UNIVERSIDAD NACIONAL DE LOJA

Área de la Energía, las Industrias y los Recursos Naturales No Renovables.

Carrera de Ingeniería en Sistemas

TEMA:

Construcción de un Framework para desarrollo de aplicaciones Web 2.0 con acceso a base de datos MySql

"Tesis previa a la Obtención del título en Ingeniero en Sistemas".

Autores:

Oscar Rafael Guamán Gutiérrez

Lady Jackelinne Rosillo Cumbicus

Director:

Ing. Milton Leonardo Labanda Jaramillo

LOJA – ECUADOR

2012

CERTIFICACION DEL DIRECTOR

Ing. Milton Leonardo Labanda Jaramillo

Director de Tesis

CERTIFICA:

Que el presente trabajo de Tesis de los Egresados Oscar Rafael Guamán Gutiérrez y Lady Jackelinne Rosillo Cumbicus, titulado "**Construcción de un Framework para desarrollo de aplicaciones Web 2.0 con acceso a base de datos MySql**", ha sido dirigido, asesorado supervisado y realizado bajo mi dirección en todo su desarrollo, y aprobado para su sustentación.

Ing. Milton Leonardo Labanda Jaramillo

Director de Tesis

AUTORÍA

Los autores certifican que los criterios y opiniones vertidas en el presente trabajo de investigación, métodos y procedimientos utilizados en la información, análisis e interpretación de resultados son de exclusiva responsabilidad de los mismos.

AGRADECIMIENTO

A Dios por ser el arquitecto de nuestras vidas, por sustentarnos en cada paso que hemos dado a lo largo de la vida.

Al director de Tesis Ing. Milton Labanda Jaramillo por su asesoría, ayuda y paciencia durante la realización del trabajo de tesis.

Al personal del Área de Recursos Naturales No Renovables y de manera especial al coordinador de carrera Ing. Edison Coronel por su gran apoyo y compromiso con su trabajo.

A todas aquellas personas que nos han acompañado durante estos años, creyeron en nosotros, tanto profesionales como estudiantes; que de alguna forma nos incentivaron hacer posible este trabajo que significa esfuerzo, entrega, compromiso y dedicación en contribución con nuestra sociedad.

Gracias a todos.

DEDICATORIA

Dedico el presente Proyecto

A Dios, por ser mi luz y mi camino; por darme las fuerzas necesarias en los momentos en que más las necesité y bendecirme con la posibilidad de caminar a su lado durante toda mi vida.

A mis queridos padres, José y Rosa por todo lo que me han dado en esta vida, especialmente por su gran amor, sus sabios consejos y por estar a mi lado en los momentos difíciles.

A mis hermanos Sandra y Joselo por su apoyo incondicional siempre en mi vida, mis cómplices.

A mis amores Cristian y Karen que fueron parte fundamental para la culminación de este proyecto, por su amor y paciencia.

Gracias.

Lady.

Dedico este Proyecto

A Dios por ser la fuente de inspiración para poder realizar y terminar este trabajo, por estar siempre presente en todos los momentos difíciles del diario vivir.

A mi Madre Lilia por el apoyo y el esfuerzo que ha realizado para sacarme adelante en mis estudios y en mi carrera profesional.

A todos los docentes de la carrera que me han ayudado con sus conocimientos para poder llegar a cumplir esta meta.

A mis familiares, amigos y demás allegados que siempre han estado presentes con su motivación para que culmine este proyecto.

Gracias.

Oscar

CESIÓN DE DERECHOS

Los autores Oscar Rafael Guamán Gutiérrez y Lady Jackelinne Rosillo Cumbicus egresados de la carrera de Ingeniería en Sistemas declaramos conocer que nuestro trabajo de investigación forma parte del patrimonio de la Universidad Nacional de Loja, por lo tanto autorizamos hacer uso pertinente del mismo, cediendo todos sus derechos de autoría a la Universidad Nacional de Loja.

