurso Humano.
  - **Proceso:** Esta función debe, asignar un usuario y contraseña a un determinado recurso humano para acceder al sistema.
  - **Salida:** Éxito de la operación.

- Auditoria
  - **Introducción:** Se necesita permitir la consulta de la información correspondiente a las acciones realizadas a cierta norma.
  - **Entradas:** norma en cuestión.
  - **Proceso:** Aquí en base a los datos de entrada se muestra a detalle los movimientos realizados a dicha norma.
  - **Salida:** Visualización de información relacionada con la norma seleccionada.

2.3.9. Usabilidad

- Requisitos del Sistema

Servidor de aplicaciones, en particular su contenedor de objetos java (OC4J), base de datos MySQL, sistema operativo indistinto.

Para el lado del cliente debe ser necesario contar con una PC con requerimientos mínimos de 64 Mb de memoria RAM, y un procesador Pentium a 166 MHz

- Capacitación

El usuario deberá de contar con conocimientos básicos en herramientas de navegación de Internet, con lo antes mencionado la duración de la capacitación será de 2 sesiones de 1 hora como máximo, la segunda sesión solo si su caso lo amerita.

- Estándares de Usabilidad

Se cumplen con los estándares de un sitio web.

2.3.10. Disponibilidad

El sistema tendrá disponibilidad de acceso los 7 días de la semana, las 24 horas, ya que éste será un desarrollo vía web.

El usuario puede copiar, y descargar todas y cada una de las normas, según lo desee, sin ninguna limitación o problema.

2.3.10.1. Fiabilidad del hardware

- Infraestructura en la conexión a Internet

Para que el sistema tenga un desempeño eficiente es necesario contar con una infraestructura de red segura y robusta.

- Pool de conexiones

El sistema dependerá de la configuración del pool de conexiones del servidor de Base de Datos.

2.3.10.2. Fiabilidad del software

- Administrador

La mala administración del sistema puede ocasionar el funcionamiento inadecuado de éste, ya que la especificación incorrecta de sus distintas funciones puede ocasionar fallos en los procesos del mismo.

- Operador

La calidad de la información es responsabilidad del cliente.

2.3.11. Mantenimiento

Planificando cuidadosamente el proyecto y adhiriendo el paradigma conocido "Model-View-Controller" se tendrá un programa que contará con un sencillo mantenimiento.

Se mantendrá contacto constante con el cliente en todas las partes del proceso de diseño del programa para evitar hacer cambios significativos en las últimas etapas del programa que podría ser difícil de entender.

2.3.12. Funcionamiento

El tiempo de las transacciones dependerá del ancho de banda con que cuente su infraestructura de red.

La degradación del sistema dependerá de las restricciones del Servidor de aplicaciones, en particular su contenedor de objetos java (OC4J) y la base de datos MySQL.

2.3.13. Soportabilidad

Un rango predefinido para todos los valores que son dados por el operador debe ser definido y el sistema verificará que todas las entradas del operador caen dentro de este.

2.3.14. Restricciones de diseño

- Estándares cumplidos

El producto debe cumplir las reglas de un sitio Web para poder garantizar así que el programa realmente se va a comportar correctamente.

La notación, definiciones y acrónimos utilizados son los estándares en la realización de un proyecto con metodología RUP.

- Limitaciones hardware

Las limitaciones hardware del producto son las limitaciones que posee un servidor de aplicaciones en particular su contenedor de objetos Java (OC4J), base de datos MySQL, los sistemas operativos indistintos.

2.3.15. Interfaz

- Interfaz de Usuario

En este apartado, vamos a comentar los procesos en el que el usuario interactúa con ACLE.

Primero la interacción visual, en pantalla aparecerá un formulario donde le pedirá su nombre de usuario y contraseña, en caso de ser usuario administrador; en caso contrario se mostrará otra pantalla en la que el usuario tiene la posibilidad de combinar todas y cada una de las opciones de búsqueda, donde los resultados de ésta serán mostrados en la siguiente ventana.

- Interfaz Hardware

Servidor de aplicaciones, en particular su contenedor de objetos java (OC4J), base de datos MySQL, sistema operativo indistinto.

Para el lado del cliente debe ser necesario contar con una PC con requerimientos mínimos de 64MB de memoria RAM y un procesador Pentium a 166 MHz con un sistema operativo compatible con los navegadores Internet Explorer o Mozilla (Netscape o Firefox).

- Interfaz de Software

El producto que estamos describiendo está desarrollado en Web y es multiplataforma, con el cual el usuario no tendrá ningún problema ya que la única restricción es que debe ser compatible con los navegadores, Internet Explorer y Mozilla (Netscape o Firefox). Donde toda la funcionalidad es un ambiente Web.

2.3.16. Características de los usuarios

Para proveer de una forma efectiva productos y servicios que se ajusten a las necesidades de los usuarios, es necesario identificar e involucrar a todos los participantes en el proyecto como parte del proceso de modelado de requerimientos.

También es necesario identificar a los usuarios del sistema y asegurarse de que el conjunto de participantes en el proyecto los representa adecuadamente. Esta sección muestra un perfil de los participantes y de los usuarios involucrados en el proyecto, así como los problemas más importantes que éstos perciben para enfocar la solución propuesta hacia ellos.

A continuación vamos a ver qué usuarios van a usar el producto y como afectan sus funciones al producto.

Se considera como un usuario del sistema ACLE a la persona que demande un servicio o usuario administrador que tenga acceso al sistema, esta persona debe conocer el medio y manejo de la aplicación.

También son usuarios las personas que realizaran el mantenimiento del producto, a la que les debe resultar sencillo de configurar, instalar y reinstalar.

- Obligaciones Generales

Es mandatario, para mantener la eficiencia del sistema, mantener los catálogos con información adecuada al mismo. Así mismo se obliga a mantener el software inicial u otro equivalente en las mismas condiciones al momento de la puesta en marcha del sistema ACLE.

- Demografía de Mercado

La sociedad, así como todos los departamentos jurídicos pertenecientes serán los principales usuarios del sistema.

- Resumen de Participantes

Nombre	Descripción	Responsabilidades
Desarrollo e Implementación de Software	Desarrollador	Análisis, diseño y desarrollo de un sistema capaz de gestionar solicitudes de servicio del cliente, en base a sus requerimientos.
Departamentos Jurídicos	Cliente	Enunciar requerimientos, desarrollar procesos, validar la propuesta, realizar pruebas de stress y funcionalidad.

Tabla 2.3 Resumen de participantes

- Resumen de Usuarios

Nombre	Descripción	Responsabilidades	Afectado
Ciudadanía	Usuario de consulta	Consulta de leyes y reglamentos Búsqueda de leyes y reglamentos	N/A
Áreas Jurídicas	Usuario de consulta	Consulta de leyes y reglamentos. Búsqueda de leyes y reglamentos.	Ciudadanía

Áreas Jurídicas	Usuario administrador del sistema, encargado de mantener información de calidad en el mismo.	Administración de leyes y reglamentos. Administración de usuarios. Consulta de leyes y reglamentos. Búsqueda de leyes y reglamentos.	Ciudadanía, Áreas jurídicas.

Tabla 2.4 Resumen de Usuarios

- Perfil de usuario

Descripción	Abogado, usuario final del sistema.
Tipo	Usuario de sistemas Microsoft (Microsoft Office), Pascal, IUS-2004
Responsabilidades	Reportar los beneficios y limitaciones que produzca el sistema; fomentando con esto el enlace con los desarrolladores del sistema. Capacitación a usuarios finales. Captura de información en el sistema. Actualización de la información manejada dentro del sistema.
Criterio de éxito	Rapidez en el manejo de la información. Amigable al usuario final. Confiable en cuanto a la información que se maneja en el sistema.
Involucrado en	Reportar los beneficios y limitaciones que produzca el sistema.
Entregables	Sistema ACLE, Manual del Usuario

Tabla 2.5 Perfil de usuario

2.3.17. Suposiciones y Dependencias

ACLE requiere de la definición de un administrador y de usuarios.

Al ser una aplicación Web su disponibilidad depende de la infraestructura de red, servidores y base de datos.

La lista de suposiciones y restricciones se incrementará durante el desarrollo del proyecto.

NOTA: La veracidad y la confiabilidad de la información publicada en el sitio es responsabilidad exclusiva de los usuarios encargados de administrar el sitio.

El presente sistema ACLE depende enteramente de un servidor de aplicaciones, en particular su contenedor de objetos java (OC4J), base de datos MySQL, sistema operativo indistinto, de preferencia UNIX (solaris).

2.3.18. Requerimientos de rendimiento

Para que el producto tenga un desempeño eficiente es necesario contar con una infraestructura de red segura y robusta.

2.3.19. Requerimientos tecnológicos

Servidor de aplicaciones, en particular su contenedor de objetos java (OC4J), base de datos Oracle, sistema operativo indistinto, de preferencia UNIX (Solaris).

3. Diseño

Especificación de Casos de Uso (ECU)

3.1 Caso de uso: Agregar Elemento

Breve descripción

Este caso de uso permite agregar un nuevo elemento (libro, título, capítulo, artículo, transitorio) a una norma previamente definida.

Actores

Usuario

Flujo de Eventos

Flujo Básico

1b El sistema muestra un formulario solicitando lo siguiente (véase Tabla 3.1):

Información del Articulo			
Nombre del Registro	**Descripción**	**Tipo de Dato**	**Dominio**
Título	Se refiere al título del elemento que se desea agregar.	Texto	No nulo, letras del alfabeto, números, signos de puntuación
Descripción	Se refiere a la descripción o contenido del elemento.	Texto	No nulo, letras del alfabeto, números, signos de puntuación

Tabla 3.1 Agregar Elemento

Y el botón “Aceptar”.

2b El usuario coloca el título del elemento como se solicita y pulsa en el botón “Aceptar”.
3b El sistema verifica que los campos cumplan con las especificaciones establecidas en el punto 1b y los almacena en la base de datos; si no existe conexión con la base de datos el sistema emite un mensaje de error.
4b El sistema pasa al caso de uso **“Modificar Norma”/”Agregar Norma”**.

Flujos Alternos

1a <<**1b**>> El usuario pulsa el botón “Cancelar”, el sistema pasa al caso de uso **“Agregar Norma”/”Modificar Norma”**.

Precondiciones

El usuario deberá ingresar al sistema por medio de un usuario y una contraseña.

Diagrama de Actividades

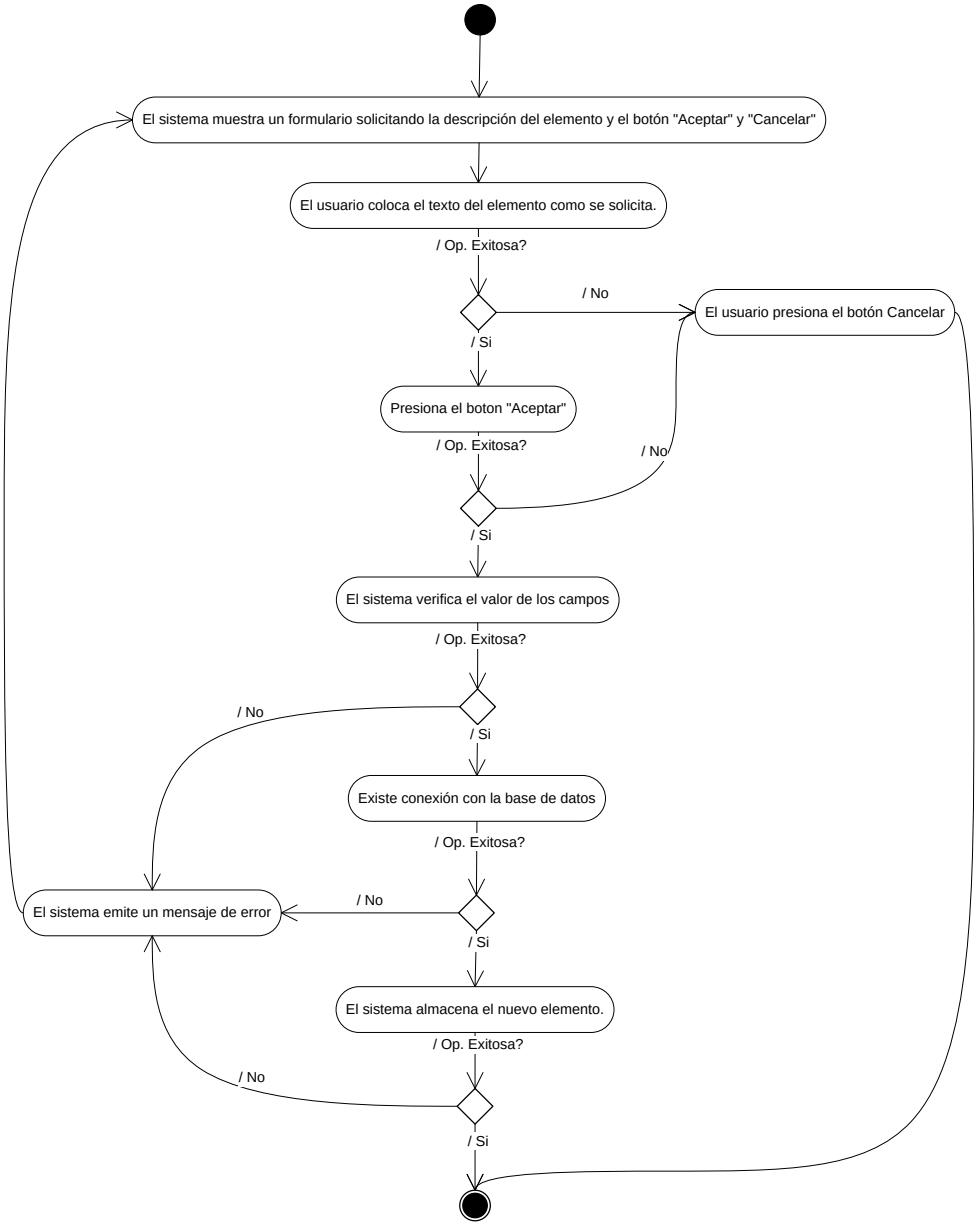

Figura 3.1 Diagrama de Actividades "Agregar Elemento"

Diagrama de secuencia

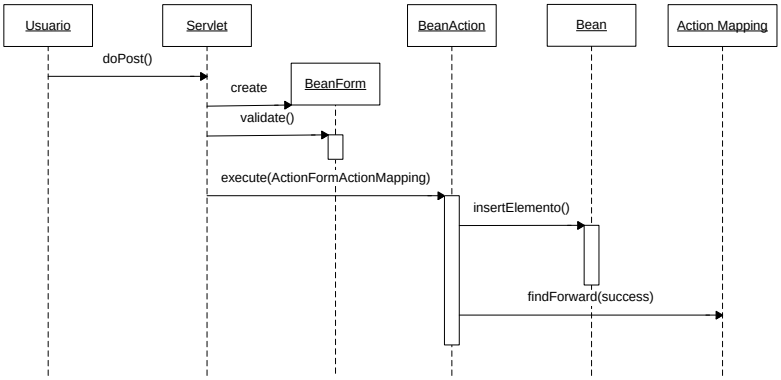

Figura 3.2 Diagrama de Secuencia "Agregar Elemento"

Diagrama de colaboración

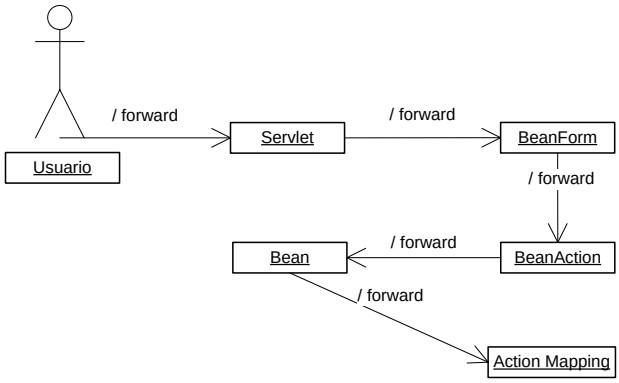

Figura 3.3 Diagrama de Colaboración "Agregar Elemento"

3.2 Caso de uso: Agregar Archivo

Breve descripción

Este caso de uso permite agregar un nuevo archivo a una norma previamente definido.

Actores

Usuario

Flujo de Eventos

1b El sistema muestra un formulario solicitando la ruta del archivo que se desea asociar a la norma, así como los botones "Examinar", "Regresar al Listado" y "Agregar Archivo".

2b El usuario selecciona el archivo deseado y presiona el botón "Agregar Archivo".

3b El sistema valida la ruta del archivo y la almacena en la base de datos; de igual forma se almacena el archivo en el servidor de aplicaciones para su posterior consulta; si no existe conexión con la base de datos el sistema emite un mensaje de error.

4b El sistema pasa al caso de uso **"Agregar Archivo"**.

Flujos Alternos

1a <<**2b**>> El usuario realiza la selección de otra operación pasando al caso de uso correspondiente.

2a <<**2b**>> El usuario presiona en la opción "Regresar al Listado", el sistema pasa al caso de uso **"Listar Elementos"**.

3a <<**3b**>> La ruta del archivo especificada no es correcta, el sistema emite un mensaje de error.

Precondiciones

El usuario deberá ingresar al sistema por medio de un usuario y una contraseña.

Diagrama de Actividades

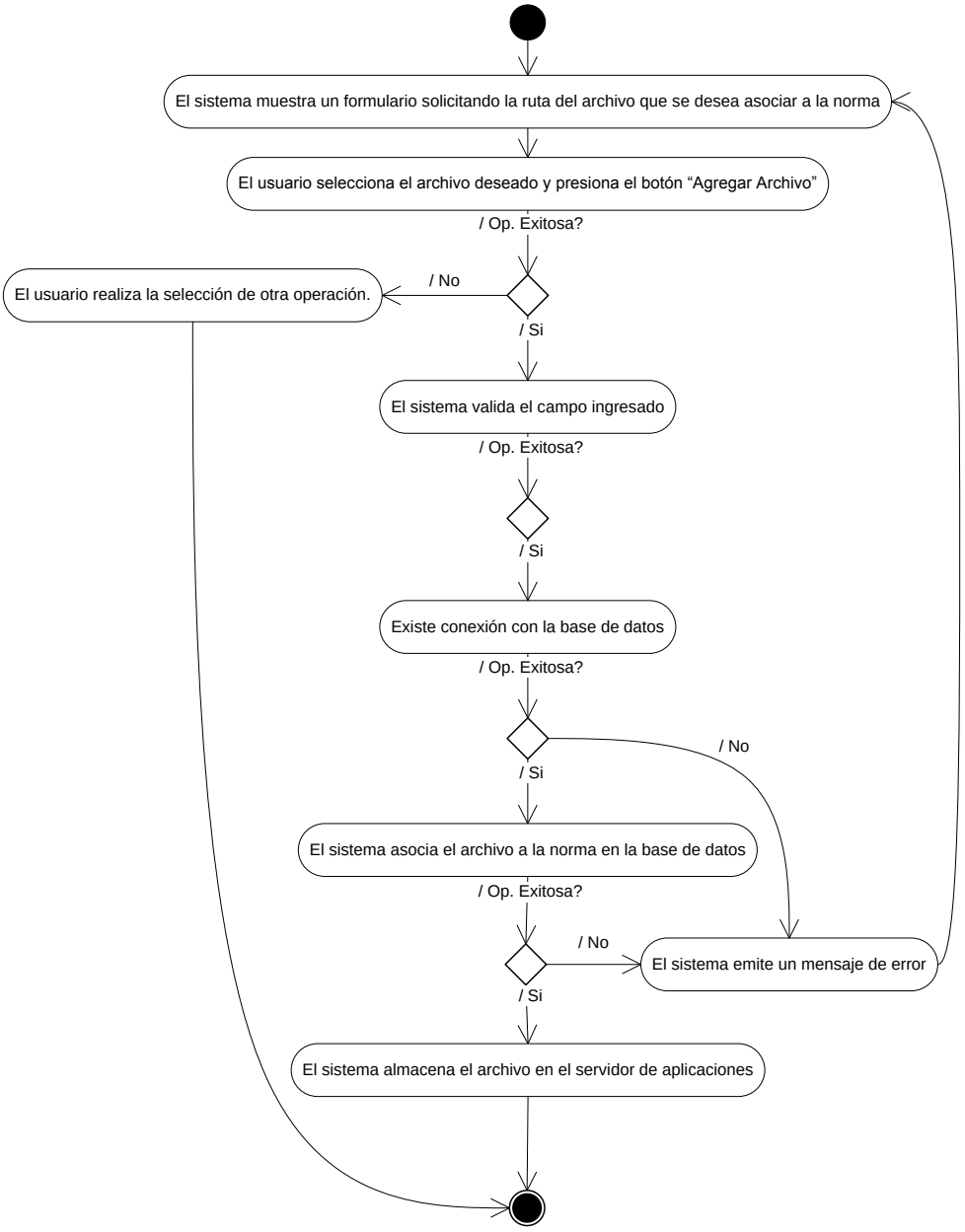

Figura 3.4 Diagrama de Actividades “Agregar Archivo”

Diagrama de Secuencia

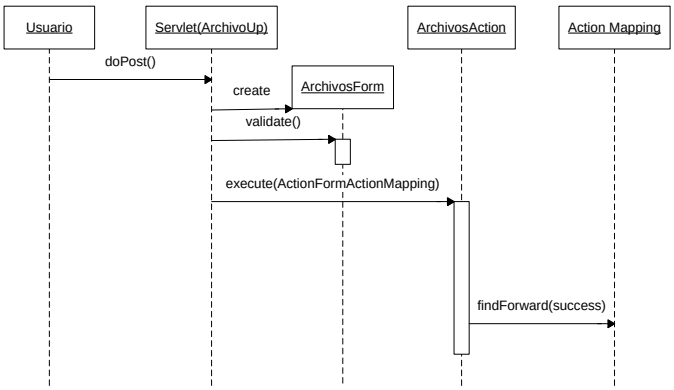

Figura 3.5 Diagrama de Secuencia “Agregar Archivo”

Diagrama de Colaboración

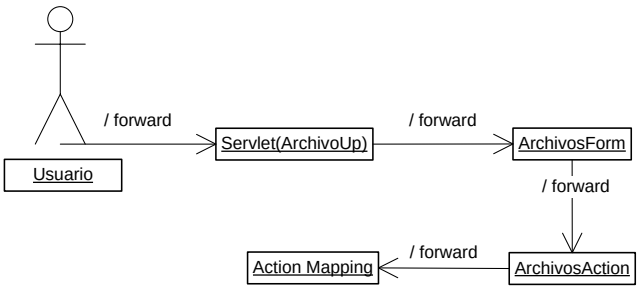

Figura 3.6 Diagrama de Colaboración "Agregar Archivo"

3.3 Caso de uso: Agregar Norma

Breve descripción

Este caso de uso permite agregar una nueva norma al sistema.

Actores

Usuario

Flujo de Eventos

Flujo Básico

1b El sistema muestra un formulario vacío solicitando los siguientes datos (ver tabla 3.2):

Datos de la Norma			
Nombre del Registro	**Descripción**	**Tipo de Dato**	**Dominio**
Fecha de Publicación	Se refiere a la fecha en la que se publicó la norma jurídica.	Numérico	No nulo.
Fecha de última reforma	Se refiere a la fecha en la que se realizó la última reforma a la norma jurídica.	Numérico	No nulo.
Nombre	Se refiere al título principal de la norma jurídica.	Texto	No nulo, letras del alfabeto y números, signos de puntuación.
Descripción	Se refiere a la descripción que acompaña al título principal de la norma jurídica.	Texto	No nulo, letras del alfabeto y números, signos de puntuación.
Archivo	Se refiere a la dirección local del archivo que contiene la norma jurídica.	Texto	Letras del alfabeto y/o números.
Tipo de Norma	Se refiere al tipo de norma jurídica a agregar al sistema	Numérico	No nulo.
Nivel	Se refiere al nivel de la norma jurídica que se desea agregar al sistema.	Numérico	No nulo.
Libro	Se refiere a el(los) libro(s) correspondiente(s) a la norma jurídica que se desea ingresar.	Texto	Letras del alfabeto y números, signos de puntuación.
Titulo	Se refiere a el(los) título(s) correspondiente(s) a la norma jurídica que se desea ingresar.	Texto	Letras del alfabeto y/o números, signos de puntuación.
Capitulo	Se refiere a el(los) capítulo(s) correspondiente(s) a la	Texto	Letras del

	norma jurídica que se desea ingresar.		alfabeto y/o números, signos de puntuación.
Articulo	Se refiere a el(los) artículo(s) correspondiente(s) a la norma jurídica que se desea ingresar.	Texto	Letras del alfabeto y/o números, signos de puntuación.
Transitorio	Se refiere a el(los) transitorio(s) correspondiente(s) a la norma jurídica que se desea ingresar.	Texto	Letras del alfabeto y/o números, signos de puntuación.

Tabla 3.2 Agregar Norma

Y los botones “Examinar”, “Subir Archivo”, “Agregar/Eliminar Archivo”, “Eliminar Archivos Seleccionados”, “Realizar Operación”, “Siguiente”, “Anterior”, “Cancelar” y “Terminar”.

2b El usuario llena los campos solicitados y pulsa en el botón “Siguiente”.
3b El sistema verifica que los campos de la primera parte de la captura cumplan con las especificaciones de dominio mostradas en la tabla del punto 1b.
4b El sistema muestra un listado de operaciones que se pueden realizar con libros, títulos, capítulos, artículos, transitorios y archivos para terminar de ingresar la norma jurídica deseada.
5b El usuario selecciona la operación deseada y pasa al caso de uso correspondiente.
6b El usuario llena los campos solicitados y termina pulsando en el botón “Terminar”.
7b El sistema agrega los datos de la norma en la base de datos; si no existe conexión con la base de datos el sistema emite un mensaje de error.
8b El sistema pasa al caso de uso **“Listar Elementos”.**

Flujo Alterno

1a <<**1b**>> El usuario pulsa en “Cancelar”, el sistema pasa al caso de uso **“Listar Elementos”**.
2a <<**3b**>> Los campos ingresados en la tabla del punto 1b no cumple con las especificaciones de dominio establecidos, el sistema muestra un mensaje informando al usuario los datos incorrectos o faltantes y regresa al punto 1b.
3a <<**4b**>> El usuario selecciona la operación “Agregar” junto con el elemento “Artículo”, “Capítulo”,”Título”,”Libro” ó “Transitorio” y presiona el botón “Realizar Operación”, el sistema pasa al caso de uso “**Agregar Elemento**”.
4a <<**4b**>> El usuario selecciona la operación “Modificar” junto con el elemento “Artículo”,“Capítulo”,”Título”,”Libro” ó “Transitorio” y presiona el botón “Realizar Operación”, el sistema pasa al caso de uso “**Modificar Elemento**”.
5a <<**4b**>> El usuario selecciona la operación “Eliminar” junto con el elemento “Artículo”, “Capítulo”,”Título”,”Libro” ó “Transitorio” y presiona el botón “Realizar Operación”, el sistema pasa al caso de uso “**Eliminar Elemento**”.
6a <<**4b**>> El usuario selecciona la operación “Agregar/Eliminar Archivos” y presiona el botón “Realizar Operación”, el sistema pasa al caso de uso “**Listar Elementos**”.

Requerimientos Especiales

El usuario deberá de reconocer de forma previa a la acción de “Agregar Norma”, la estructura de la norma a anexar en el sistema.

Precondiciones

El usuario deberá ingresar al sistema por medio de un usuario y una contraseña.

Diagrama de Actividades

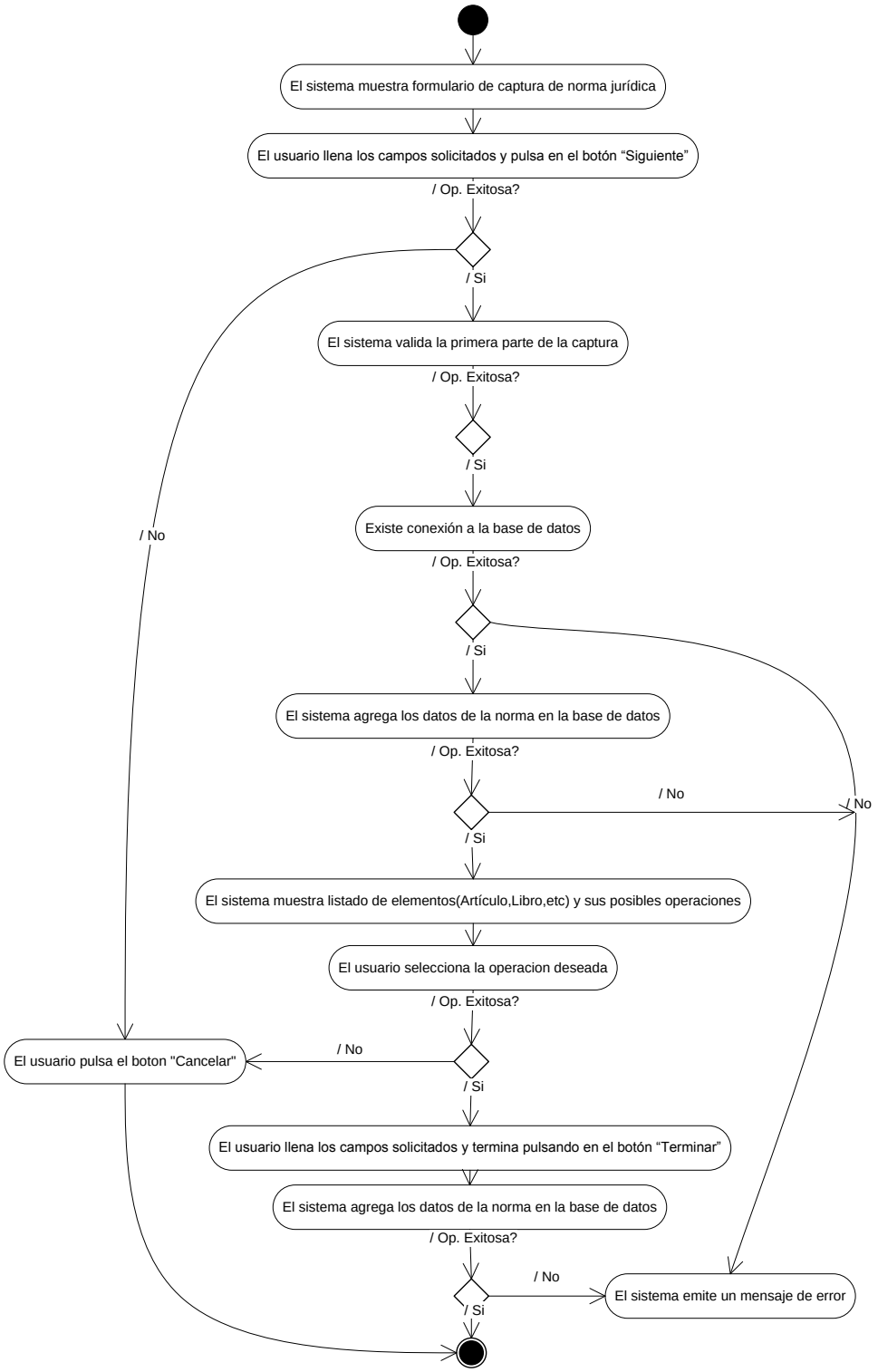

Figura 3.7 Diagrama de Actividades “Agregar Norma”

Diagrama de secuencia

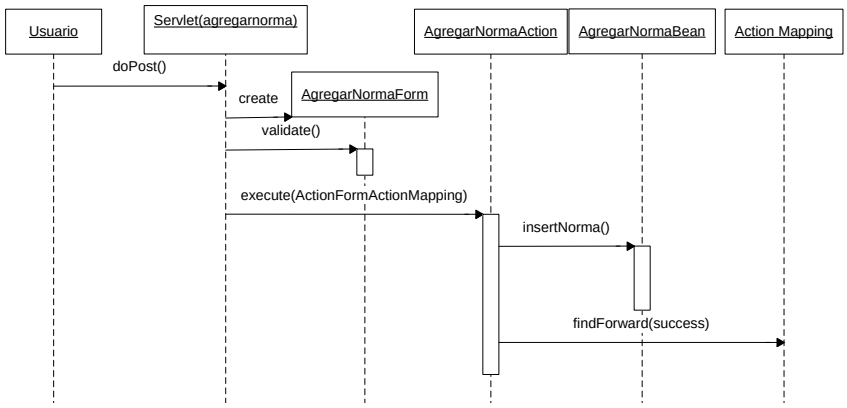

Figura 3.8 Diagrama de Secuencia "Agregar Norma"

Diagrama de colaboración

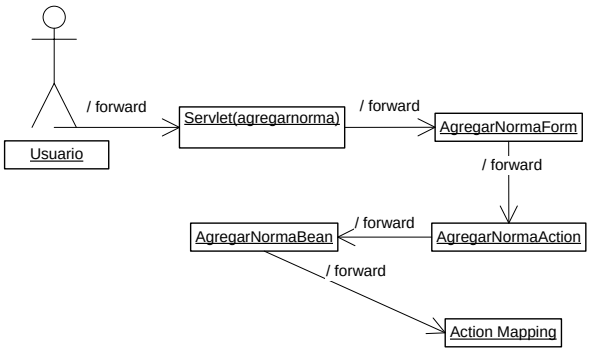

Figura 3.9 Diagrama de Colaboración "Agregar Norma"

3.4 Caso de uso: Agregar Sugerencia

Breve descripción

Este caso de uso permite al usuario agregar una recomendación o comentario para la mejora del sistema.

Actores

Usuario

Flujo de Eventos

Flujo Básico

1b El sistema muestra un formulario solicitando los siguientes datos:

Datos del comentario o Sugerencia"			
Nombre del Registro	**Descripción**	**Tipo de Dato**	**Dominio**
Correo Electrónico	La dirección de correo electrónico del usuario	Texto	No nulo, formato de email
Sugerencia	El texto de la sugerencia o recomendación	Texto	No nulo, letras del alfabeto, signos de puntuación.

Tabla 3.3 Agregar Sugerencia

Y los botones “Enviar” y “Cancelar”.

2b El usuario llena los campos requeridos en el formulario.
3b El usuario pulsa en el botón “Enviar”, el sistema verifica que los campos cumplan con las especificaciones de dominio establecidas en el punto 1b y muestra un mensaje de confirmación con una leyenda de confirmación y el botón “Aceptar”.
4b El usuario pulsa en el botón “Aceptar”, el sistema pasa al caso de uso **“Buscar Artículo”.**

Flujo Alterno

1a **<<1b>>** El usuario pulso el botón “Cancelar”, el sistema pasa al caso de uso **“Buscar Artículo”.**
2a **<<3b>>** El usuario pulsa el botón “Enviar”, el sistema muestra un mensaje indicando la razón por la que no se realiza la operación y el botón “Aceptar”.
2a.1 El usuario pulsa en el botón “Aceptar”.
2a.2 El sistema pasa al punto 1b.

Requerimientos Especiales

Ninguno.

Precondiciones

Ninguna.

Diagrama de Actividades

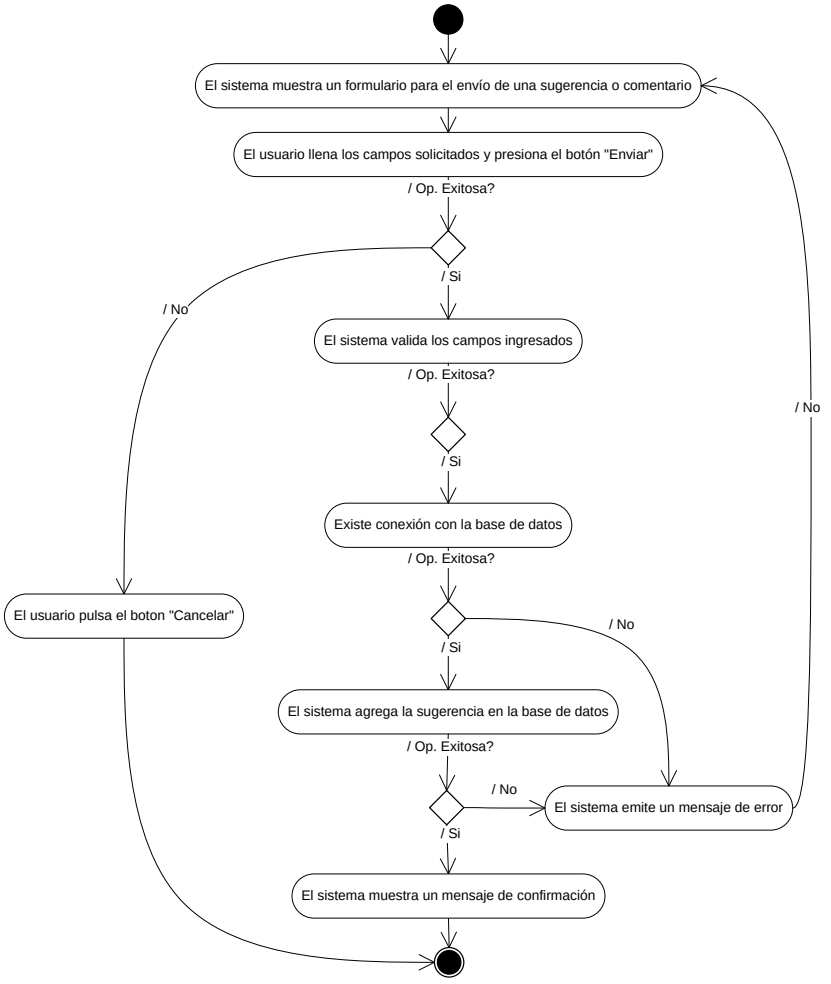

Figura 3.10 Diagrama de Actividades “Agregar Sugerencia”

Diagrama de secuencia

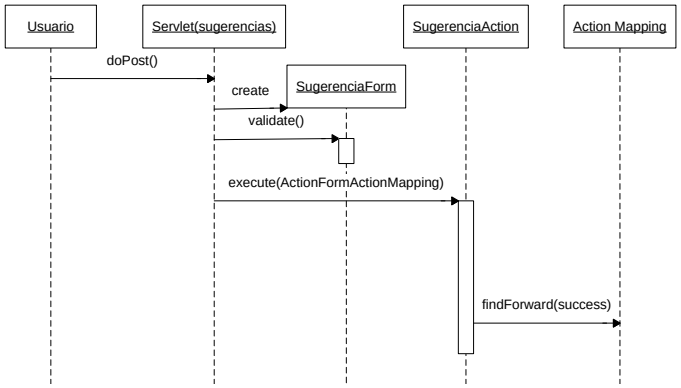

Figura 3.11 Diagrama de Secuencia "Agregar Sugerencia"

Diagrama de colaboración

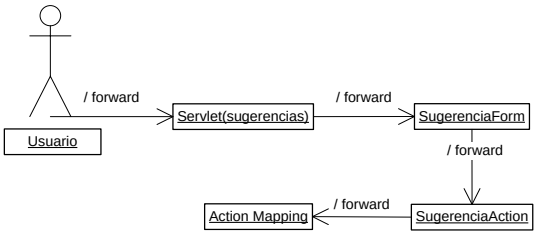

Figura 3.12 Diagrama de Colaboración "Agregar Sugerencia"

3.5 Caso de uso: Agregar Usuario

Breve descripción

Este caso de uso permite agregar un nuevo usuario administrador para el uso del sistema.

Actores

Usuario

Flujo de Eventos

Flujo Básico

1b El sistema muestra un formulario solicitando lo siguiente:

Datos del Usuario			
Nombre del campo	**Descripción**	**Tipo de Dato**	**Dominio**
Login	Se refiere a el nombre del usuario	Texto	No nulo, letras del alfabeto, números.
Password	Se refiere a la contraseña del usuario	Texto	No nulo, letras del alfabeto, números.
Nombre	Se refiere a el nombre del recurso humano	Texto	No nulo, letras del alfabeto, números.

Tabla 3.4 Agregar Usuario

Y el botón "Aceptar" y "Cancelar".

2b El usuario coloca la información solicitada y pulsa en el botón "Aceptar".

3b El sistema verifica que los campos cumplan con las especificaciones de dominio establecidas en el punto 1b y los almacena en la base de datos; si no existe conexión con la base de datos el sistema emite un mensaje de error.

4b El usuario pulsa el botón "Aceptar", el sistema pasa al caso de uso **"Listar Elementos"**.

Flujos Alternos

1a <<**1b**>> El usuario pulsa el botón "Cancelar" y el sistema pasa al caso de uso **"Listar Elementos"**.

Precondiciones

El usuario deberá ingresar al sistema por medio de un usuario y una contraseña.