Egdo. Oscar Rafael Guamán Gutiérrez
C.I: 1103643134

Egda. Lady Jackelinne Rosillo Cumbicus
C.I: 1104113848

RESUMEN

El siguiente trabajo de tesis se basa en la construcción de un Framework para el desarrollo de aplicaciones web 2.0 con acceso a bases de datos MySql.

En el primer capítulo trata de forma general lo que son las aplicaciones web, como van a estar estructuradas cada una de las aplicaciones web desarrolladas con la herramienta y cuál sería su formulario de inicio que se visualizara en el navegador.

En el segundo capítulo se describe lo que es la web 2.0, como es su funcionamiento y sobre cada una de las tecnologías que intervienen para la creación de las aplicaciones web desarrolladas en el Framework, como lo son **XML**: se crean dos archivos iniciales el primer archivo utilizamos para guardar los datos de cada uno de los componentes que se crean dentro de la herramienta y el otro archivo guarda las propiedades de los componentes de entrada/salida de datos y a la vez los datos de la conexión a la base de datos como es el nombre de la base de datos, usuario y contraseña del motor de base de datos MySql, **XSL**: son plantillas que utilizamos respectivamente con los archivos xml; el primero para almacenar cada componente con los atributos estándares de cada uno de ellos y la segunda plantilla almacena la codificación de un archivo java con la generación de los parámetros de los componentes de entrada/salida del formulario, **XHTML**: que es el resultado de la transformación de XML y XSL, **JavaScript**: en la cual se programan las funciones para darle dinamismo a las aplicaciones, **CSS**: que se lo ha utiliza para dar un mejor estilo de presentación a las aplicaciones, **Objeto XMLHttpRequest**: que es la parte de Ajax de las aplicaciones web, y **Java:** que es el resultado de la transformación de XML y XSL, clase servlet que contiene la conexión a la base de datos y la generación de los parámetros utilizados en el formulario. Además trata de algunas razones importantes de por qué se debe utilizar la web 2.0.

En el capítulo tres especificamos las herramientas de desarrollo como son el IDE NetBeans que se lo utilizó para la construcción de la interfaz gráfica, JDBC que se utiliza para realizar las conexiones a la base de datos con java; las librerías: JDOM el mismo que fue utilizado para el parseo de los datos, SEVLETS que se la utiliza en lado del servidor para enviar los datos, QUAQUA una librería para cambiarle la apariencia a la interfaz gráfica, y por ultimo TOMCAT el mismo que nos sirve como servidor para poder visualizar nuestras aplicaciones realizadas con el Framework TOOLWEB.

SUMARY

The following theis research it`s based on Framework construction for the developing of applications web 2.0 with acces to the database MySql.

Chapter number one is about general ideas what the are web applications,, structure, how are developed with the tool and what is it`s begin form that you can observe in the web.

Chapter number two describe what is web 2.0, its operations and each techbology that interviewing for the creation of web applications in Framework like **XML**, there are two initials registers, the number one we use to save the components that are in the tool and the number two save the copyrights of the components of enter and exit of dates at the same time the connection dates of database like usurious and key of the database motor MySql, **XSL:** are templates, we use the register xml to store up one by one the components an the other template sore up the codification of a java register with the parameter of the components of enter and exit, **XHTML** it`s the result of the transformations of XML and XSL, **JavaScript:** in which we program the functions to give dynamism to the applications, **CSS** its used to give a better style of presentation to the applications, **Object XMLHttpRequest** it is the Ajax part of the web applications and **Java** that is the result of the transformation of XML and XSL, servlet that contains the database connection and the generation of the parameters used in the formulary. Also this chapter is about some important reasons of why people must use web 2.0

In chapter number three we specify the developing tools like IDE NetBeans that were used for the construction o graph interface, **JDBC** that is used to realize the connections of the database with java; **JDOM** is used to the SERVLETS it is used to send dates, **QUAQUA** it is used to change the image to the graph interface and finally; **TOMCAT** it serves for visualizing our applications realized with the Framework TOOLWEB