Diagrama de Actividades

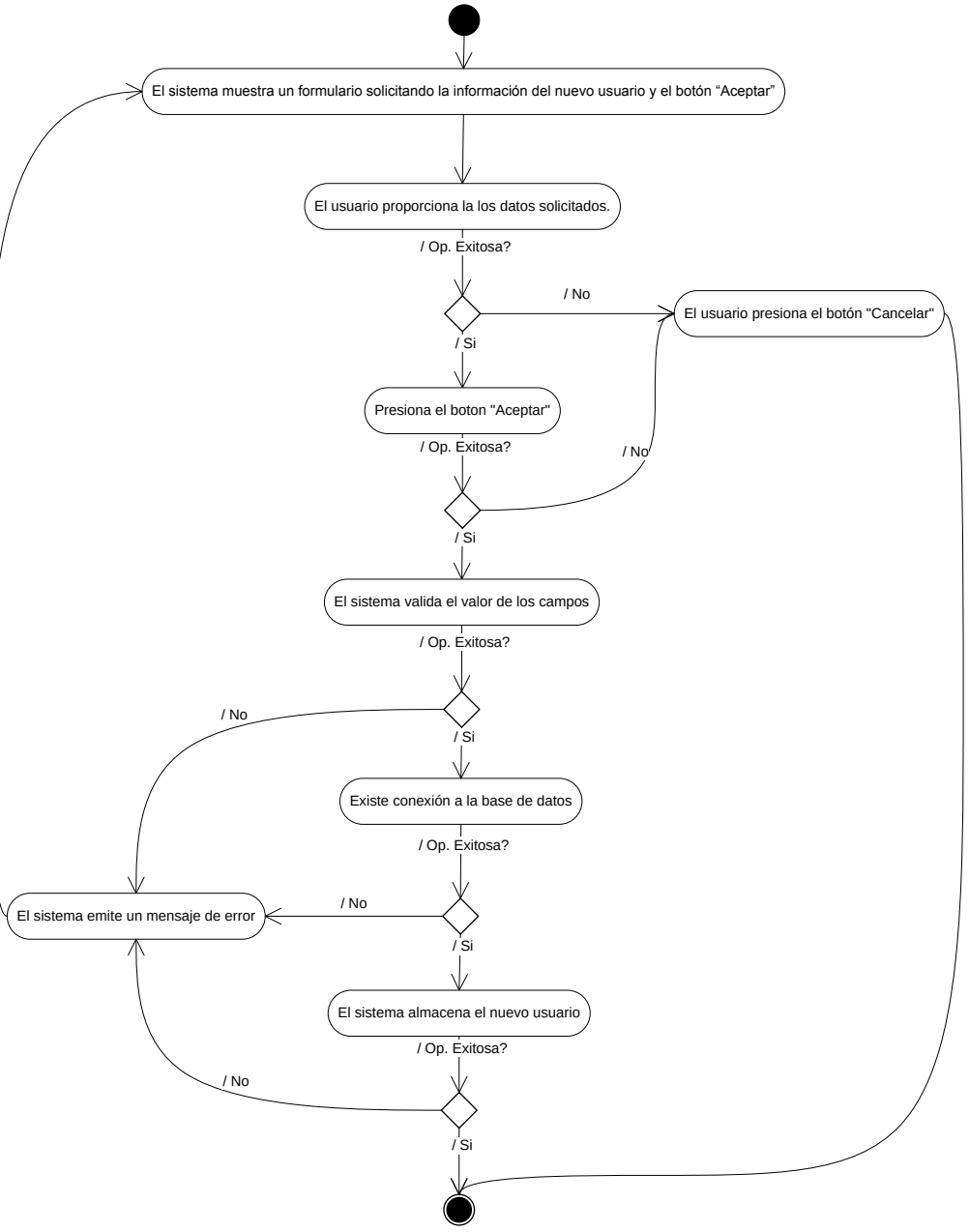

Figura 3.13 Diagrama de Actividades "Agregar Usuario"

Diagrama de Secuencia

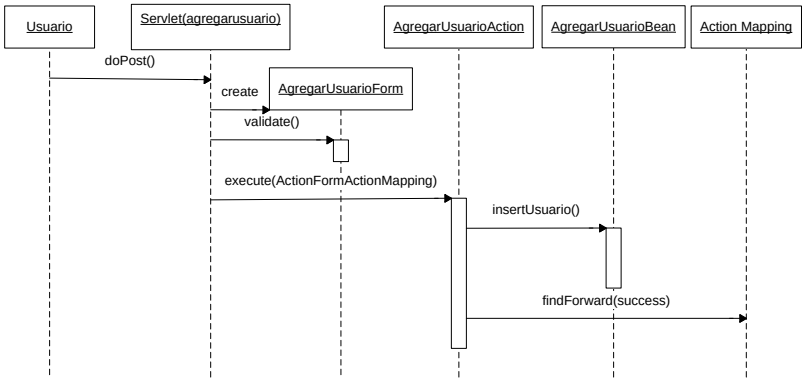

Figura 3.14 Diagrama de Secuencia "Agregar Usuario"

Diagrama de Colaboración

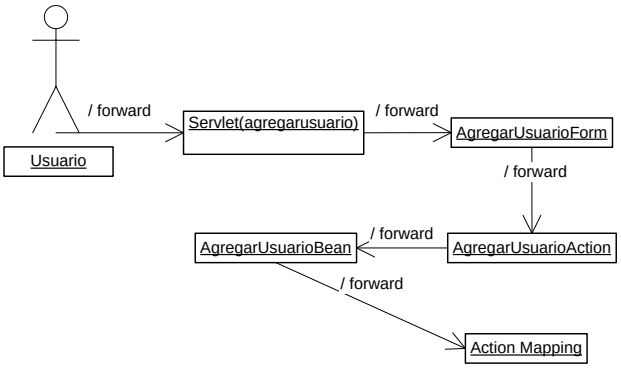

Figura 3.15 Diagrama de Colaboración "Agregar Usuario"

3.6 Caso de uso: Buscar Artículo

Breve descripción

Este caso de uso permite al usuario definir criterios de búsqueda sobre los artículos existentes en la base de datos.

Actores

Usuario

Flujo de Eventos

Flujo Básico

1b El sistema muestra un formulario solicitando la siguiente información:

Datos de la búsqueda			
Nombre del campo	**Descripción**	**Tipo de Dato**	**Dominio**
Tipo de Norma	Se refiere al tipo de norma sobre la que se quiere consultar.	Texto	No nulo, “Ley”,”Reglamento”,”Código”,”Decreto”, “Todos”.
Nivel de la Norma	Se refiere al nivel de las normas sobre las cuales se quiere hacer la consulta	Texto	No nulo, “Municipal”, “Estatal”, “Federal”, “Todos”.
Palabra	Se refiere a la(s) palabra(s) a buscar sobre los artículos	Texto	No nulo, letras del alfabeto.
Operador Lógico	Es un operador que permite establecer una relación entre dos expresiones	Texto	“Y”, “O”.

Y los botones “Agregar Criterio”, “Eliminar Criterio” y “Buscar”.

2b El usuario selecciona y llena los campos requeridos en la forma, establecidos en el punto 1b y pulsa “Agregar Criterio”.

3b El sistema verifica que los campos cumplan con las especificaciones de dominio establecidas en el punto 1b y construye una cadena de consulta.

4b El usuario pulsa en el botón “Buscar”.

5b El sistema consulta en la base de datos los artículos que corresponden a los criterios enviados y pasa al caso de uso **“Mostrar Resultados”**.

Flujo Alterno

1a **<<4b>>** El usuario **Tabla 3.5 Buscar Artículo** pulso en el botón “Eliminar Criterio”, el sistema verifica que existan criterios a eliminar y se realiza la operación; en caso contrario, se informa al usuario la razón por la que no se completo la solicitud. El sistema pasa al caso de uso **“Buscar Artículo”.**

Requerimientos Especiales

Ninguno.

Precondiciones

El usuario deberá ingresar al sistema por medio de un login y un password.

Diagrama de Secuencia

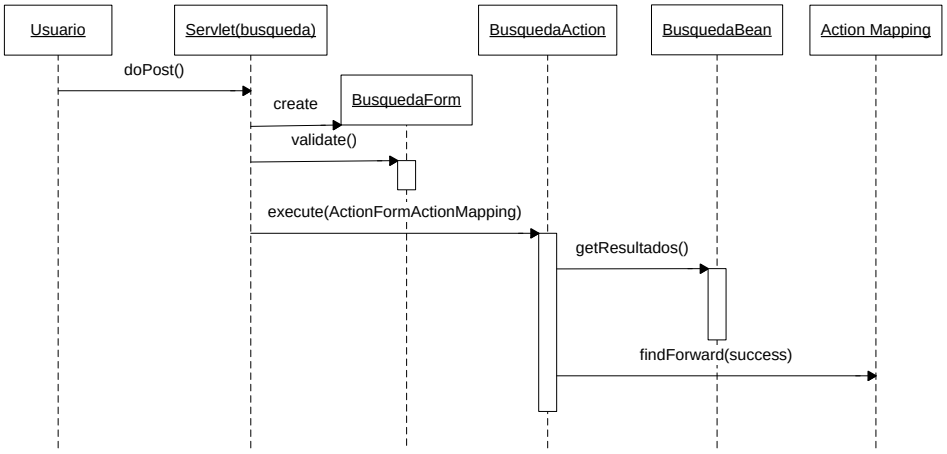

Figura 3.16 Diagrama de Secuencia “Buscar Artículo”

Diagrama de Actividades

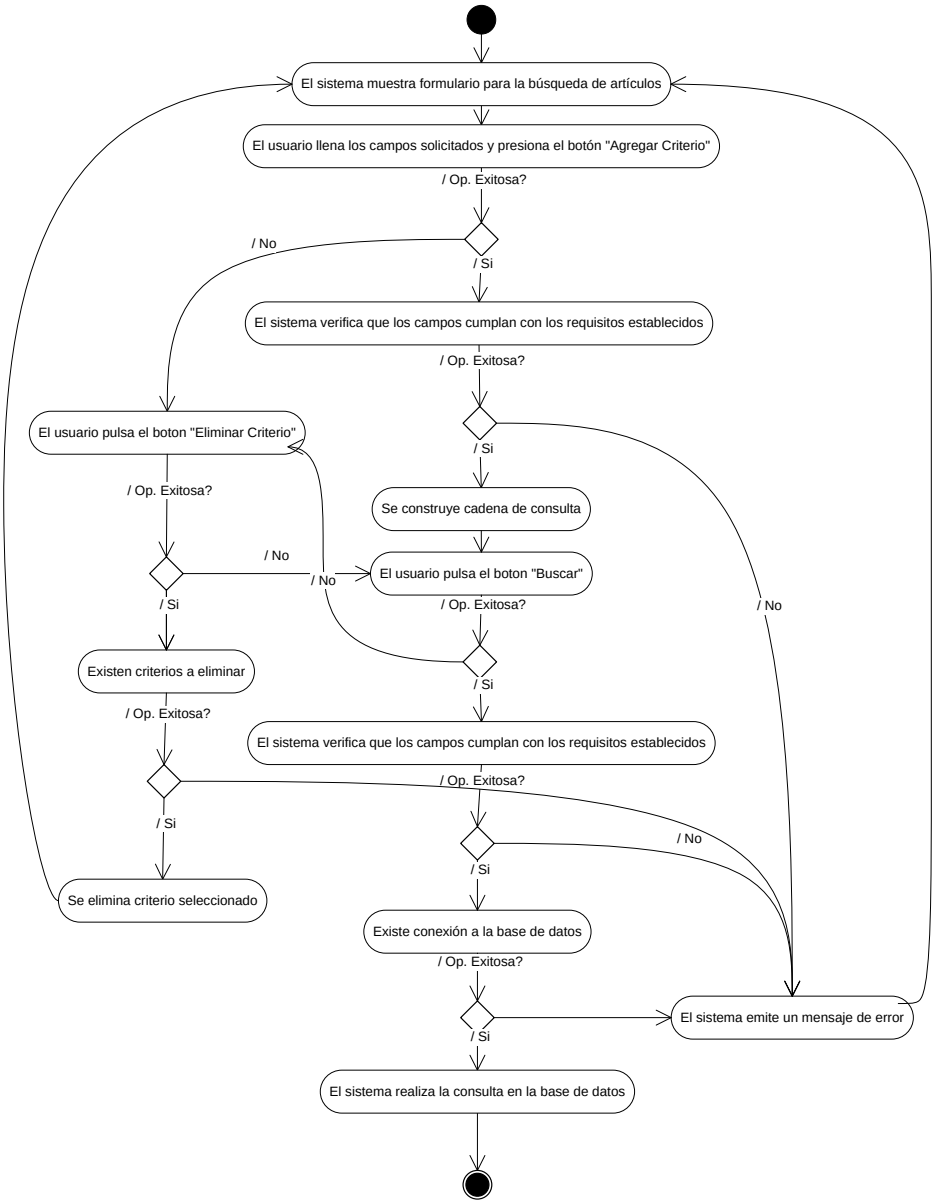

Figura 3.17 Diagrama de Actividades "Buscar Artículo"

Diagrama de Colaboración

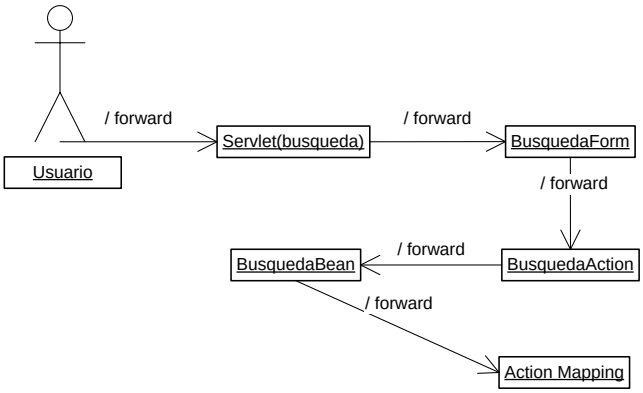

Figura 3.18 Diagrama de Colaboración "Buscar Artículo"

3.7 Caso de uso: Consultar Artículo

Breve descripción

Este caso de uso permite al usuario consultar un artículo de una lista de artículos, resultado criterios de una búsqueda previa.

Actores

Usuario

Flujo de Eventos

Flujo Básico

1b El sistema muestra un listado de artículos que corresponden a el(los) criterio(s) de búsqueda previamente definido(s).
2b El usuario selecciona y pulsa el vínculo del artículo que desea consultar.
3b El sistema muestra el detalle del artículo seleccionado, así como los artículos y artículo transitorios resultado de la búsqueda.
4b El usuario pulsa un vínculo de la lista de artículos/transitorios, el sistema pasa al caso de uso **"Consultar Artículo"**.

Flujo Alterno

1a **<<2b>>** El usuario pulso en el botón "Administración", el sistema pasa al caso de uso **"Login".**
2a **<<2b>>** El usuario pulso en el botón "Búsqueda", el sistema pasa al caso de uso **"Buscar Artículo".**
3a **<<2b>>** El usuario pulso en el botón "Sugerencias", el sistema pasa al caso de uso **"Agregar Sugerencia".**

Requerimientos Especiales

Ninguno.

Precondiciones

Ninguna.

Diagrama de Actividades

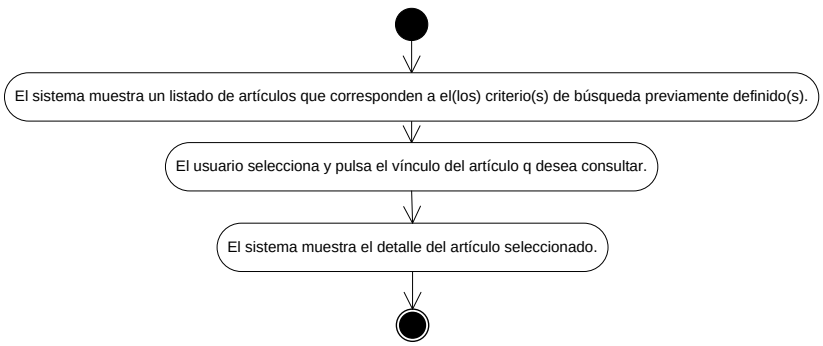

Figura 3.19 Diagrama de Actividades "Consultar Artículo"

3.8 Caso de uso: Eliminar Elemento

Breve descripción

Este caso de uso permite eliminar un elemento de una lista.

Actores

Usuario

Flujo de Eventos

Flujo Básico

1b El sistema muestra un listado de los elementos con una opción que permite seleccionarlos y un botón "Eliminar".
2b El usuario selecciona el elemento que desea eliminar y pulsa el botón "Eliminar".
3b El sistema elimina el elemento y la información asociada a éste, el sistema pasa al caso de uso **"Listar Elementos"**.

Flujo Alterno

1a <<**1b**>> El usuario realiza la selección de otra operación y pasa al caso de uso correspondiente.
2a <<**3b>>** Si existe algún error en la eliminación de los datos en la base de datos, el sistema muestra una página de error con un botón "Regresar".
2a.1 El usuario da clic en el botón "Regresar".
2a.2 El sistema regresa al punto desde donde fue llamado este caso de uso.

Requerimientos Especiales

Ninguno.

Precondiciones

El usuario tuvo que haber iniciado sesión en el sistema.

Diagrama de Colaboración

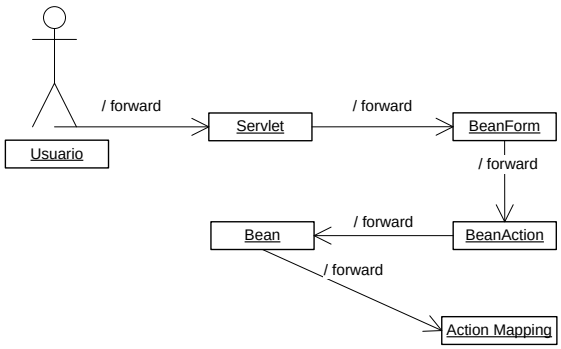

Figura 3.20 Diagrama de Colaboración "Eliminar Elemento"

Diagrama de Actividades

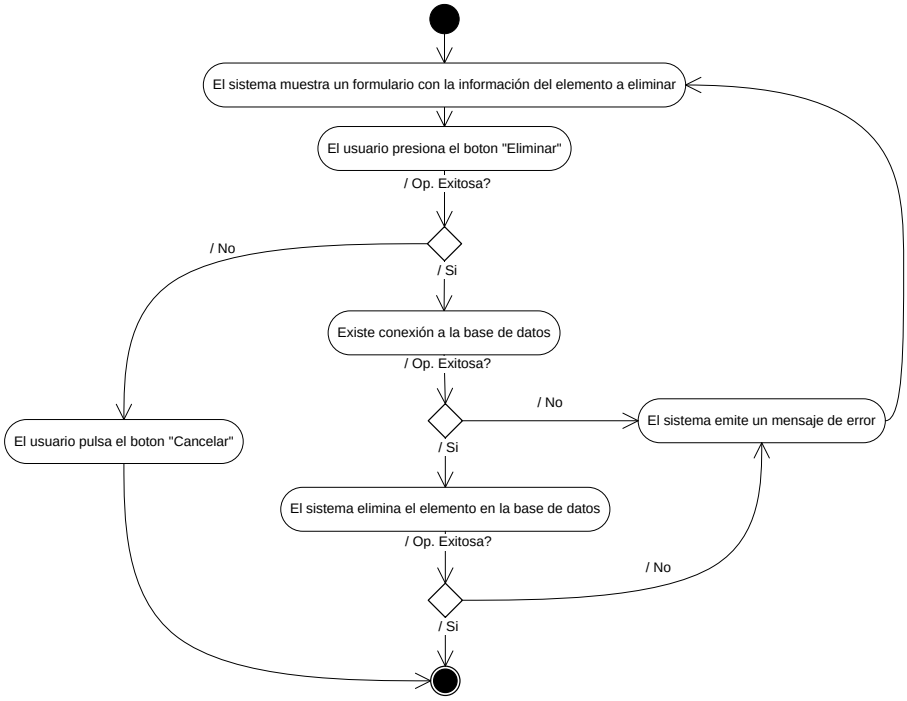

Figura 3.21 Diagrama de Actividades "Eliminar Elemento"

Diagrama de Secuencia

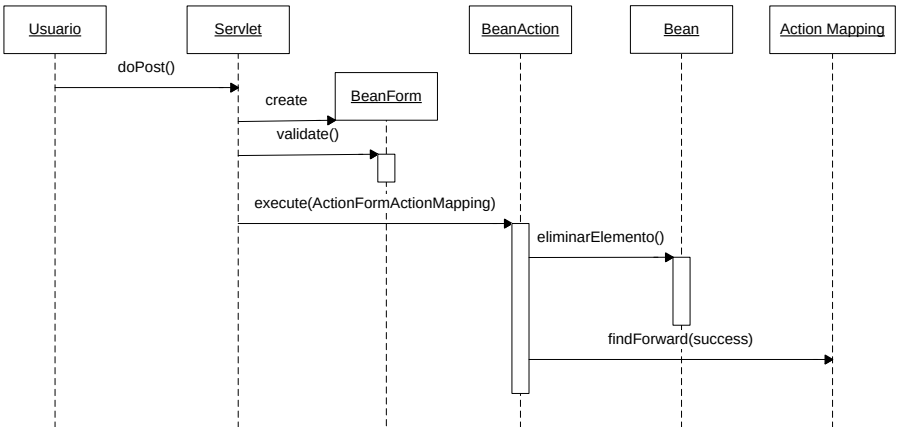

Figura 3.22 Diagrama de Secuencia "Eliminar Elemento"

3.9 Caso de uso: Listar Elementos

Breve descripción

Este caso de uso muestra todos los elementos que se encuentran asociados a la lista seleccionada.