INDICE

INDICE GENERAL

CONTENIDOS: **PAG.**

CONTENIDOS: **Pág.**

CONTENIDOS: **PAG.**

INDICE DE FIGURAS

CONTENIDOS: **Pág.**

CONTENIDOS: **Pág.**

CONTENIDOS: **Pág.**

CONTENIDOS: **Pág.**

INDICE DE TABLAS

CONTENIDOS: **Pág.**

A. INTRODUCCION.

Con la llegada de Internet han surgido las aplicaciones Web. Este tipo de aplicaciones permiten usar la infraestructura de la Web para desarrollar aplicaciones globales.

Reportes recientes indican que las aplicaciones Web representan más de la mitad del total de todas las aplicaciones de la industria de software. Esa cifra indica que las aplicaciones Web siguen creciendo y ganando popularidad en un mercado muy competitivo, y que se proyectan a ser las aplicaciones del futuro.

El presente proyecto de tesis es el resultado de la investigación sobre los avances tecnológicos que se han surgido en los últimos años.

El proyecto titulado "Construcción de un Framework para desarrollo de aplicaciones Web 2.0 con acceso a base de datos MySql", al cual lo denominamos Framework **"TOOLWEB",** herramienta que permite crear las aplicaciones web de una forma dinámica y rápida para los usuarios, ya que cuenta con la Web 2.0 que es la tecnología que se utiliza hoy en día.

La herramienta fue desarrollada en el lenguaje de programación JAVA el mismo que es multiplataforma. **TOOLWEB** se centra en la creación de aplicaciones web con la generación de código fuente de los archivos principales como son: XML, XSL, JavaScript, HTML y Java. Con la finalidad de brindar al usuario facilidad de creación de las aplicaciones y bases de datos MySql; dichas aplicaciones se ejecutaran con el servidor TOMCAT.

B. METODOLOGIA.

Para la elaboración del Framework fue necesario seguir con un esquema metodológico, basado en el uso de métodos y técnicas las cuales nos facilitaron la recolección de información y se convirtieron en guías para desarrollar la presente investigación.

1. Métodos.

1.1 Método Deductivo.

Este método lo utilizamos partiendo desde el enfoque de cómo realizar una aplicación web de tal manera que hemos investigado cada uno de las partes que intervienen en dichas aplicaciones, realizando el análisis necesario de cada una de estas, llegando a determinar que es factible la construcción del Framework TOOLWEB.

1.2 Método Analítico.

El método Analítico se lo utiliza para el análisis e interpretación de la información obtenida en el transcurso de nuestro proyecto de investigación, este método lo implementamos en la determinación de la funcionalidad de cada parte que intervienen en la estructura del Framework y de las aplicaciones web desarrolladas, las mismas que están integradas por varias tecnologías resultado del análisis de la información obtenida acerca de la tecnología Web 2.0. También se aplica el método analítico para el desarrollo del marco teórico, introducción del proyecto.

1.3 Método Descriptivo.

El método descriptivo consiste en hacer una identificación del problema, la recolección de datos, como la extracción de conclusiones razón por la cual hemos creído conveniente utilizar este método, el mismo que lo aplicamos en la elaboración del tema y resultados del objeto de la investigación.

1.4 Método Sintético.

Este método conjuntamente con el método analítico se los utiliza para la interpretación de la información; se empleó este método para determinar las conclusiones y recomendaciones de nuestro proyecto, los mismos que son parte fundamental en la síntesis de esta investigación.

1.5 Metodología para el desarrollo del Software.

1.5.1 Metodología ICONIX.

ICONIX es un proceso simplificado en comparación con otros procesos más tradicionales, que unifica un conjunto de métodos de orientación a objetos con el objetivo de abarcar todo el ciclo de vida de un proyecto. Además Iconix está adaptado a los patrones y ofrece el soporte de UML.[1]

Esta metodología consta de las siguientes tareas:

Análisis de Requisitos

Dentro de esta fase determinamos el modelo del Domino en el que se encuentran las clases iniciales principales del proyecto, luego se hizo el prototipo de la pantalla inicial del Framework, posteriormente se identifican el modelo de caso de usos que permiten estructurar y articular los requerimientos.