Actores

Usuario

Flujo de Eventos

Flujo Básico

1b El sistema realiza una consulta a la base de datos para obtener los elementos asociados a la lista seleccionada y sus atributos; si no existe conexión con la base de datos el sistema emite un mensaje de error.

2b El sistema muestra un listado de los archivos con una opción que permite seleccionarlos.

Flujos Alternos

Ninguno.

Requerimientos especiales

El usuario tuvo que haber iniciado sesión en el sistema.

Precondiciones

Ninguna.

Diagrama de Actividades

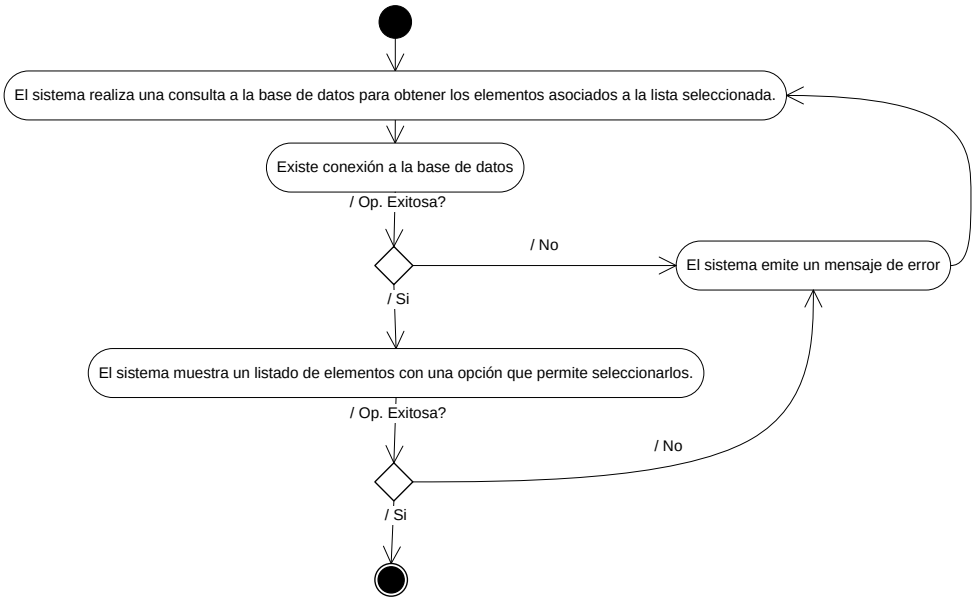

Figura 3.23 Diagrama de Actividades "Listar Elementos"

Diagrama de Colaboración

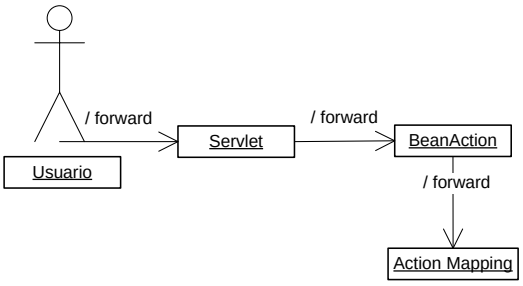

Figura 3.24 Diagrama de Colaboración "Listar Elementos"

Diagrama de Secuencia

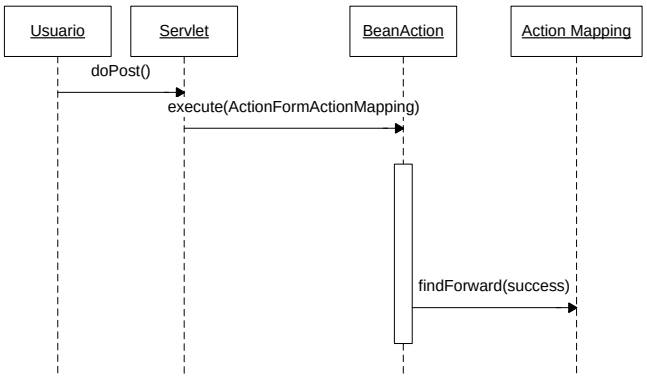

Figura 3.25 Diagrama de Secuencia "Listar Elementos"

3.10 Caso de uso: Login

Breve descripción

Este caso de uso permite a los usuarios firmarse en el sistema.

Actores

Usuario

Flujo de Eventos

Flujo Básico

1b El sistema muestra un formulario solicitando los siguientes datos:

Datos del Login			
Nombre del campo	**Descripción**	**Tipo de Dato**	**Dominio**
Login	Es el nombre del usuario	Texto	No nulo, letras del alfabeto, números.
Password	Es la contraseña del usuario	Texto	No nulo, letras del alfabeto, números.

Tabla 3.6 Login

Y los botones "Aceptar" y "Cancelar".

2b El usuario llena los campos requeridos de acuerdo a lo indicado en la tabla en 1b, y pulsa el botón "Aceptar".

3b El sistema verifica que los campos cumplan con las especificaciones de dominio establecidas en el punto 1b, que correspondan al usuario y retorna una sesión con la información del usuario en el sistema.

4b El sistema pasa al caso de uso **"Listar Elementos".**

Flujo Alterno

1a <<**2b**>> El usuario pulsa el botón "Cancelar", el sistema limpia el formulario y pasa al caso de uso **"Buscar Artículo"**.

2a <<**2b**>> Si los datos no corresponden a lo definido en la base de datos, el sistema muestra un mensaje de error.

4a.1 El usuario da clic en el botón "Regresar".

4a.2 El sistema regresa al caso de uso **"Login"**.

Requerimientos Especiales

El usuario deberá de permitir al sistema mostrar ventanas emergentes sobre su navegador.

Precondiciones

El usuario deberá de tener predefinida una cuenta en el sistema.

Poscondiciones

Para asegurar la confidencialidad de la información publicada, el usuario deberá de concluir su sesión de trabajo.

Diagrama de Actividades

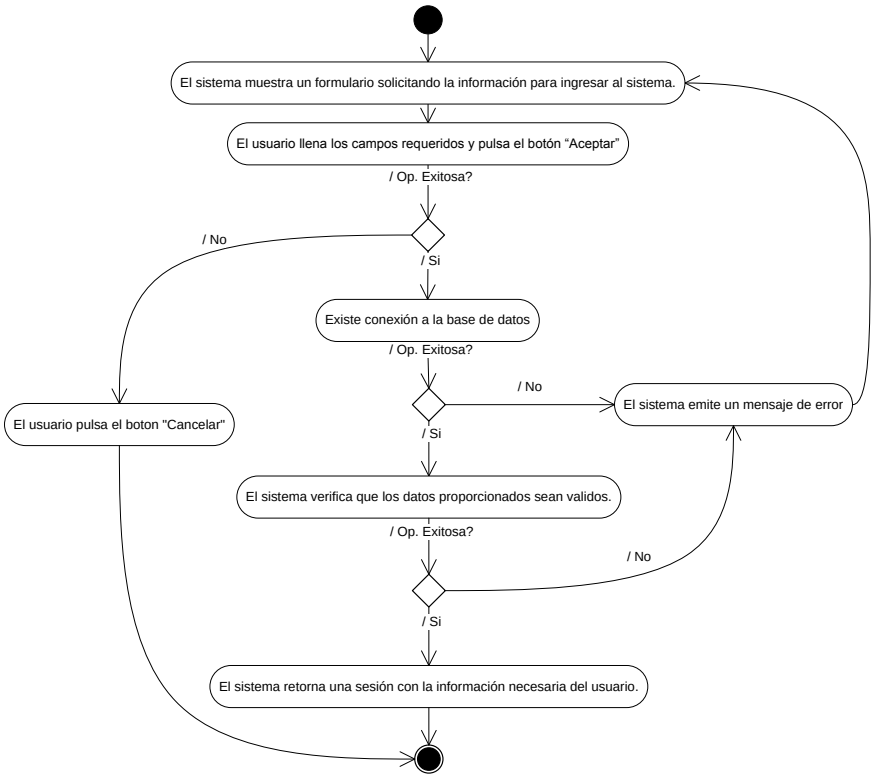

Figura 3.26 Diagrama de Actividades “Login”

Diagrama de Secuencia

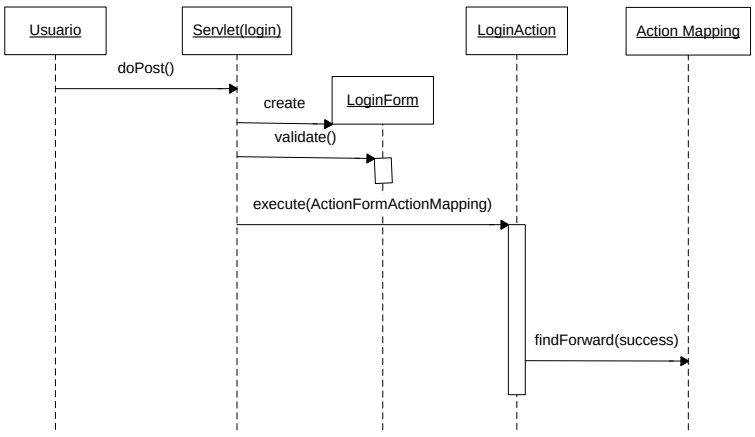

Figura 3.27 Diagrama de Secuencia “Login”

Diagrama de Colaboración

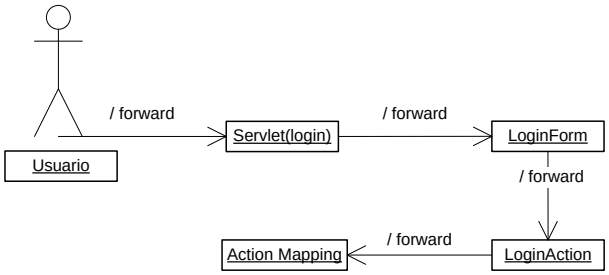

Figura 3.28 Diagrama de Colaboración "Login"

3.11 Caso de uso: Modificar Elemento

Breve descripción

Este caso de uso permite modificar un elemento (libro, título, capítulo, artículo, transitorio).

Actores

Usuario

Flujo de Eventos

Flujo Básico

1b El sistema muestra un formulario con los campos especificados en la tabla 3.1 anteriormente almacenados:
Y dos botones "Cancelar" y "Aceptar".
2b El usuario modifica el texto del elemento solicitado y pulsa en el botón "Aceptar".
3b El sistema verifica que los campos cumplan con las especificaciones de dominio establecidas en el punto 1b y se almacenan los cambios en la base de datos.
4b El sistema pasa al caso de uso **"Modificar Norma"/"Agregar Norma"**.

Flujo Alterno

1a <<**1b**>> El usuario pulsa el botón "Cancelar", el sistema pasa al caso de uso **"Agregar Norma"/"Modificar Norma"**.
2a <<**3b**>> Si existe algún error en la modificación de los datos en la base de datos, el sistema muestra una página de error con un botón "Regresar".
2a.1 El usuario da clic en el botón "Regresar".
2a.2 El sistema regresa al caso de uso **"Agregar Norma"/"Modificar Norma"**.

Requerimientos Especiales

Ninguno.

Precondiciones

El usuario tuvo que haber iniciado sesión en el sistema.

Diagrama de Actividades

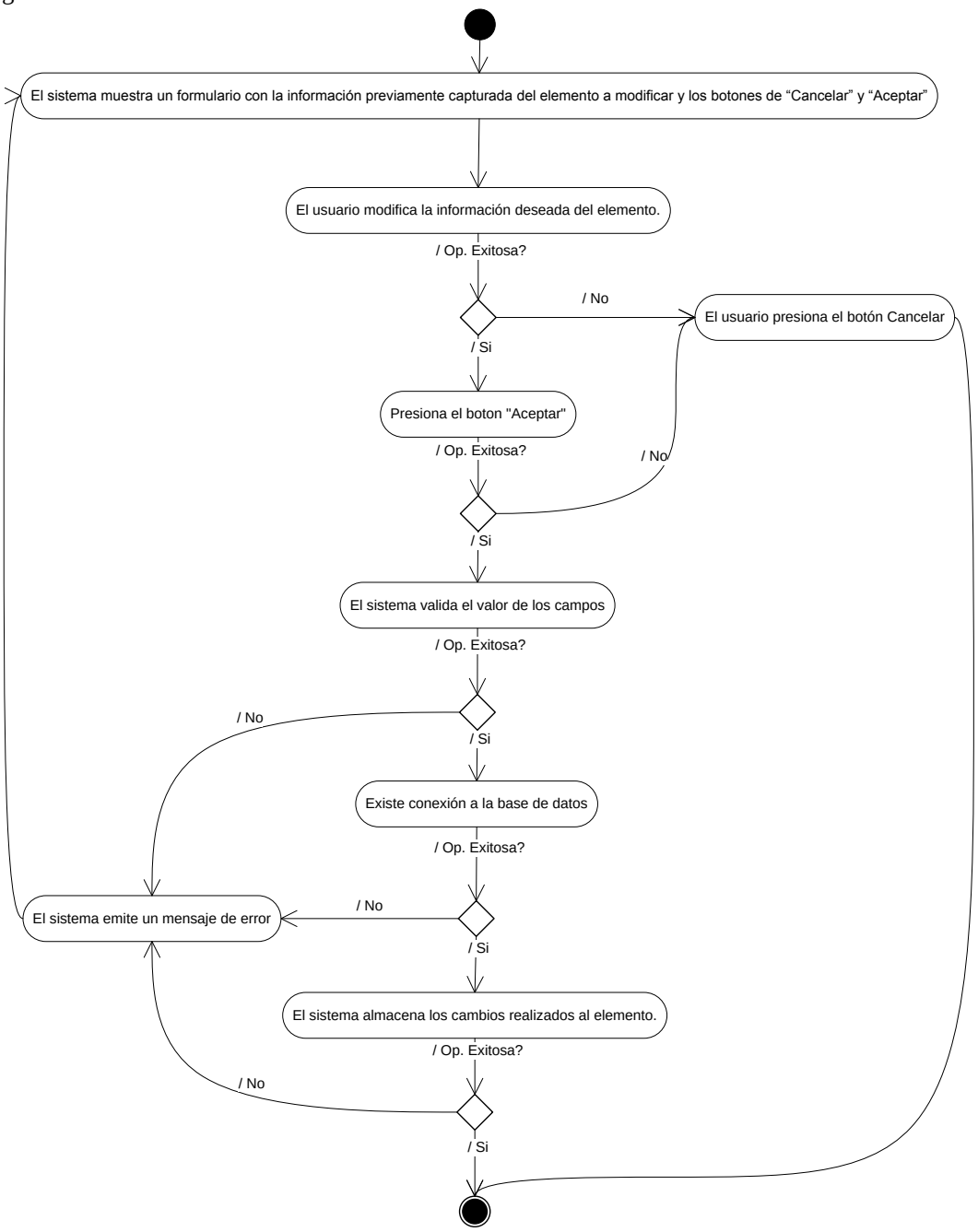

Figura 3.29 Diagrama de Actividades “Modificar Elemento”

Diagrama de Colaboración

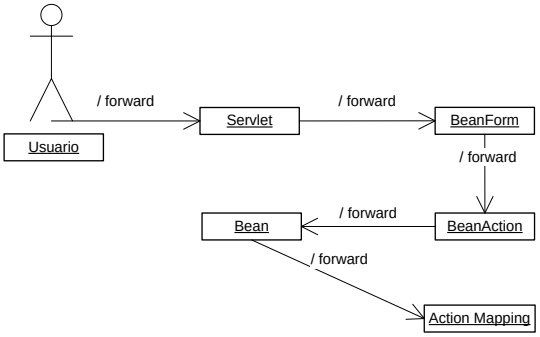

Figura 3.30 Diagrama de Colaboración “Modificar Elemento”

Diagrama de Secuencia

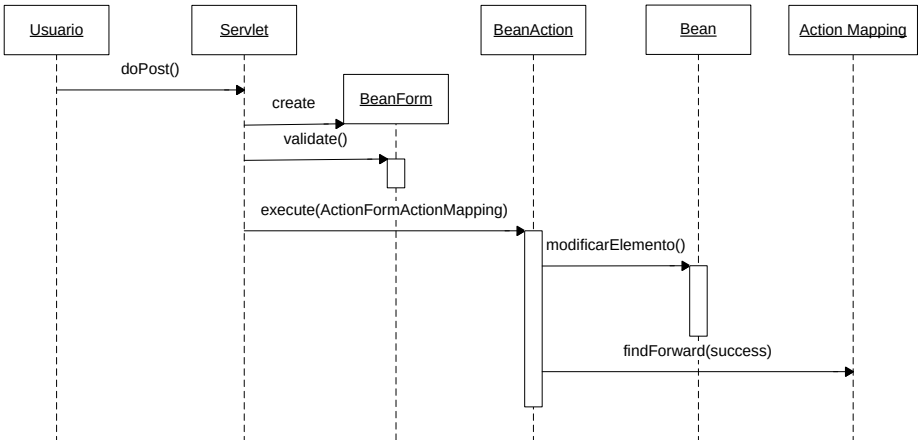

Figura 3.31 Diagrama de Secuencia "Modificar Elemento"

3.12 Caso de uso: Modificar Norma

Breve descripción

Este caso de uso permite modificar una norma previamente definida.

Actores

Usuario

Flujo de Eventos

Flujo Básico

1b El sistema muestra un formulario con los campos que se especifican en la tabla 3.2 anteriormente almacenados.
Y los botones "Examinar", "Subir Archivo", "Agregar/Eliminar Archivo", "Eliminar Archivos Seleccionados", "Realizar Operación", "Siguiente", "Anterior", "Cancelar" y "Terminar".

2b El usuario modifica el texto de los campos deseados y pulsa en el botón "Siguiente".

3b El sistema verifica que los campos de la primera parte de la modificación cumplan con las especificaciones de dominio mostradas en la tabla del punto 1b.

4b El sistema muestra un listado de operaciones que se pueden realizar con libros, títulos, capítulos, artículos, transitorios y archivos para terminar de modificar la norma jurídica seleccionada.

5b El usuario selecciona la operación deseada y pasa al caso de uso correspondiente.

6b El usuario modifica los campos solicitados y termina pulsando en el botón "Terminar".

7b El sistema modifica los datos de la norma en la base de datos; si no existe conexión con la base de datos el sistema emite un mensaje de error.

8b El sistema pasa al caso de uso **"Listar Normas".**

Flujo Alterno

1a <<**1b**>> El usuario pulsa en "Cancelar", el sistema pasa al caso de uso **"Listar Normas"**.

2a <<**3b**>> Los campos ingresados en la tabla del punto 1b no cumple con las especificaciones de dominio establecidos, el sistema muestra un mensaje informando al usuario los datos incorrectos o faltantes y regresa al punto 1b.

3a <<**4b**>> El usuario selecciona la operación "Agregar" junto con el elemento "Artículo", "Capítulo","Título","Libro" ó "Transitorio" y presiona el botón "Realizar Operación", el sistema pasa al caso de uso "**Agregar Elemento**".

4a <<**4b**>> El usuario selecciona la operación "Modificar" junto con el elemento "Artículo", "Capítulo","Título","Libro" ó "Transitorio" y presiona el botón "Realizar Operación", el sistema pasa al caso de uso "**Modificar Elemento**".

5a <<**4b**>> El usuario selecciona la operación "Eliminar" junto con el elemento "Artículo", "Capítulo","Título","Libro" ó "Transitorio" y presiona el botón "Realizar Operación", el sistema pasa al caso de uso "**Eliminar Elemento**".

6a <<**4b**>> El usuario selecciona la operación "Agregar/Eliminar Archivos" y presiona el botón "Realizar Operación", el sistema pasa al caso de uso "**Listar Elementos**".

Requerimientos Especiales

El usuario deberá de reconocer de forma previa a la acción de "Modificar Norma", la estructura de la norma a anexar en el sistema.

Precondiciones

El usuario tuvo que haber iniciado sesión en el sistema.