Análisis y Diseño Preliminar

En la segunda tarea de la metodología ICONIX se realizó la descripción de cada uno de los casos de uso, los cuales describen las acciones y reacciones que permiten definir los límites del sistema y las relaciones con el entorno. Se empieza a definir el diagrama de clases partiendo del modelo del dominio.

Diseño

Se desarrollan los diagramas de secuencia que describen los cursos alternos que pueden tomar cada uno de los casos de uso especificando el comportamiento de

[1] UML: Lenguaje Unificado de Modelado, es el lenguaje de modelado de sistemas de software.

interacción del Framework con el usuario. Además adicionamos los detalles del diseño en nuestro Diagrama de clases.

En esta fase queda verificado el diseño con los requerimientos identificados.

Implementación

Dentro de esta fase se realizaron las pruebas respectivas, recodificando las sugerencias por parte de los usuarios, logrando así obtener una acogida aceptable para la utilización del mismo.

2 Técnicas Aplicadas.

2.1 Encuesta

Cuestionario prediseñado dirigido a un grupo de personas con la finalidad de obtener datos.

Esta técnica la seleccionamos porque a través de la encuesta nos permite recolectar criterios de los usuarios que tienen mayor conocimiento sobre la web, los mismos que son parte fundamental en la técnica aplicada.

Este proceso se lo realizo a estudiantes de sexto y octavo módulos de la carrera de Ingeniería en Sistemas del Área de la Energía, las Industrias y los Recursos Naturales no Renovables, los cuales nos proporcionaron la información necesaria para determinar si el Framework TOOLWEB tuvo la funcionalidad y acogida aceptable.

2.2 Observación Directa

Esta técnica se la aplico para realizar el sondeo sobre la aceptación de la herramienta TOOLWEB y las sugerencias para el mejoramiento de la misma; esto se realizó en el aula virtual con la colaboración de los estudiantes de sextos y octavos módulos con sus correspondientes docentes Ing. Hernán Torres y él Ing. Wilman Chamba de la carrera de Ingeniería en Sistemas.

C. REVISION DE LITERATURA O MARCO TEORICO.

APLICACIONES WEB.

1.1 ¿Qué es una Aplicación Web?

En inglés se denomina “browser-based aplication”, es decir, aplicación basada en navegadores. Son programas que se diseñan para funcionar a través de un navegador de internet, es decir, son aplicaciones que se ejecutan de forma online. [2]

- **Ventajas**: proporcionan movilidad, debido a que se pueden ejecutar desde cualquier ordenador con conexión a internet. La información que manejan se accede a través de internet, motivo por el cual son especialmente interesantes para desarrollar aplicaciones multiusuario basadas en la compartición de información. El cliente o usuario que utiliza la aplicación no necesita tener un ordenador con tecnología avanzada para trabajar con ella.

- **Desventajas**: la comunicación constante con el servidor que ejecuta la aplicación establece una dependencia con una buena conexión a internet. Además, el servidor debe tener las características necesarias para ejecutar la aplicación de manera fluida, no sólo para un usuario sino para todos los que la utilicen de forma concurrente.

1.2 Estructura de una Aplicación Web.

Una APLICACIÓN WEB en Java consiste en un conjunto de servlets, ficheros html, clases Java de apoyo empaquetadas o no en ficheros jar y otro tipo de recursos tales como ficheros de imágenes, de sonidos, de texto, etc.[3]

Una aplicación web puede existir de dos modos:

[2] Jose María de Pereda (JMPereda), ¿Que es una Aplicacion Web?, [internet], agosto 24, 2007, <http://jmpereda.wordpress.com/2007/08/24/definiendo-la-plantilla/>, [Fecha de Consulta: 15 octubre 2008].