Diagrama de Secuencia

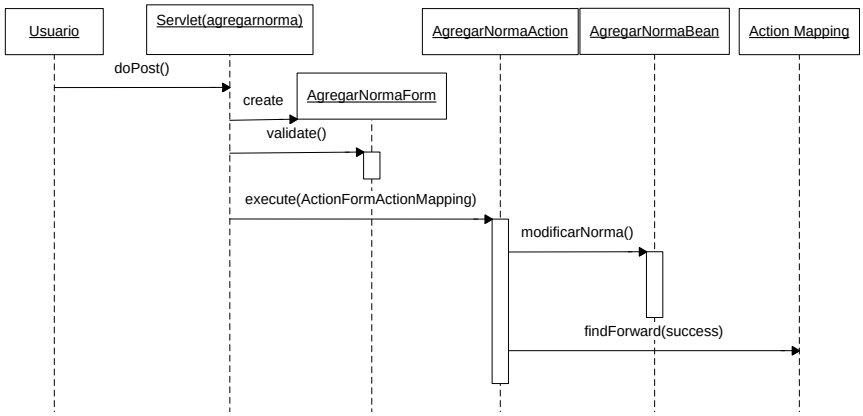

Figura 3.32 Diagrama de Secuencia "Modificar Norma"

Diagrama de Colaboración

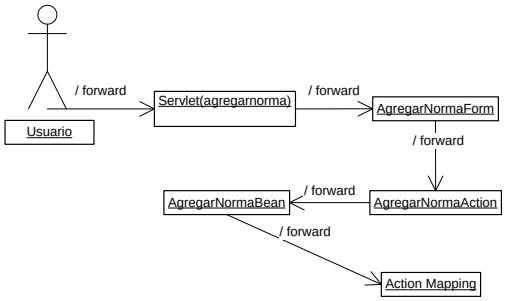

Figura 3.33 Diagrama de Colaboración "Modificar Norma"

Diagrama de Actividades

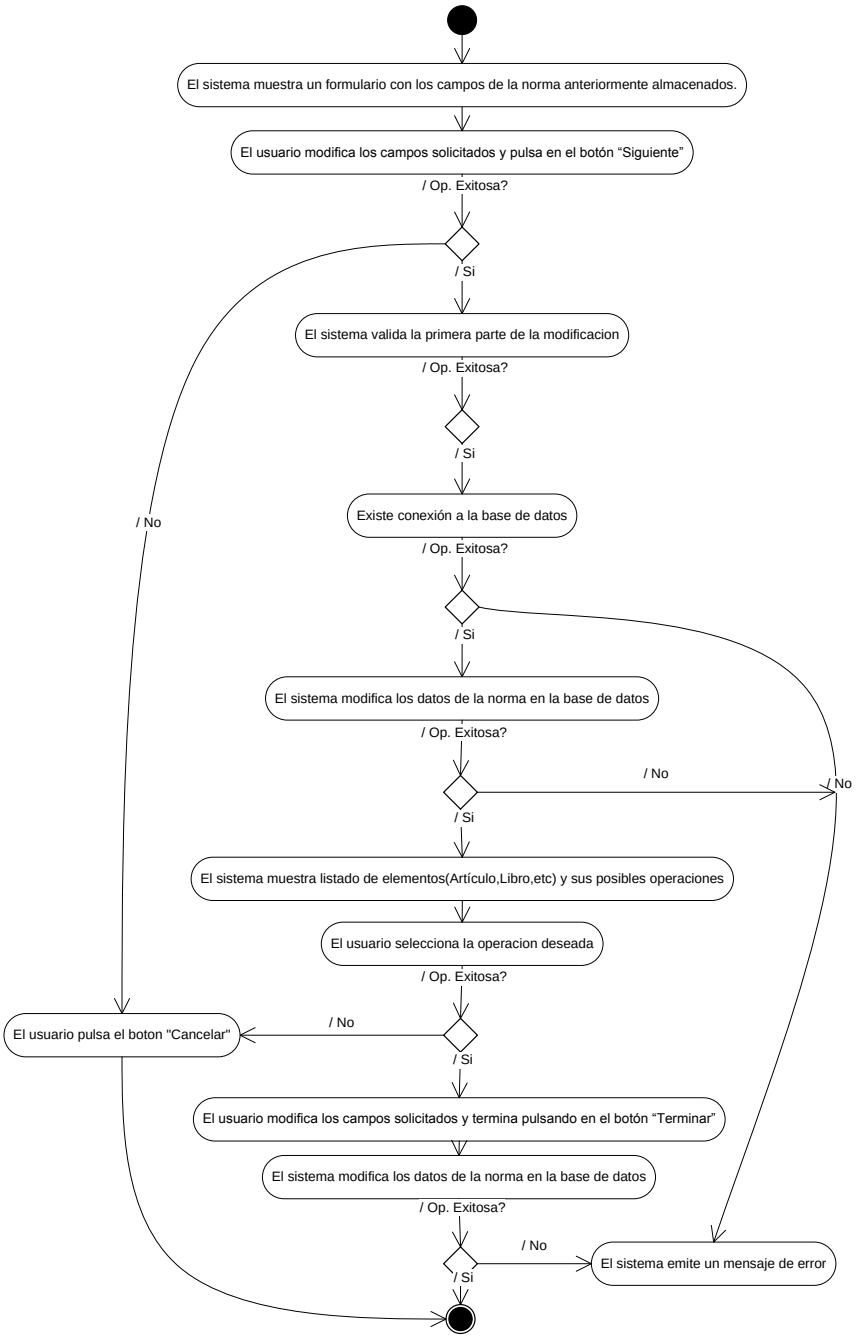

Figura 3.34 Diagrama de Actividades “Modificar Norma”

3.13 Caso de uso: Modificar Usuario

Breve descripción

Este caso de uso permite modificar un usuario.

Actores

Usuario

Flujo de Eventos

Flujo Básico

1b El sistema muestra un formulario con campos anteriormente almacenados, véase tabla 3.4.

Y el botón "Aceptar" y "Cancelar".

2b El usuario modifica el texto del usuario solicitado y pulsa en el botón "Aceptar".

3b El sistema verifica que los campos cumplan con las especificaciones de dominio establecidas en el punto 1b y se almacenan los cambios en la base de datos.

4b El sistema pasa al caso de uso **"Listar Elementos"**.

Flujo Alterno

1a **<<1b>>** El usuario pulsa el botón "Cancelar", el sistema pasa al caso de uso **"Listar Elementos"**.

2a <<**3b**>> Si existe algún error en la modificación de los datos en la base de datos, el sistema muestra una página de error con un botón "Regresar".

2a.1 El usuario da clic en el botón "Regresar".

2a.2 El sistema regresa al caso de uso **"Listar Elementos"**.

Requerimientos Especiales

Ninguno.

Precondiciones

El usuario tuvo que haber iniciado sesión en el sistema.

Diagrama de Secuencia

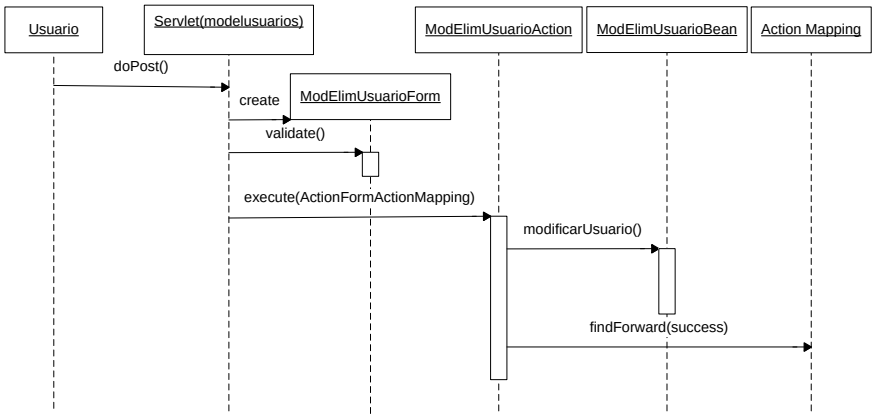

Figura 3.35 Diagrama de Secuencia "Modificar Usuario"

Diagrama de Colaboración

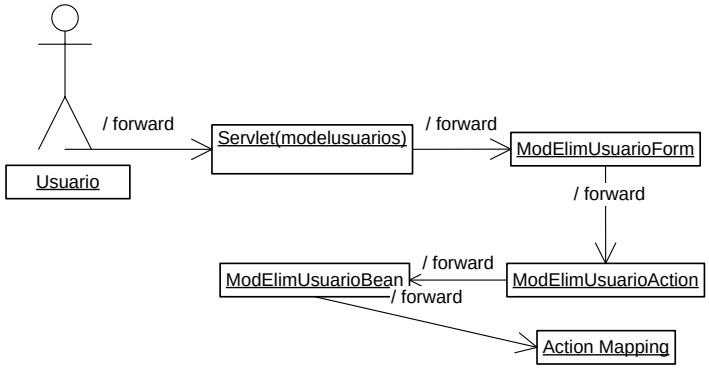

Figura 3.36 Diagrama de Colaboración "Modificar Usuario"

Diagrama de Actividades

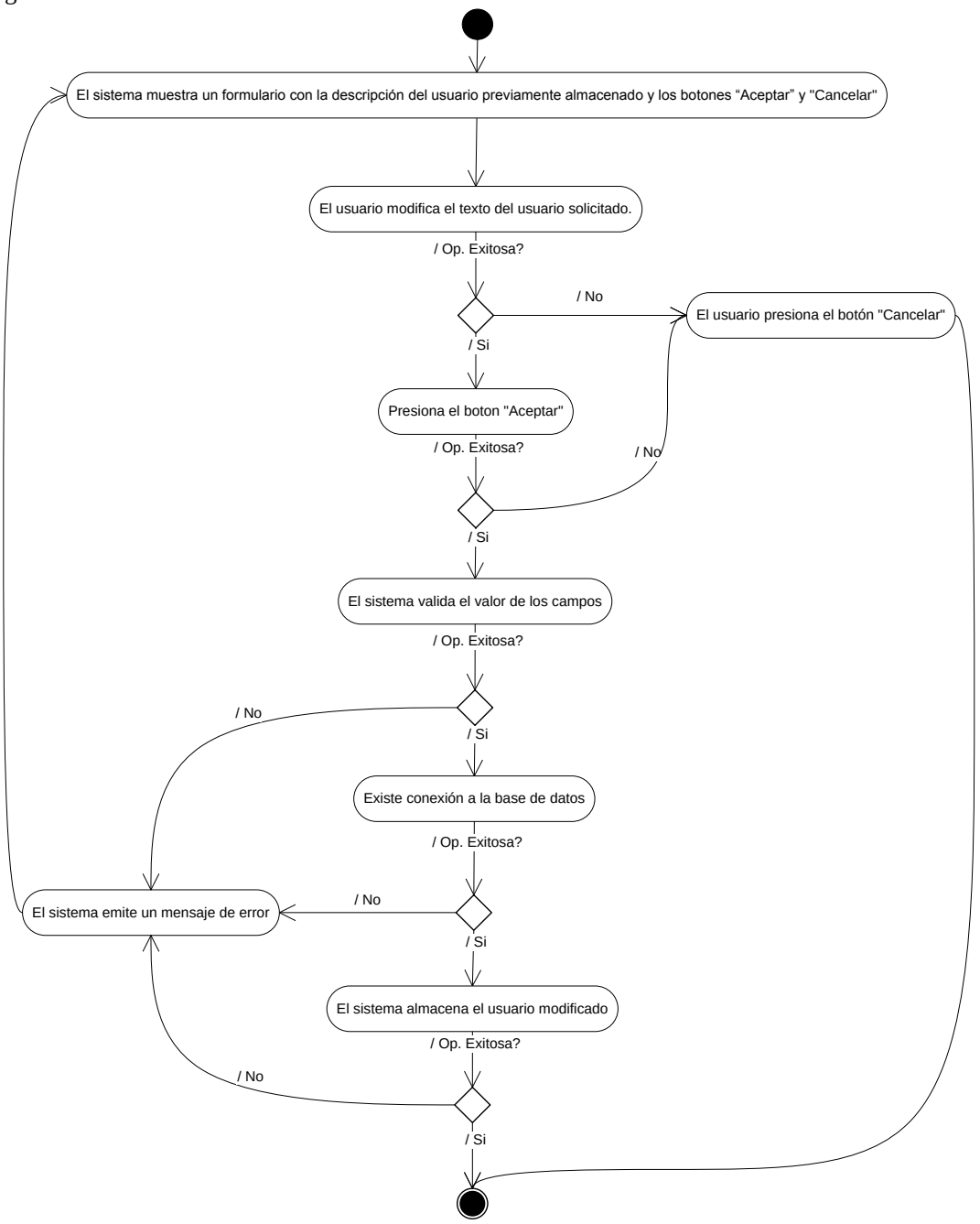

Figura 3.37 Diagrama de Actividades “Modificar Usuario”

3.14 Caso de uso: Mostrar Resultados

Breve descripción

Este caso de uso muestra todos los artículos que coinciden con los criterios de búsqueda previamente definidos.

Actores

Usuario

Flujo de Eventos

Flujo Básico

1b El sistema realiza una consulta a la base de datos para obtener los artículos que coinciden los criterios de búsqueda deseados y sus atributos; si no existe conexión con la base de datos el sistema emite un mensaje de error.

2b El sistema muestra un listado de artículos.

Flujos Alternos

Ninguno.

Requerimientos especiales

El usuario debió de haber realizado una búsqueda con el caso de uso **"Buscar Artículo".**

Precondiciones

Ninguna.

Diagrama de Actividades

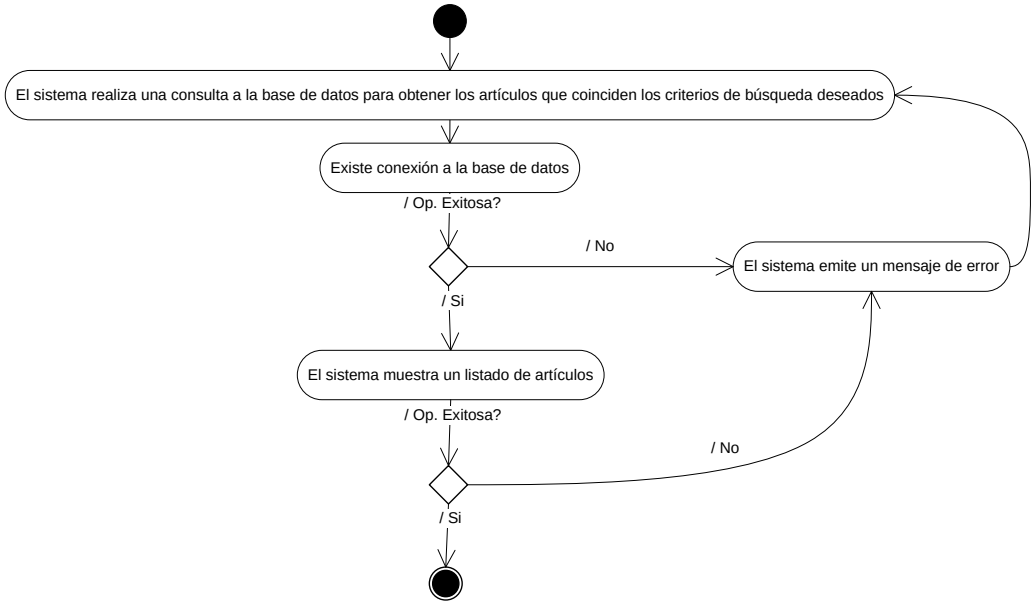

Figura 3.38 Diagrama de Actividades "Mostrar Resultados"

Diagrama de Secuencia

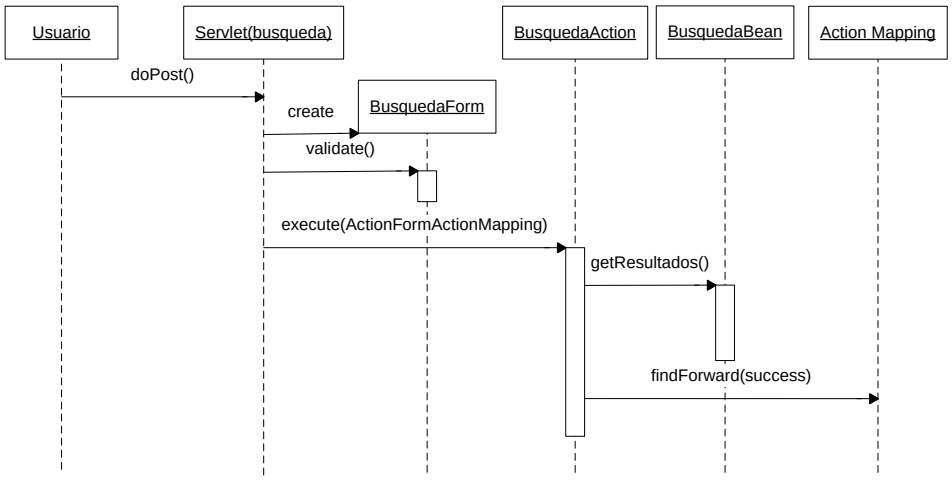

Figura 3.39 Diagrama de Secuencia "Mostrar Resultados"

Diagrama de Colaboración

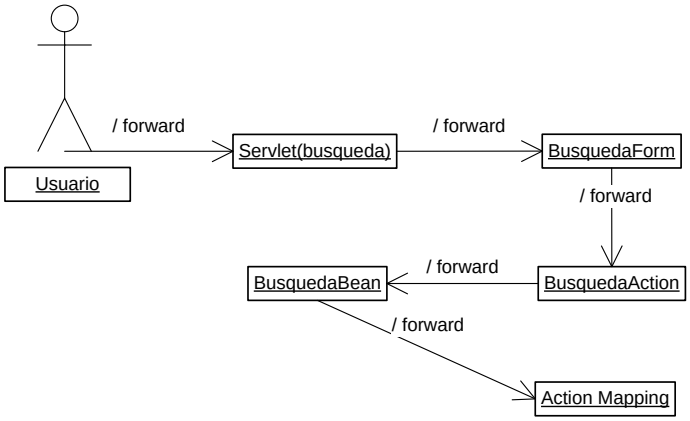

Figura 3.40 Diagrama de Colaboración "Mostrar Resultados"

3.15 Diagrama de paquetes

El siguiente diagrama (Figura 3.41) muestra la distribución funcional de los paquetes en el sistema.

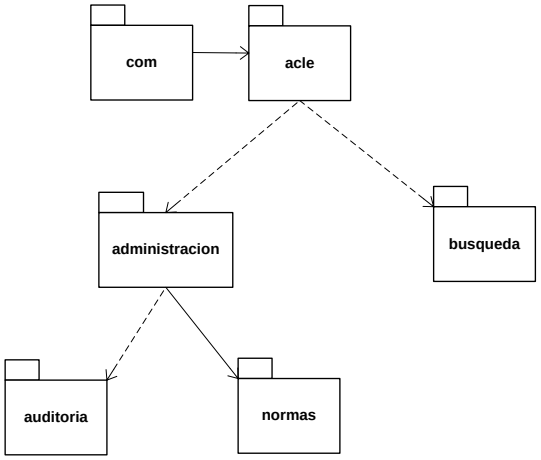

Figura 3.41 Diagrama de Paquetes.

3.16 Diagrama de clases

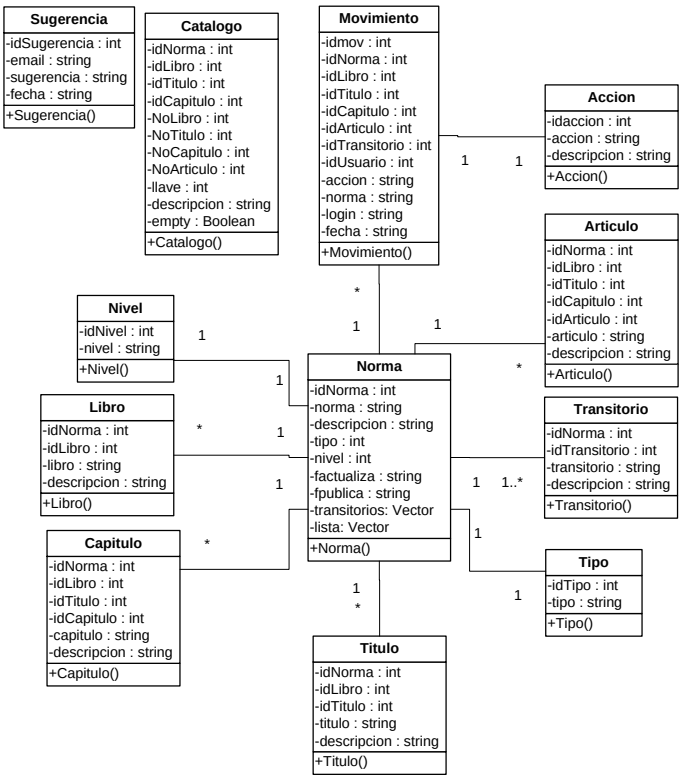

Figura 3.42 Diagrama de Clases.