[3] ADR Infor S.L., Estructura y despliegue de aplicaciones web, [internet], 2004, < http://www.adrformacion.com/curso/javaservidor/leccion3/estructura_servidor_web.htm>, [Fecha de consulta 10 noviembre 2009].

- **Mediante un fichero de extensión war (Web Application Resource, a veces también se le suele llamar Web Archive)** que engloba a todo su contenido. Se crea mediante la herramienta jar incluido en el J2SE, del mismo modo que el fichero jar. El empaquetamiento se produce en la etapa de producción, es decir, cuando la aplicación ha sido comprobada y depurada para su comercialización.

- **Mediante una estructura de directorios basada en la especificación definida por Sun para los Servlets.** Dentro de esta estructura deben ubicarse de forma adecuada los componentes de la aplicación.

Es el modo de trabajo habitual en la etapa de desarrollo de la aplicación, es decir, cuando se realizan pruebas y modificaciones constantes en sus componentes.

Con estos modos de existencia se persigue que la misma aplicación web pueda ser desplegada en diferentes servidores web manteniendo su funcionalidad y sin ninguna modificación de código.

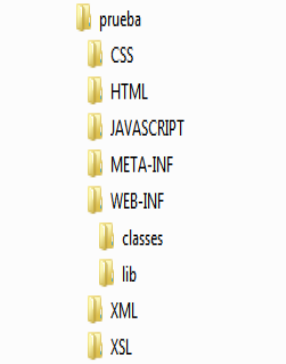


Figura 1: Estructura de la aplicación web

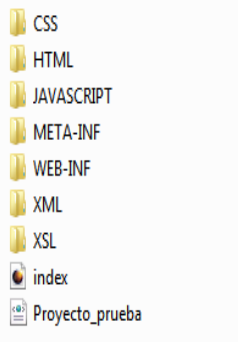


Figura 2: Directorios de la aplicación web

1.3 Formularios Web.

Un formulario web es un tipo de formulario que es presentado en un navegador y puede ser rellenado a través de una red como internet. Generalmente cuando se ingresan los datos, se envían a un servidor web para ser procesados.[4]

Los formularios web pueden ser usados para suscripciones, encuestas, elección de opciones, enviar palabras para los buscadores, etc.

[4] ALEGSA, Definicion de Formulario web - ¿qué es Formulario web?,[internet], < http://www.alegsa.com.ar/Dic/formulario%20web.php>, [Fecha de consulta octubre 2008].

Los formularios web generalmente son hechos en HTML y en estos pueden usarse los siguientes elementos:

- Campo de texto (input field)
- Campo oculto (hide field)
- Área de texto (textarea)
- Casilla de verificación (checkbox)
- Botón de opción (radio button)
- Grupo de opción
- Lista o Menú (con opción a Menú de salto)
- Botón (submit)

Estos elementos son los más comunes en una interfaz GUI.

Los formularios pueden ser combinados con lenguajes de programación/script tanto del lado del cliente como del lado del servidor.

Del lado del cliente el estándar de factor es JavaScript, y se utiliza en los formularios especialmente en la validación de datos. Estos códigos en JavaScript son ejecutados en el navegador del usuario.

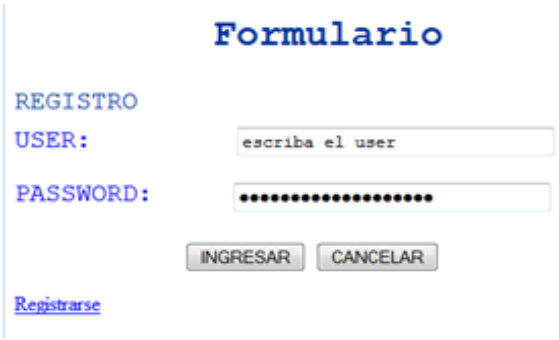


Figura 3: Ejemplo de formulario

LA WEB 2.0

2.1 Generalidades.

La Web 2.0 es la representación de la evolución de las aplicaciones tradicionales hacia aplicaciones web enfocadas al usuario final. El Web 2.0 es una actitud y no precisamente una tecnología. [5]

La Web 2.0 es la transición que se ha dado de aplicaciones tradicionales hacia aplicaciones que funcionan a través de las web enfocadas al usuario final. Se trata de aplicaciones que generen colaboración y de servicios que remplacen las aplicaciones de escritorio.