3.17 Modelo relacional

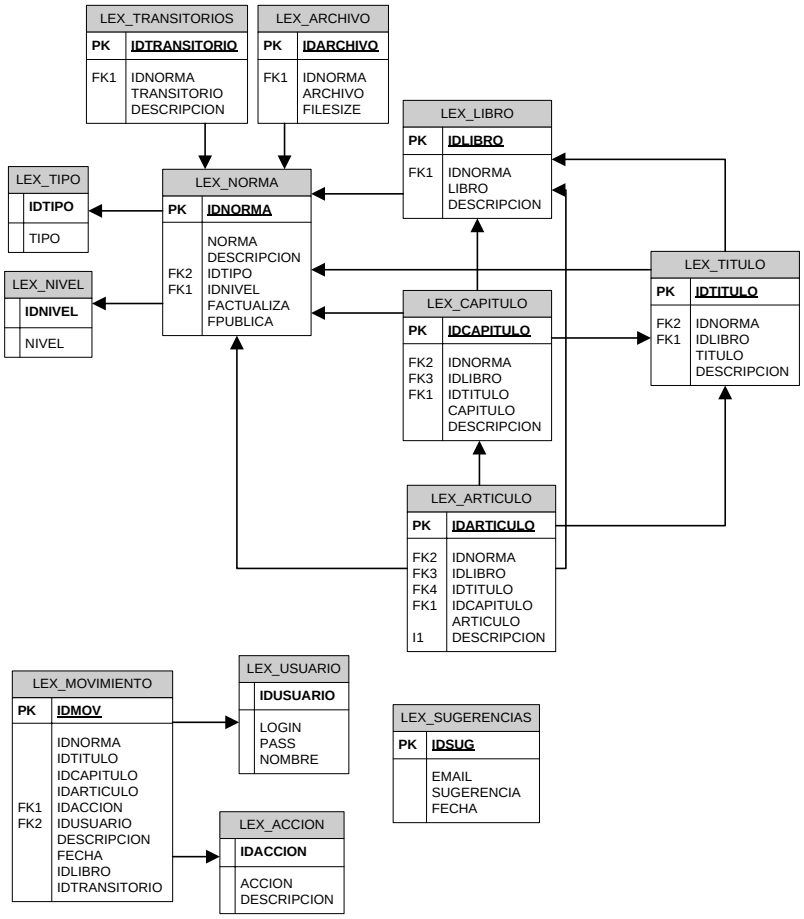

Figura 3.43 Modelo Relacional

4. Implementación

Al desarrollar un sistema, se tiene que analizar el problema y los objetivos a seguir, para así poder elegir las herramientas adecuadas, con el fin de cumplir las necesidades del usuario. En este capítulo se presenta la implementación de ACLE que se planteó en el capítulo 2 y 3, se justifican los diferentes componentes usados en el desarrollo de la tesis, además de explicar a detalle cada una de las partes que conforman la herramienta.

4.1. Justificación de Software Utilizado

Ya que se decidió que ACLE debería ser una herramienta que pudiera ser accesible desde cualquier parte del mundo y por cualquier computadora con cualquier sistema, se decidió usar Java como lenguaje de programación debido a que es independiente de plataforma y facilita la creación de herramientas que corran vía Web. La interfaz de ACLE se implementó con páginas JSP, las cuales son especiales para programar scripts en sintaxis Java que son ejecutables en el servidor.

4.2. Conexión a la base de datos MySQL

ACLE interactúa con la base de datos de MySQL bajo el protocolo JDBC.

Para configurar la fuente de datos (DataSources) de la aplicación, el DataSource manager de Struts es el componente encargado de la gestión de los DataSources; se configura directamente en el fichero de configuración struts-config.xml. Podemos usarlo para proporcionar cualquier conjunto de conexiones que implemente la interfaz javax.sql.DataSource.

Si no tenemos ningún componente nativo, podemos usar el Jakarta Commons dbcp's[21] BasicDataSource [org.apache.commons.dbcp.BasicDataSource.

En nuestro caso utilizamos el siguiente código para realizar la conexión a la base de datos en MySQL:

```
<data-sources>
        <data-source type="org.apache.commons.dbcp.BasicDataSource">
                <set-property property="autoCommit" value="true"/>
                <set-property property="descripcion" value="Base de Datos ACLE"/>
                <set-property property="driverClassName" value="com.MySQL.jdbc.Driver"/>
                <set-property property="maxCount" value="30"/>
                <set-property property="minCount" value="1"/>
                <set-property property="username" value="root"/>
                <set-property property="password" value="gaby2007"/>
                <set-property property="SetBigStringTryClob" value="true"/> <!-- Agregado 07/02/2006 -->
                <set-property property="url" value="jdbc:MySQL://localhost:3306/acle"/>
        </data-source>
</data-sources>
```

Para realizar la conexión a MySQL en las clases Java, se crea un objeto de tipo “Connection” dentro de un bloque “try-catch” y a través de un DataSource que es un objeto JNDI[22], obtendremos una conexión desde un pool de conexiones hacia la base de datos.

Entre las llaves de “try” escribiremos el código que hará la solicitud de conexión a la base de datos. Para capturar la excepción que puede generar este código necesitaremos otra instrucción llamada “catch” (capturar).

[21] Commons es un subproyecto de Jakarta focalizado en todos los aspectos de componentes Java reutilizables. Entre estos componentes encontrams Commons DBCP, que nos ofrece el servicio de un pool de conexiones a la base d datos.

[22] La Interfaz de Nombrado y Directorio Java (JNDI) es una Interfaz de Programación de Aplicaciones para servicios de directorio. Esto permite a los clientes descubrir y buscar objetos y nombres a través de un nombre y, como todas las APIs de Java que hacen de interfaz con sistemas host, es independiente de la implementación subyacente. Adicionalmente, especifica una interfaz de proveedor de servicio (SPI) que permite que las implementaciones del servicio de directorio sean enchufadas en el framework. Las implementaciones pueden hacer uso de un servidor, un fichero, o una base de datos; la elección depende del vendedor.

```
Connection conexion = null;
try{
    DataSource ds = getDataSource(request);
    conexion = ds.getConnection();
}catch(Exception sqlEx ) {
        erroresGlobales.add(ActionMessages.GLOBAL_MESSAGE , new ActionMessage("errors.conexionDB"));
        saveErrors(request,erroresGlobales);
        return (new ActionForward(mapping.getInput()));
}
```

4.3. Módulo de Consulta

En este módulo el usuario puede personalizar una consulta sobre las leyes almacenadas dentro del sistema, para que mediante el uso de JSP, HTML y MySQL se realice la consulta y se muestren las coincidencias de la misma.

Un criterio de búsqueda es construido con ayuda de campos de texto, casillas de verificación y el lenguaje interpretado de Javascript. En este caso, en la página JSP llamada "busqueda.jsp" se creó el método "sendQuery" creado en lenguaje javascript, el cual permite validar e interpretar cada selección de criterios con argumentos y conectores previamente definidos para ser concatenados posteriormente y asì formar una sentencia SQL válida.

La sentencia SQL resultado de la validación de criterios llamada "query" será enviada a la acción "busqueda.do" a través de un formulario con el método post.

La acción "busqueda.do" es definida dentro del archivo struts-config.xml como se muestra a continuación:

- Definición de JavaBean utilizado en esta acción, siendo "name" un identificador único y "type" el nombre cualificado de la clase Java de nuestro Bean de formulario.

```
<form-bean name="BusquedaForm"
        type="com.acle.busqueda.BusquedaForm"/>
```

- Definición de la acción, siendo el parámetro "path" la clase action en relación al contexto de la aplicación, "type" es el nombre totalmente cualificado de la clase Action de Java, el "name" es el elemento <form-bean> para usar con esta clase Action donde se validará que la cadena recibida "query" se encuentre en el formato apropiado. El alacance o "scope" del bean es de tipo "request", el cual implica que el bean será visible dentro de una sola página y almacenará su valor solo durante la petición de la misma; el valor "input" nos ayuda a que si una validación falla, la petición será re-direccionada por Struts a la página que contenía el formulario inicialmente, dada por este atributo. Una vez realizada la acción se realizará un re-direccionamiento a la página JSP llamada "resultados.jsp" como se muestra en la declaración del "forward".

```
<action  path="/pages/busqueda"
          type="com.acle.busqueda.BusquedaAction"
        name="BusquedaForm" scope="request"
        input="/pages/error.jsp">
          <forward name="success" path="/pages/resultados.jsp"/>
</action>
```

La metodología de Struts nos establece el flujo de operaciones de una acción a través de la interacción de dos o más archivos; en este caso, la acción "busqueda.do" mantendrá un flujo de interacción entre los archivos "BusquedaForm.java", "BusquedaAction.java" y "BusquedaBean.java", los cuales realizarán la búsqueda en los artículos de las normas previamente ingresadas en el sistema.

El archivo "BusquedaAction.java" obtendrá el parámetro "query" enviado desde la página JSP llamada "busqueda.jsp"; ésta operación se realizará a través del método "getQuery" declarado en el bean "BusquedaForm.java", como se muestra en la siguiente línea de código:

```
String query        = ((BusquedaForm) form).getQuery();
```

Posteriormente se realizará la conexión a la base de datos en MySQL para obtener un vector de resultados; se ejecuta la sentencia SQL llamada "query" por medio del método "getResultados" declarado en "BusquedaBean.java". Éste método ejecuta la sentencia SQL dentro del método público

"getArticulos", devolviendo como resultado un vector de artículos que coinciden con las características ingresadas por el usuario en la pagina "busqueda.jsp", dicho vector se almacena bajo el nombre de "articulos".

Una vez que se hayan obtenido los artículos se necesita de igual forma obtener las normas a las que pertenece cada artículo, así como los nombres de los archivos y transitorios respectivos.

Se realiza una nueva consulta a la base de datos para formar un nuevo vector con las normas, artículos, transitorios y archivos a través del método "getNormas", cuyos parámetros son la conexión a la base de datos y el vector "articulos".

El vector "resultados" contiene todas las características de los artículos para mostrar la información necesaria al usuario que consulta el sistema, éste vector se construye con elementos de tipo "Item", clase declarada dentro del archivo "Item.java" con los siguientes atributos:

```
private String tituloNorma;
private String tipo;
private String nivel;
private String factualiza;
private String fpublica;
private Vector articulos;
private Vector listado;
private Vector transitorios;
```

Por lo tanto, el vector "resultados" constará de uno o varios elementos "item" y cada elemento de esta clase contendrán tres vectores y cinco atributos de tipo String (cadena) que completarán la información que será mostrada en la siguiente página JSP llamada "resultados.jsp" como se muestra en la figura 4.1.

Figura 4.1 Resultados de la búsqueda.

La página "ítems.jsp" se muestra a través de "resultados.jsp", es decir, ésta última contiene un iframe que hace referencia a la página "ítems.jsp". Dicha página recorre el vector "resultados" que se generó después de haber ejecutado la acción "busqueda.do"; se ocupa el ciclo "for" para interactuar con el codigo HTML de la pagina y así generar dinámicamente la lista de los artículos que fueron resultado de la búsqueda por medio de la sentencia "out.print (String)".

Cada vez que se recorre el vector "resultados" se crea una liga a la página JSP llamada "consulta.jsp" para poder mostrar a detalle el texto del artículo seleccionado. Ésta página JSP cuenta con ayuda para la navegación entre los artículos del vector "resultados"; en la parte izquierda de la página se encuentra un iframe que hace referencia a la página JSP llamada "item.jsp" para mostrar el texto del artículo previamente seleccionado y en la parte superior derecha se encuentra un listado de los demás artículos que fueron resultado de la búsqueda; en la parte inferior derecha se cuenta con un listado de los artículos transitorios respectivos al artículo seleccionado, como se muestra en la figura 4.2.

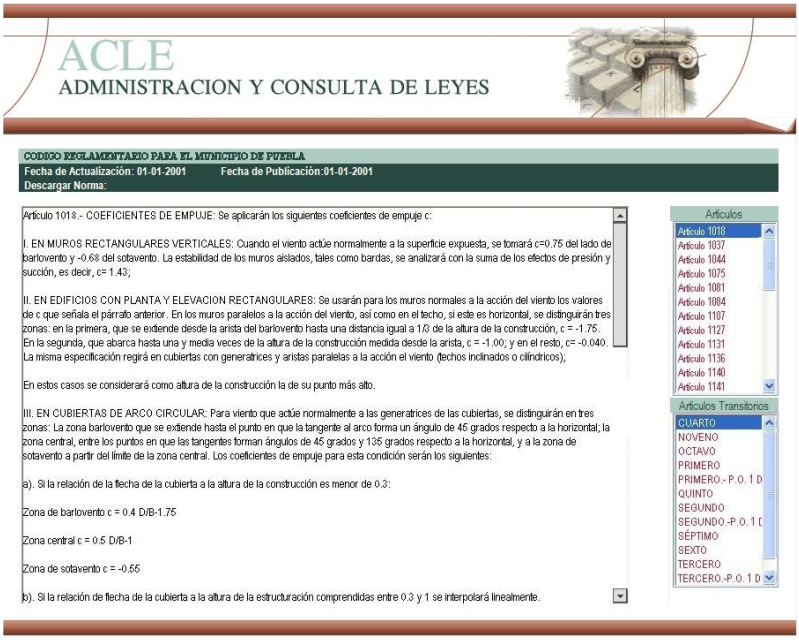

Figura 4.2 Consulta de Artículos

El módulo de consulta es descrito a través del archivo struts-config.xml y las declaraciones dentro de éste como se muestra en la figura 4.3.

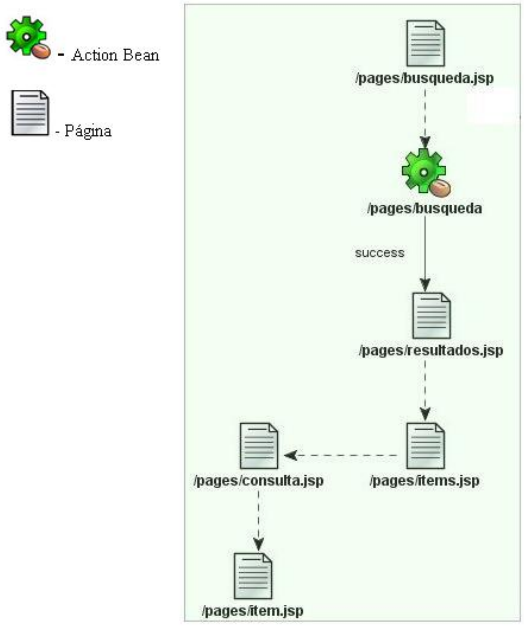

Figura 4.3 Módulo de Consulta

4.4. Módulo de Sugerencias

4.4.1. Envío de Sugerencias

El módulo de sugerencias permite al usuario enviar comentarios por medio de dos cuadros de texto, el primer cuadro de texto solicita un correo electrónico para contacto, validado por el método "checkmail" en lenguaje Javascript; continuando con un área de texto de mayor tamaño para el comentario y de igual forma validado pero con el método llamado "checkSuggestion", éste método verifica que el comentario no se encuentre vacio. Cada cuadro de texto es un dato de entrada o parámetro dentro del formulario llamado "buzón", cuyo método será "post" y los datos ingresados en el formulario serán enviados a la acción "sugerencias.do".

La acción "sugerencias.do" es definida dentro del archivo struts-config.xml como se muestra a continuación:

- Definición de JavaBean utilizado en esta acción, siendo "SugerenciaForm" el identificador único de la acción y "*com.acle.busqueda.SugerenciaForm*" el nombre cualificado de la clase Java Bean de nuestro formulario.

```
<form-bean name="SugerenciaForm "
        type="com.acle.busqueda.SugerenciaForm"/>
```

- Definición de la acción, donde la petición será re-direccionada por Struts a la página JSP llamada "busqueda.jsp" como se muestra en las siguientes líneas de código:

```
<action path="/pages/sugerencias"
                type="com.acle.busqueda.SugerenciaAction"
                name="SugerenciaForm"
                scope="request"
                input="/pages/error.jsp">
                <forward name="success" path="/pages/busqueda.jsp"/>
</action>
```

El flujo de operaciones en la acción "sugerencias.do" será entre los archivos "SugerenciaForm.java" y "SugerenciaAction.java".

El Java Bean "SugerenciaAction.java" recibe "email" y "sugerencia" como parámetros desde "SugerenciaForm.java" a través de los métodos "getEmail" y "getSugerencia" respectivamente, podemos observarlo en el código siguiente:

```
String email      = ((SugerenciaForm) form).getEmail();
String sugerencia = ((SugerenciaForm) form).getSugerencia();
```

Posteriormente se realizará la conexión a la base de datos MySQL para insertar a la tabla llamada "ACLE_SUGERENCIAS" los campos parámetro junto con la fecha en la que se realiza la inserción de la información y un identificador único para cada comentario. La inserción anteriormente mencionada se realiza a través de un objeto de tipo Statement que nos permite ejecutar sentencias SQL para hacer consultas o modificaciones en la base de datos, el objeto necesita ser instanciado como se muestra a continuación:

```
Statement stmt = conexion.createStatement();
```

De esta forma hemos creado un objeto Statement. Ahora podemos usar este objeto Statement para hacer modificaciones en la base de datos a través del lenguaje SQL. Para realizar modificaciones, es decir, instrucciones INSERT, UPDATE o DELETE, se usa el método executeUpdate pasando como parámetro una cadena de texto String que contenga la instrucción SQL.

```
stmt.executeUpdate("INSERT INTO...");
```

Al terminar la inserción, se realiza la re-dirección a la página JSP previamente definida en la acción; el flujo del módulo de sugerencias se muestra gráficamente en la figura 4.4.

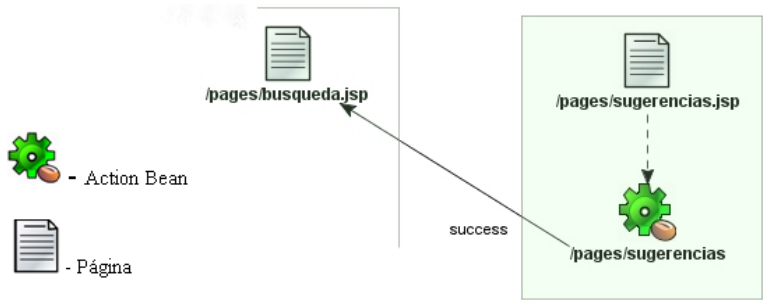

Figura 4.4 Envío de Sugerencias

4.4.2. Consulta de Sugerencias

Este módulo permite al usuario administrador consultar todas las sugerencias recibidas a través del sistema.

Al pulsar en el menú principal el botón de "sugerencias" se ejecutará la acción "lstsugerencias.do" definida como se muestra a continuación:

```
<action path="/pages/lstsugerencias"
        type="com.acle.administracion.auditoria.ListSugerenciasAction"
        scope="request"
        input="/pages/error.jsp">
          <forward name="success" path="/pages/lstsugerencias.jsp"/>
</action>
```

La página JSP llamada "lstsugerencias.jsp" contiene un iframe haciendo referencia a la página JSP llamada "listado.jsp", ésta última recorre el vector resultado de realizar una consulta a la base de datos por medio del Java Bean llamado "ListSugerenciasAction.java" para obtener todas las sugerencias registradas en la tabla "ACLE_SUGERENCIAS", véase figura 4.5.

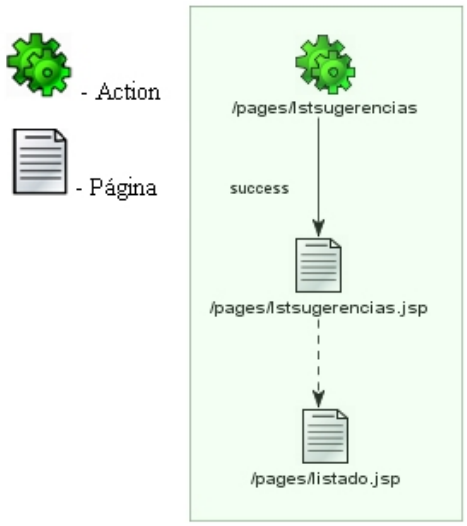

Figura 4.5 Consulta de Sugerencias

4.5. Módulo de Ingreso al Sistema

El sistema ACLE contiene un módulo de administración como se explica en los capítulos 2 y 3, para ingresar a éste se necesita presionar el botón "Administración" del menú principal, este botón hace un re-direccionamiento a la página JSP llamada "login.jsp" que cuenta con dos cuadros de texto para ingresar el usuario (login) y contraseña (password), datos proporcionados por el administrador del sistema.

Los parámetros login y password son enviados por un formulario a través del método "post" a la acción "login.do" definida como se muestra a continuación:

- Definición de JavaBean utilizado en esta acción, siendo "LoginForm" el identificador único de la acción y "*com.acle.administracion.LoginForm*" el nombre cualificado de la clase Java Bean de nuestro formulario.

  ```
  <form-bean name="LoginForm"
          type="com.acle.administracion.LoginForm"/>
  ```

- Definición de la acción, donde la petición será re-direccionada por Struts a la página JSP llamada "norma.jsp" como se muestra en las siguientes líneas de código:

  ```
  <action path="/pages/login"
          type="com.acle.administracion.LoginAction"
          name="LoginForm"
  ```

```
        scope="request"
        input="/pages/errorLogin.jsp">
        <forward name="success" path="/pages/norma.jsp"/>
</action>
```

La clase LoginAction.java realiza una consulta a la tabla "ACLE_USUARIO", para verificar si existe un usuario registrado que coincida con las características proporcionadas en la página JSP llamada "login.jsp". Si el resultado es afirmativo entonces se realizará una re-dirección a la página JSP llamada "norma.jsp"; en caso contrario se mostrará un mensaje al usuario informando que no existe ningún usuario registrado y que verifique su información.

La representación gráfica de este módulo se muestra en la figura 4.6.

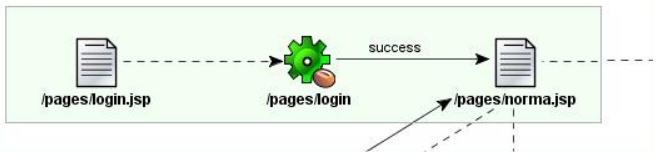

Figura 4.6 Módulo de Ingreso al sistema

4.6. Módulo de Administración de normas

El módulo de administración de normas se compone de tres acciones: agregar norma, modificar norma y eliminar norma. Cualquiera de estas acciones puede ser seleccionada en la página principal del módulo de administración que es la página JSP llamada "norma.jsp".

La representación gráfica del flujo de este módulo se muestra a en la figura 4.7, que se muestra a continuación.

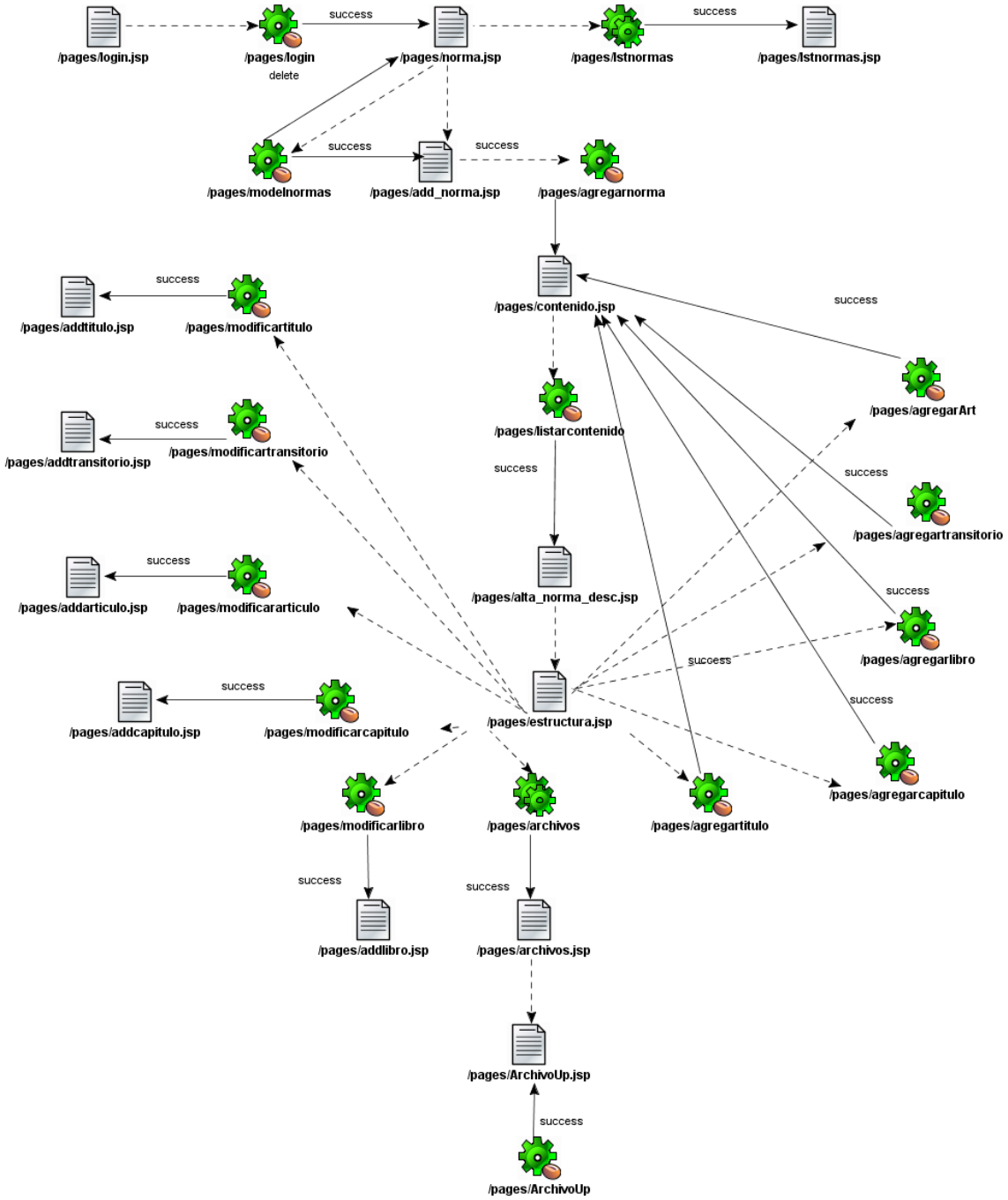

Figura 4.7 Administración de normas

4.6.1. Agregar Norma

Los datos necesarios para ingresar una norma se dividieron en dos partes:

- Datos generales de la norma:

Nombre del Registro	**Descripción**
Fecha de Publicación	Se refiere a la fecha en la que se publicó la norma jurídica.
Fecha de última reforma	Se refiere a la fecha en la que se realizó la última reforma a la norma jurídica.
Nombre	Se refiere al título principal de la norma jurídica.
Descripción	Se refiere a la descripción que acompaña al título principal de la norma jurídica.
Tipo de Norma	Se refiere al tipo de norma jurídica a agregar al sistema
Nivel	Se refiere al nivel de la norma jurídica a agregar al sistema.

Tabla 4.1 Datos generales

- Estructura de la norma:

Nombre del Registro	Descripción
Archivo	Se refiere a la dirección local del archivo que contiene la norma jurídica.
Libro	Se refiere a el(los) libro(s) correspondiente(s) a la norma jurídica que se desea ingresar.
Titulo	Se refiere a el(los) título(s) correspondiente(s) a la norma jurídica que se desea ingresar.
Capitulo	Se refiere a el(los) capítulo(s) correspondiente(s) a la norma jurídica que se desea ingresar.
Articulo	Se refiere a el(los) artículo(s) correspondiente(s) a la norma jurídica que se desea ingresar.
Transitorio	Se refiere a el(los) transitorio(s) correspondiente(s) a la norma jurídica que se desea ingresar.

Tabla 4.2 Estructura de la norma

Al presionar la opción "Agregar Norma" de la página JSP "norma.jsp" se realiza un re-direccionamiento a la página llamada "add_norma.jsp" que contiene los datos generales de la norma y por medio de cuadros de texto y botones de opción se pasan como parámetros a la acción "agregarnorma.do" por medio del método "post" del formulario llamado "forma1", éste formulario cuenta con una variable llamada "alta" con valor igual a 1 para indicar que es el alta de la norma.

La definición de la acción "agregarnorma.do" se muestra como sigue:

- Definición de JavaBean utilizado en esta acción, siendo "AgregarNormaForm" el identificador único de la acción y "*com.acle.administracion.normas.AgregarNormaForm*" el nombre cualificado de la clase Java Bean de nuestro formulario.

```
<form-bean name="AgregarNormaForm"
        type="com.acle.administracion.normas.AgregarNormaForm"/>
```

- Definición de la acción, donde la petición será re-direccionada por Struts a la página JSP llamada "contenido.jsp" como se muestra en las siguientes líneas de código:

```
<action   path="/pages/agregarnorma"
        type="com.acle.administracion.normas.AgregarNormaAction"
        name="AgregarNormaForm"
        scope="request"
        input="/pages/error.jsp">
        <forward name="success" path="/pages/contenido.jsp"/>
</action>
```

El archivo "AgregarNormaAction.java" obtiene los datos generales de la norma a través de "AgregarNormaForm.java", este Java Bean valida que el título, el tipo, el nivel y la fecha de publicación de la norma no se encuentren vacíos.

Se crea el objeto "newnorma" de tipo "Norma" con el constructor definido en el archivo "Norma.java" como sigue:

```
//definición del objeto
Norma newnorma =  new Norma (idNorma,norma,descripcion,tipo,nivel,factualiza,fpublica);
//Constructor
public Norma(int idNorma, String norma, String descripcion,
        int tipo, int nivel, String factualiza, String fpublica,
        Vector lista){
    this.idNorma=idNorma;
    this.norma=norma;
    this.descripcion=descripcion;
    this.tipo=tipo;
    this.nivel=nivel;
    this.factualiza=factualiza;
    this.fpublica=fpublica;
    this.lista = lista;
    }
```

Una vez creado el objeto "newnorma" que contiene los datos enviados desde el formulario de la página JSP "add_norma.jsp", se valida si la variable "alta" tiene el valor 1, si es una comparación exitosa entonces se hace llamado al método "insertNorma" declarado en "AgregarNormaBean.java" cuyos

parámetros serán la conexión creada, el objeto "request" de tipo "HttpServletRequest" y el objeto "newnorma".

```
AgregarNormaBean lb = new AgregarNormaBean();
idley  = lb.insertNorma(conexion, request, newnorma);
```

Este método regresará el identificador único de la norma ingresada y se almacenará en la variable "idley" para después ser agregado como una referencia en el módulo de auditoría.

Se realizan tres operaciones principales en este método:

1. Inserción de la norma en la tabla "ACLE_NORMA" a través de un objeto de tipo "Statement" y se ejecuta también la sentencia "COMMIT" para que los cambios sean reflejados en la base de datos.

   ```
   stmt = conexion.createStatement();
   stmt.executeUpdate(query);
   stmt.executeUpdate("COMMIT");
   ```

2. Obtención del identificador de la norma ingresada.
3. Envío a sesión de las variables necesarias para continuar la captura de los datos de la estructura de la norma.

   ```
   session.setAttribute("idNorma",idNorma);
   session.setAttribute("norma_descripcion",newnorma.getNorma());
   session.setAttribute("norma",newnorma);
   session.setAttribute("idLibro","0");
   session.setAttribute("idTitulo","0");
   session.setAttribute("idCapitulo","0");
   session.setAttribute("idArticulo","0");
   ```

Posteriormente se muestra la página JSP "contenido.jsp" que contiene un formulario con los identificadores de los valores de una norma, los cuales son enviados a la acción "listarcontenido.do" a través del método "post" del formulario.

```
<form name="estructura" method="post" >
<input name="idnorma" id="idnorma" type="hidden" value="<%=session.getAttribute("idNorma")%>">
<input name="idlibro" id="idlibro" type="hidden" value="<%=session.getAttribute("idLibro")%>">
<input name="idtitulo" id="idtitulo" type="hidden" value="<%=session.getAttribute("idTitulo")%>">
<input name="idcapitulo" id="idcapitulo" type="hidden" value="<%=session.getAttribute("idCapitulo")%>">
<input name="idarticulo" id="idarticulo" type="hidden" value="<%=session.getAttribute("idArticulo")%>">
<input name="idtransitorio" id="idtransitorio" type="hidden"
value="<%=session.getAttribute("idTransitorio")%>">
</form>
```

La acción "listarcontenido.do" se define a continuación:

- Definición de JavaBean utilizado en esta acción, siendo "AgregarNormaForm" el identificador único de la acción y "*com.acle.administracion.normas.AgregarNormaForm*" el nombre cualificado de la clase Java Bean de nuestro formulario.

  ```
  <form-bean name="ListarForm"
          type="com.acle.normas.ListarForm"/>
  ```

- Definición de la acción, donde la petición será re-direccionada por Struts a la página JSP llamada "alta_norma_desc.jsp" como se muestra en las siguientes líneas de código:

  ```
  <action   path="/pages/listarcontenido"
          type="com.acle.administracion.normas.ListarAction"
          name="ListarForm"
          scope="request"
          input="/pages/error.jsp">
          <forward name="success" path="/pages/alta_norma_desc.jsp"/>
  </action>
  ```

Debido a que se ingresarán los libros, títulos, capítulos y artículos en diferentes pantallas, se necesitan recuperar todos los cambios realizados a la estructura de la norma que se está ingresando para poder mostrarlos y así continuar con la captura. La acción "listarcontenido.do" realiza estas operaciones con el uso de tres archivos "ListarForm.java", "ListarAction.java" y "ListarBean.java".

El Java Bean Action obtiene el identificador de la norma a través del Bean Form y es almacenado en la variable "intnorma".

```
int idnorma = ((ListarForm) form).getIdnorma();
```

Para obtener los vectores de libros, títulos, capítulos y artículos se crea una instancia de "ListarBean" llamada "contenido" para poder utilizar los métodos definidos en el archivo "ListarBean.java".

```
ListarBean contenido = new ListarBean();
```

Primeramente se declaran los vectores, a continuación se hace el llamado a los métodos que realizarán la consulta a la base de datos en la tabla adecuada para obtener la información necesaria, todos los métodos ocupados tienen como parámetro la conexión y el identificador de la norma de la cual queremos recuperar información.

```
//declaración de vectores
Vector  catLibros;
Vector  catTitulos;
Vector  catCapitulos;
Vector  catArticulos;
Vector  catTransitorios;

catLibros = (Vector)contenido.getLibro(conexion, request,idnorma);
catTitulos = (Vector)contenido.getTitulo(conexion, request,idnorma);
catCapitulos = (Vector)contenido.getCapitulo(conexion, request,idnorma);
catArticulos = (Vector)contenido.getArticulo(conexion, request,idnorma);
catTransitorios = (Vector)contenido.getTransitorio(conexion, request,idnorma);
```

Cada uno de los métodos utilizados realizan una consulta a la tabla correspondiente y el resultado de la misma es recorrido por un ciclo "while" para crear un elemento de tipo "Catalogo" y ser almacenado en el vector temporal llamado "cats". La diferencia entre cada uno de estos métodos es la forma en la que se almacena la información obtenida en el elemento de tipo "Catalogo" debido a la estructura que se necesita para mostrar la norma en la página JSP llamada "estructura.jsp". Ésta página es mostrada a través de un iframe en la página JSP llamada "alta_norma_desc.jsp" como se muestra en la figura 4.8.

Figura 4.8 Estructura de la norma.

En la página “estructura.jsp” se muestran seis elementos de tipo lista donde se muestran las operaciones, elementos, libro/transitorio, título, capítulo y articulo respectivamente. El manejo de estos elementos dentro de esta representación de información evita bastante la navegación entre diferentes ventanas. Éstos elementos son creados con ayuda de un archivo en lenguaje Javascript llamado “chainedselects.js”, contiene funciones y ciclos que permite unir multiples elementos de tipo lista para que en la selección de una lista “padre” se pueda construir un elemento “hijo” también de tipo lista. El script antes mencionado es llamado al ingresar a la página “estructura.jsp” de forma automática. La construcción de los elementos de tipo “lista“ se compone principalmente de ciclos “for” y de comparaciones en lenguaje java, estas comparaciones son propias de la estructura de los elementos en cada vector creado desde la acción “listarcontenido.do”. A continuación se muestra un bloque que contiene código para la creación del elemento lista con el vector de libros.

```
<%
Vector catLibros = (session.getAttribute("catLibros")!=null?(Vector)session.getAttribute("catLibros"):null);
 if(catLibros != null){
        if (!catLibros.isEmpty()){
                for (int i=0; i<catLibros.size(); i++){
                        String libro = ((Catalogo)catLibros.elementAt(i)).getDescripcion();
                        int llavelibro  = ((Catalogo)catLibros.elementAt(i)).getLlave();
                        if (llavelibro == idLibro){
                        out.println("addOption(\"cs-sub-libro\",\""+libro+"\",\""+llavelibro+"\",1);");
                        }else{
                        out.println("addOption(\"cs-sub-libro\",\""+libro+"\",\""+llavelibro+"\");");
                        }
                }//fin for (int i=0; i<catLibros.size(); i++){
        }else{
                out.println("addOption(\"cs-sub-libro\",\"<Vacio>\",\"vacio\");");
        }
} %>
```

Una vez creada la estructura anterior, el usuario administrador puede seleccionar la operación que desea realizar para capturar la segunda parte de la norma y presionar el botón “Realizar Operación” que llamará a la función de javascript llamada “acciones”.

La función javascript “acciones” es la que re-direcciona la acción a realizar en base a la operación y elemento que el usuario administrador haya seleccionado. A continuación se listan las posibles operaciones a seleccionar junto con la acción que se realiza como resultado de la selección.

Operación	**Elemento**	**Acción**	
Agregar	Libro	addlibro.jsp	Muestra un formulario para capturar el título y descripcion del libro.
Agregar	Título	addtitulo.jsp	Muestra un formulario para capturar el título y descripcion del título.
Agregar	Capítulo	addcapitulo.jsp	Muestra un formulario para capturar el título y descripcion del capítulo.
Agregar	Artículo	addarticulo.jsp	Muestra un formulario en una nueva ventana para capturar el título y descripcion del artículo.
Agregar	Transitorio	addtransitorio.jsp	Muestra un formulario para capturar el título y descripcion del transitorio.
Modificar/Eliminar	Libro	modificarlibro.do	Acción que recupera los datos del libro a modificar/eliminar.
Modificar/Eliminar	Título	modificartitulo.do	Acción que recupera los datos del título a modificar/eliminar.
Modificar/Eliminar	Capítulo	modificarcapitulo.do	Acción que recupera los datos del capítulo a modificar/eliminar.
Modificar/Eliminar	Artículo	modificararticulo.do	Acción que recupera los datos del artículo a modificar/eliminar.
Modificar/Eliminar	Transitorio	modificartransitorio.do	Acción que recupera los datos del transitorio a modificar/eliminar.
Archivos		archivos.do	

Tabla 4.3 Operaciones

Las acciones ocupadas con la operación Modificar, Eliminar y Archivos se declaran de igual forma en el archivo struts-config.xml como se muestra a continuación:

```
<form-bean name="ModificarLibroForm"
                    type="com.acle.administracion.normas.ModificarLibroForm"/>

<action   path="/pages/modificarlibro"
                    type="com.acle.administracion.normas.ModificarLibroAction"
          name="ModificarLibroForm"
          scope="request"
          input="/pages/error.jsp">
          <forward name="success" path="/pages/addlibro.jsp"/>
</action>
```

```
<form-bean name="ModificarTituloForm"
                    type="com.acle.administracion.normas.ModificarTituloForm"/>

<action path="/pages/modificartitulo"
                    type="com.acle.administracion.normas.ModificarTituloAction"
          name="ModificarTituloForm"
          scope="request"
          input="/pages/error.jsp">
          <forward name="success" path="/pages/addtitulo.jsp"/>
</action>
```

```
<form-bean name="ModificarCapituloForm"
                   type="com.acle.administracion.normas.ModificarCapituloForm"/>

<action
          path="/pages/modificarcapitulo"
          type="com.acle.administracion.normas.ModificarCapituloAction"
          name="ModificarCapituloForm"
          scope="request"
          input="/pages/error.jsp">
          <forward name="success" path="/pages/addcapitulo.jsp"/>
</action>
```

```
<form-bean name="ModificarArticuloForm"
                    type="com.acle.administracion.normas.ModificarArticuloForm"/>

<action path="/pages/modificararticulo"
          type="com.acle.administracion.normas.ModificarArticuloAction"
          name="ModificarArticuloForm"
          scope="request"
          input="/pages/error.jsp">
          <forward name="success" path="/pages/addarticulo.jsp"/>
</action>
```

```
<form-bean name="ModificarTransitorioForm"
                    type="com.acle.administracion.normas.ModificarTransitorioForm"/>

<action path="/pages/modificartransitorio"
          type="com.acle.administracion.normas.ModificarTransitorioAction"
          name="ModificarTransitorioForm"
          scope="request"
          input="/pages/error.jsp">
          <forward name="success" path="/pages/addtransitorio.jsp"/>
</action>
```

Las acciones que se muestran en la parte superior realizan las mismas operaciones, tales como obtener los datos almacenados en la base de datos para su modificación o eliminación pero de diferente elemento. Ésta información se muestra en la página JSP respectiva, la diferencia entre las dos operaciones son las acciones que se realizarán al presionar los botones “Aceptar” o “Eliminar” según sea el caso.

La acción “archivos.do”, a diferencia de las acciones anteriores, devuelve un vector de archivos asociados a la norma, su definición se muestra en el siguiente bloque.

```
<action path="/pages/archivos"
        type="com.acle.administracion.normas.ListArchivosAction"
        scope="request"
        input="/pages/error.jsp">
        <forward name="success" path="/pages/archivos.jsp"/>
</action>
```

En la figura 4.9 se visualiza el vector resultado que corresponde a la página JSP llamada “archivos.jsp”.

Figura 4.9 Archivos asociados a la norma.

La liga “Subir un Archivo” nos re-direcciona a la página JSP llamada “ArchivoUp.jsp” que contiene un cuadro de texto y un botón “Examinar” que permite buscar dentro de la maquina cliente el archivo que se desea asociar a la ley y posteriormente presionar el botón “Agregar Archivo” que hace referencia a la acción “ArchivoUp.do”, ver figura 4.10.

Figura 4.10 Agregar un archivo.