2.2 Ajax.

2.2.1 Definición

AJAX, esta no es otra cosa que las siglas de la palabra inglesa Asynchronicus JavaScript and XML, es decir, algo así como JavaScript asincrónico y XML. Como su nombre lo indica, AJAX es una técnica de desarrollo web que incluye una variedad de tecnologías diferentes, que ofrecen a los desarrolladores la posibilidad de crear aplicaciones web interactivas, que funcionan como herramientas cliente/servidor.[6]

2.2.2 El Funcionamiento de Ajax

El funcionamiento de las aplicaciones AJAX, es bastante simple: Las tareas de programación se ejecutan directamente en el navegador web, es decir, en el lado del cliente. Durante la ejecución, la aplicación mantiene una comunicación en segundo plano con el servidor (comunicación asincrónica), de manera que se pueden realizar cambios en la aplicación, sin necesidad de realizar una recarga de la aplicación web.

[5] Christian Van Der Henst S. para Maestros del web, ¿Qué es la web 2.0?, [internet], Noviembre 2005, http://www.maestrosdelweb.com/editorial/web2/, [fecha de consulta agosto 2008].
[6] Bulmaro Noguera, Qué es AJAX, [internet], 5 de julio 2011, http://culturacion.com/2011/07/que-es-ajax/, [fecha de consulta: noviembre del 2011].

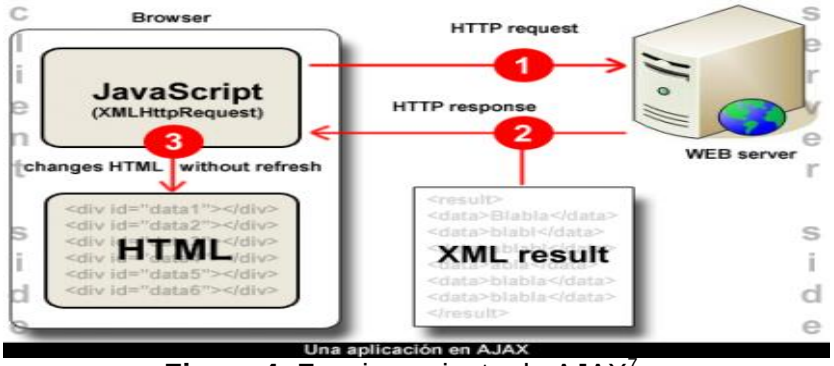


Figura 4: Funcionamiento de AJAX[7]

De esta manera, podemos realizar aplicaciones web completamente dinámicas y con una velocidad de ejecución impresionante. AJAX utiliza JavaScript para crear acciones interactivas en la pantalla del navegador, y XML para tener un acceso envidiable de los datos que se van a transmitir a través de la red. Otra de las características de AJAX, es que utiliza el concepto de DOM, el cual es muy empleado en las nuevas tendencias de programación web.

2.3. Tecnologías que se Utilizan para Realizar las Aplicaciones con el Framework TOOLWEB.

2.3.1 XML

XML es un sistema estándar de codificación de información. Los programas que utilizan el formato XML pueden intercambiar fácilmente sus datos, ya que responden a una misma lógica interna.