La acción “ArchivoUp.do” tiene como parámetro la ruta ingresada en el cuadro de texto de la figura 4.10 y su definición se muestra en el siguiente bloque.

```
<form-bean name="ArchivosForm"
                type="com.acle.administracion.normas.ArchivosForm"/>

<action path="/pages/ArchivoUp"
        type="com.acle.administracion.normas.ArchivosAction"
        name="ArchivosForm"
        scope="request"
        validate="true"
       input="/pages/error.jsp">
        <forward name="success" path="/pages/ArchivoUp.jsp"/>
</action>
```

Una vez proporcionada la ruta del archivo, la acción “ArchivoUp.do” obtiene la ruta del servidor que está destinada para almacenar los archivos.

La clase “InputStream” nos permite leer datos de los archivos del sistema, así como escribir en ellos y la clase “Properties” que nos permite crear archivos de configuración complejos y manejarlos con facilidad. Tanto la estructura de la clase como la sintaxis de los archivos de propiedades están comentadas en la documentación del API y puede ser consultada online.

Se crea un archivo con extensión “.prop” donde se coloca la siguiente cadena: “file.path=”, seguida de la ruta física deseada dentro de nuestro servidor. El archivo se llama “Fileproperties.prop” y se accesa mediante las siguientes líneas:

```
InputStream fis = getClass().getResourceAsStream("../../../../../../Fileproperties.prop");
Properties props = new Properties();
props.load(fis);
path = props.getProperty("file.path");
```

Ya que se obtuvo la ruta donde se desean almacenar los archivos se ocupa la clase “OutputStream” y la siguiente rutina para copiar el archivo seleccionado en la ruta definida en la cadena “file.path” del archivo “Fileproperties.prop”.

```
InputStream in = myFile.getInputStream();
OutputStream aut= new FileOutputStream(path + fileName);
byte[] buffer= new byte[256];
while (true) {
    int n= in.read(buffer);
    if (n < 0)
      break;
     aut.write(buffer, 0, n);
}
in.close();
aut.close();
```

Una vez colocado el archivo en el servidor se inserta en la tabla "ACLE_ARCHIVO" el nombre del archivo que se agrego, el identificador de la norma a la que está asociado y un identificador único para el archivo.

4.6.2. Modificar Norma

Para modificar una norma, primero se debe de seleccionar la norma deseada y presionar el botón "Modificar Norma" y se ejecutará la acción "modelnormas.do" que se define a continuación.

```
<form-bean name="ModElimForm"
                type="com.acle.administracion.normas.ModElimForm"/>

<action path="/pages/modelnormas"
        type="com.acle.administracion.normas.ModElimAction"
        name="ModElimForm"
        scope="request"
        input="/pages/error.jsp">
        <forward name="success"  path="/pages/add_norma.jsp?modificar=1"/>
        <forward name="delete"  path="/pages/norma.jsp"/>
</action>
```

El archivo "ModElimAction.java" obtiene como parámetro, por medio del bean "ModElimForm", el identificador de la norma junto con una bandera que hace diferencia entre la acción de modificación y eliminación.

```
int idNorma = ((ModElimForm) form).getIdNorma();
int eliminar = ((ModElimForm) form).getBan();
```

Si la variable "eliminar" tiene un valor diferente a 1 se hace el llamado al método "getNormaData" declarado en el bean "ModElimBean", que permite obtener de la tabla "ACLE_NORMA" todos los datos de la norma para ser modificados; para esto se construye un objeto de tipo "Norma" con los datos obtenidos. El constructor utilizado para crear el objeto se puede ver en el siguiente bloque de código:

```
public Norma(int idNorma, String norma, String descripcion, int tipo, int nivel, String factualiza, String
fpublica){
        this.idNorma=idNorma;
        this.norma=norma;
        this.descripcion=descripcion;
        this.tipo=tipo;
        this.nivel=nivel;
        this.factualiza=factualiza;
        this.fpublica=fpublica;
}
```

Se colocan en sesión el objeto construido y se re-direcciona la acción a la página JSP llamada "add_norma.jsp" con una variable de parámetro "modificar=1", ésta indica que se está modificando información y es necesaria debido a que se utiliza la misma página JSP que se ocupa para dar de alta una norma solo que ahora esta página contendrá la información que se almaceno con anterioridad.

4.6.3. Eliminar Norma

En la página JSP principal llamada “norma.jsp” se presiona el botón “Eliminar Norma” y la función “del()” de Javascript emitirá una ventana de confirmación, si el usuario la confirma se re-direccionará a la acción “modelnormas.do” con el parámetro “ban” igual a 1, para indicar que se desea eliminar la norma seleccionada. A continuación se muestra el código Javascript que emite la ventana de confirmación.

```
function del()
{
    if (document.ifrnormas.listado.rdnormas.value>0){
       answer = confirm("Desea eliminar la norma seleccionada?")
       if (answer !=0){

document.form1.action="modelnormas.do?ban=1&idNorma="+document.ifrnormas.listado.rdnormas.value;
          document.form1.submit();
       }
   }else{alert("Seleccione una Norma");}
}
```

La acción “modelnormas.do”, cuya definición se encuentra en el modulo de “Modificar Norma”, obtiene los mismos parámetros que en la modificación de una norma, solo que ahora se realiza un llamado al método “getBorrarNorma” definido en el bean “ModElimBean.java”. En este método se elimina la norma y toda la información que tenga asociada en las diferentes tablas.

4.7. Módulo de Auditoria

4.7.1. Registro de movimientos

La función del módulo de auditoría es registrar los movimientos que se realizan en la norma, a continuación se muestra el listado del catálogo de las acciones que se registran en cada acción.

	Acción
1	AGREGAR NORMA
2	ELIMINAR NORMA
3	MODIFICAR NORMA
4	AGREGAR TITULO
5	ELIMINAR TITULO
6	MODIFICAR TITULO
7	AGREGAR CAPITULO
8	ELIMINAR CAPITULO
9	MODIFICAR CAPITULO
10	AGREGAR ARTICULO
11	ELIMINAR ARTICULO
12	MODIFICAR ARTICULO
13	AGREGAR ARCHIVO
14	ELIMINAR ARCHIVO
15	AGREGAR USUARIO
16	ELIMINAR USUARIO
17	MODIFICAR USUARIO
18	ELIMINAR LIBRO
19	AGREGAR LIBRO
20	MODIFICAR LIBRO
22	AGREGAR TRANSITORIO
23	MODIFICAR TRANSITORIO
24	ELIMINAR TRANSITORIO

Cada acción realizada en una clase Java Bean contiene un módulo que registra el movimiento obteniendo como primer paso el identificador de la acción que deseamos registrar. Posteriormente, se crea un objeto de tipo "Movimiento", seguido del llamado al método "insertMov" definido en el bean llamado "AuditoriaBean" que realiza una inserción a la tabla "ACLE_MOVIMIENTO".

4.7.2. Consulta de movimientos

El módulo de auditoría permite al usuario administrador consultar los movimientos realizados a las normas dentro del sistema.

La acción "auditoria.do" es invocada para que a través del archivo "AuditoriaAction.java" se consulte la información necesaria de la base de datos MySQL sin ayuda de ningún <form-bean> en este caso, la declaración de ésta acción se muestra a continuación:

- Definición de la acción; como se mencionó en el párrafo anterior, esta acción no cuenta con el atributo "name" debido a que no se necesita de ningún parámetro y por lo tanto no se invocará una validación previa; la petición será re-direccionada por Struts a la página JSP llamada "auditoria.jsp" como se muestra en las siguientes líneas de código:

```
<action path="/pages/auditoria"
        type="com.acle.administracion.auditoria.AuditoriaAction"
        scope="request"
        input="/pages/error.jsp">
        <forward name="success" path="/pages/auditoria.jsp"/>
</action>
```

El Java Bean "AuditoriaAction.java" crea una conexión con la base de datos MySQL, se ejecuta una consulta sobre las tablas "ACLE_MOVIMIENTO", "ACLE_ACCION" y "ACLE_USUARIO" obteniendo un vector con objetos de tipo "Movimiento" usando el constuctor llamado "Movimiento". Este constructor es declarado como se muestra a continuación:

```
public Movimiento(int idmov, String norma, String accion, String login, String fecha, int
idNorma) {
    this.idmov = idmov;
    this.norma = norma;
    this.accion = accion;
    this.login = login;
    this.fecha = fecha;
    this.idNorma = idNorma;
  }
```

La página "auditoria.jsp" contiene un iframe que hace referencia a la página "acciones.jsp" que muestra el vector generado por la acción "auditoria.do" llamado "movimientos" como se muestra en la figura 4.11.

Figura 4.11 Listado de normas modificadas.

La figura 4.11 muestra un listado de normas y cada una cuenta con un botón en el extremo izquierdo, este botón es un input que hace un llamado al método de Javascript llamado "NewWindow" que abre una nueva ventana mostrando la página "detalle.jsp", dicha página muestra el vector obtenido de ejecutar la acción "detalle.do" cuyo parámetro es el identificador de la norma seleccionada.

La definición de la acción "detalle.do" en el archivo struts-config.xml se muestra en el siguiente bloque:

```
<action path="/pages/detalle"
        type="com.acle.administracion.auditoria.DetalleAction"
        scope="request"
        input="/pages/error.jsp">
        <forward name="success" path="/pages/detalle.jsp"/>
</action>
```

El archivo JSP llamado "detalle.jsp" es el re-direccionamiento final de la acción y aquí se muestra de forma dinámica la construcción de un listado de acciones realizadas en una norma previamente seleccionada, en la figura 4.12 se muestra un ejemplo de esta página.

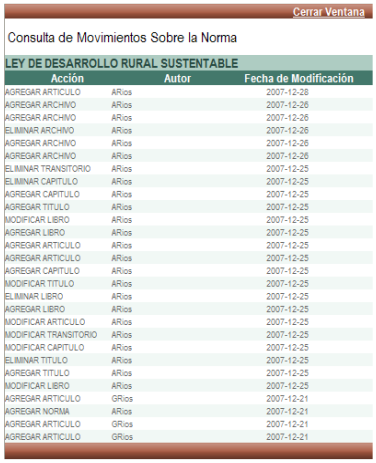

Figura 4.12 Acciones realizadas en una norma.

Al igual que en los módulos anteriores se representa gráficamente el flujo de este módulo en la figura 4.13, definido en el archivo struts-config.xml.

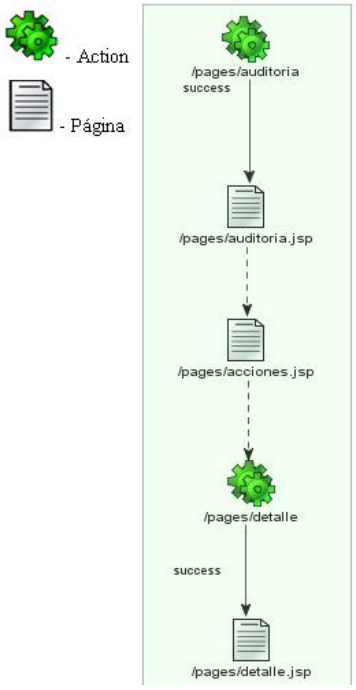

Figura 4.13 Módulo de Auditoria

4.8. Módulo de Administración de Usuarios

La administración de usuarios corresponde a dar de alta, eliminar y modificar los usuarios que interactuarán con el sistema.

Al ingresar al módulo de administración de usuarios se ejecuta la acción "lstusuarios.do" que realiza una consulta a la tabla "ACLE_USUARIO" para mostrar los usuarios que se encuentran registrados.

La declaración de la acción "lstusuarios.do" se muestra a continuación.

```
<action path="/pages/lstusuarios"
        type="com.acle.administracion.ListUsuariosAction"
        scope="request"
        input="/pages/error.jsp">
        <forward name="success" path="/pages/usuarios.jsp"/>
</action>
```

El resultado de ejecutar esta acción es un vector llamado de objetos de tipo "Usuario". Cada objeto contiene el nombre del recurso humano, usuario y contraseña como se muestra en la figura 4.14.

Figura 4.14 Listado de usuarios.

El listado de usuarios se muestra en la página JSP llamada "usuarios.jsp" que contiene un iframe haciendo referencia a la página JSP llamada "lstusuarios.jsp". Ésta última recorre el vector llamado "usu" y lo muestra haciendo la interacción entre código HTML y código Java.

En la parte inferior de la página principal de la lista de los usuarios, "usuarios.jsp" se encuentran tres botones con las operaciones de Agregar, Modificar y Eliminar usuario.

4.8.1. Agregar Usuario

Para agregar un usuario se hace un re-direccionamiento a la página JSP llamada "addusuario.jsp" que contiene un formulario con tres campos de texto (login, password y nombre); el campo de texto "password" es una caja de texto de tipo "password" para que la contraseña proporcionada no se muestre durante el proceso. Esta página JSP cuenta con dos botones, "Aceptar" y "Cancelar"; al presionar el botón "Aceptar" se ejecuta la acción "agregarusuario.do" , su declaración se muestra a continuación.

```
<form-bean name="AgregarUsuarioForm"
        type="com.acle.administracion.AgregarUsuarioForm"/>

<action path="/pages/agregarusuario"
        type="com.acle.administracion.AgregarUsuarioAction"
        name="AgregarUsuarioForm"
```

```
            scope="request"
            input="/pages/error.jsp">
            <forward name="success" path="/pages/lstusuarios.do"/>
        </action>
```

El archivo "AgregarUsuarioAction.java" recibe como parámetros el identificador del usuario, que en este caso será nulo; también recibe como parámetro el login, password, nombre y una variable llamada opción cuyo valor será 1, para que se haga el llamado al método llamado "insertUsuario" declarado en "AgregarUsuarioBean.java". Este método realizará la inserción del nuevo usuario regresando un identificador único generado automáticamente con el campo auto numérico del identificador en la tabla "ACLE_USUARIO".

4.8.2. Modificar Usuario

Para modificar un usuario, primero se debe de seleccionar el usuario deseado en la pagina "usuarios.jsp" y presionar el botón "Modificar Usuario", se ejecutará la acción "modelusuarios.do" que se define a continuación.

```
<form-bean name="ModElimUsuarioForm"
        type="com.acle.administracion.ModElimUsuarioForm"/>

 <action path="/pages/modelusuarios"
        type="com.acle.administracion.ModElimUsuarioAction"
        name="ModElimUsuarioForm"
        scope="request"
        input="/pages/error.jsp">
        <forward name="success"  path="/pages/addusuario.jsp?modificar=1"/>
        <forward name="delete"  path="/pages/lstusuarios.do"/>
</action>
```

El archivo"ModElimUsuarioAction.java"obtiene como parámetro, por medio del bean "ModElimUsuarioForm", el identificador del usuario junto con una bandera que hace diferencia entre la acción de modificación y eliminación.

```
int idusuario = ((ModElimUsuarioForm) form).getIdusuario();
int eliminar = ((ModElimUsuarioForm) form).getBan();
```

Si la variable "eliminar" tiene un valor diferente a 1 se hace el llamado al método "getUsuarioData" declarado en el bean "ModElimUsuarioBean", que permite obtener de la tabla "ACLE_USUARIO" todos los datos del usuario para ser modificados; para esto se construye un objeto de tipo "Usuario" con los datos obtenidos. El constructor utilizado para crear el objeto se puede ver en el siguiente bloque de código:

```
public Usuario(int idusuario, String login, String password, String nombre){
        this.idusuario=idusuario;
        this.login=login;
        this.password=password;
        this.nombre=nombre;
}
```

Se colocan en sesión el objeto construido y se re-direcciona la acción a la página JSP llamada "addusuario.jsp" con una variable de parámetro "modificar=1", ésta indica que se está modificando información y es necesaria debido a que se utiliza la misma página JSP que se ocupa para dar de alta una norma solo que ahora esta página contendrá la información que se almaceno con anterioridad.

4.8.3. Eliminar Usuario

En la página JSP principal llamada "usuarios.jsp" se presiona el botón "Eliminar Usuario" y la función "del ()" de Javascript emitirá una ventana de confirmación, si el usuario la confirma se re-direccionará a la acción "modelusuarios.do" con el parámetro "ban" igual a 1, para indicar que se desea

eliminar el usuario seleccionado. A continuación se muestra el código Javascript que emite la ventana de confirmación.

```
function del()
{
     if (ifrusuarios.listado.rdusuarios!=null && ifrusuarios.listado.rdusuarios.value>0){
          answer = confirm("Desea eliminar el usuario seleccionado?")
          if (answer !=0){

document.form1.action="modelusuarios.do?ban=1&idusuario="+document.ifrusuarios.listado.rdusuarios.value;
                    document.form1.submit();
          }
   }else{alert("Seleccione un usuario");}
}
```

La acción “modelusuarios.do”, cuya definición se encuentra en el modulo de “Modificar Usuario”, obtiene los mismos parámetros que en la modificación de un usuario, solo que ahora se realiza un llamado al método “getBorrarUsuario” definido en el bean “ModElimUsuarioBean.java”. En este método se elimina el usuario y toda su información.

Conclusiones

El desarrollo de esta aplicación Web nos proporciona una estructura más clara de la misma reduciendo los costos de mantenimiento y aumentando en gran medida el nivel de reusabilidad de código para la mejora de otros proyectos en un futuro, mostrando así las ventajas del patrón de arquitectura MVC y de la metodología RUP como son la utilización de Casos de Uso, los cuales no son sólo una herramienta para especificar los requisitos del sistema. También guían su diseño, implementación y prueba, ayudando a construir una arquitectura que no se vea fuertemente impactada ante cambios posteriores.

Otra gran ventaja que se propone en RUP es tener un proceso iterativo e incremental en donde el trabajo se divide en partes más pequeñas o mini proyectos. Permitiendo que el equilibrio entre Casos de Uso y arquitectura se vaya logrando durante cada mini proyecto, así durante todo el proceso de desarrollo. RUP se repite a lo largo de una serie de ciclos que constituyen la vida de un producto. Cada ciclo (Inicio, Elaboración, Construcción y Transición) concluye con una generación del producto para los clientes.

El desarrollo de este proyecto pretende mejorar la asesoría jurídica y los servicios del gobierno hacia la población, teniendo así un mayor conocimiento de las leyes para que la población, al aplicar estos conocimientos, alcance mejores condiciones de vida y un mayor crecimiento económico.

Por lo anterior, podemos concluir que los objetivos planteados para este proyecto se cumplieron satisfactoriamente. En esta primera versión, debido a la complejidad del sistema, se presentan algunas limitaciones, las cuales se plantean como posibles trabajos futuros a continuación:

- Crear una aplicación que utilice métodos de búsqueda directa en las leyes almacenadas en archivos.
- Crear una aplicación más robusta que permita el almacenamiento de archivos en la base de datos, permitiendo así, entre otras cosas el borrado en cascada sin temor a que nos queden archivos “colgando”.

La principal finalidad de la realización de esta tesis y, después de los resultados obtenidos, se pretende dar una gama de posibilidades a los alumnos de la Universidad en cuanto al desarrollo de proyectos Web bajo tecnologías como Java y Struts. Asimismo, darles a conocer los grandes beneficios del desarrollo de componentes reutilizables con la metodología RUP para motivarlos en la contribución de nuevos productos.

Bibliografía

[1] UML y patrones: Introducción al análisis y diseño orientado a objetos
Larman, Craig .México: Prentice Hall : Pearson Educación, 1999.

[2] El proceso unificado de desarrollo de Software
Ivar Jacobson, Gradi Booch, James Rumbaugh

[3] Database Design for smarties using uml for data modeling
Robert J. Muller

[4] The Unified Modeling Language User Guide
Booch, Grady. Reading Mass. Addison-Wesley, c1999.

[5] Objects, Components and frameworks with UML.
Alan Cameron Wills.

[6] The Unified Software Development Process
Jacobson, Ivar. Reading, Mass. Addison-Wesley, c1999.

[7] Estructura de datos con Java
John Lewis, Joseph Chase. Pearson, Addison Wesley

[8] Java for the Web with Servlets, JSP, and EJB
Budi Kurniawan. New Riders

[9] Struts
James Turner, Kevin Bedell. SAMS

[10] Utilización de UML en Ingeniería del Software con objetos y componentes
Perdita STEVENS, Rob POOLEY. Addison Wesley(Editorial)

[11] Guía de aprendizaje MySQL
Larry Ullman. Prentice Hall

[12] Jakarta Struts Cookbook
Bill Siggelkow. O'REILLY

[13] Servlets & JSP
Bryan Basham, Kathy Sierra & Bert Bates. O'Reilly

[14] Creación de un portal con PHP y MySQL
Jacobo Pavón Puertas. Alfaomega Ra-Ma

Referencias Web

[15] http://daguilar.evolutionperu.com/?cat=3

[16] http://www.fi.uba.ar/materias/7547/Documento%20tipo%20para%20**proyecto**s%20de%20Desarrollo.rtf

[17] http://www.spinec.org/?m=200605

[10] http://jakarta.apache.org/Struts

[11]Construyendo aplicaciones web con una metodología de diseño orientada a objetos.
Darío Andrés Silva
Bárbara Mercerat