Los documentos XML son ficheros de texto que contienen la información organizada en forma de árbol: cada rama puede tener unos atributos propios y servir de base para otras ramas. Además, los documentos XML se pueden transformar (por ejemplo, a formato HTML, para mostrar la información en una página web), o combinar: un tronco con todas sus ramas puede pasar a ser una rama de otro árbol mayor.[8]

[7] Bulmaro Noguera, Qué es AJAX, [internet], 5 de julio 2011, http://culturacion.com/2011/07/que-es-ajax/, [fecha de consulta: noviembre del 2011].
[8] ZonaClic, ¿Qué es XML?, [internet], http://clic.xtec.cat/es/jclic/xml.htm, [fecha de consulta: agosto del 2010].

Ejemplo:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl"href="XSL\Registro.xsl"?>
<componentes-formulario>
<accion />
<metodo />
<nombre />
    <etiqueta>
  <texto>REGISTRO</texto>
</etiqueta>
<saltolinea>true</saltolinea>
<etiqueta>
        <texto>Usuario:</texto>
</etiqueta>
<campotexto>
        <nombre>user</nombre>
        <tipo>text</tipo>
        <valor>Escriba su nombre</valor>
        <tamano>30</tamano>
        <maxlongitud>50</maxlongitud>
</campotexto>
<saltolinea>true</saltolinea>
<etiqueta>
        <texto>Contrase&ñtilde;a:</texto>
</etiqueta>
<campotexto>
        <nombre>password</nombre>
        <tipo>text</tipo>
        <valor>escribir password</valor>
```

```
            <tamano>30</tamano>
            <maxlongitud>30</maxlongitud>
        </campotexto>
        <saltolinea>true</saltolinea>
        <enviarboton>
            <nombre>botoningresar</nombre>
            <tipo>submit</tipo>
            <valor>INGRESAR</valor>
        </enviarboton>
        <boton>
            <nombre>botoncancelar</nombre>
            <tipo>button</tipo>
            <valor>CANCELAR</valor>
        </boton>
        <saltolinea>true</saltolinea>
        <hypervinculo>
         <texto>REGISTRARSE</texto>
            <enlace>jTextField1</enlace>
        </hypervinculo>
      </componentes-formulario>
```

2.3.2 XSL

XSL es para XML lo que CSS es para HTML. Es un acrónimo en inglés de Extensible Stylesheet Language (Lenguaje de hojas de estilo ampliable). Es un lenguaje diseñado para presentar datos XML en un formato legible. XSL consta realmente de dos partes:

- XSLT: un lenguaje para transformar documentos XML
- XPath: un lenguaje para navegar en documentos XML

XSLT significa Transformaciones XSL y es la parte más importante de XSL.

XSLT transforma un documento XML en otro documento XML, en una salida XHTML o en texto sencillo. Esto se suele hacer transformando cada elemento XML en un elemento HTML. El uso de XSL es imprescindible, ya que las etiquetas XML han sido definidas por el usuario y, por tanto, los navegadores no saben cómo interpretarlas o representarlas. Su significado se ha diseñado para ser entendido por las personas, no por las máquinas.[9]

XSLT también puede realizar las siguientes operaciones en un árbol XML:

- añadir y eliminar elementos
- añadir y eliminar atributos
- reorganizar y ordenar elementos
- ocultar o mostrar determinados elementos
- encontrar o seleccionar elementos específicos

Ejemplo:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="1.0">
 <xsl:output method="html" />
 <xsl:template match="/">
  <html>
   <head>
    <title>Pagina Principal</title>
   </head>
   <body>
    <h2>
     <center>Formulario</center>
    </h2>
    <form id="formu" action="{accion}" method="post">
     <xsl:apply-templates />
    </form>
   </body>
  </html>
 </xsl:template>
 <xsl:template match="boton">
  <input id="{identificador}" name="{nombre}" type="{tipo}"
value="{valor}" />
 </xsl:template>
 <xsl:template match="enviarboton">      <input id="{identificador}"
name="{nombre}" type="{tipo}" value="{valor}"  />
 </xsl:template>
```

[9] Marius Zaharia, Introduccion a XSL/centro de desarrolladores, 22 de agosto 2005, http://www.adobe.com/es/devnet/dreamweaver/articles/xsl_overview.html, [fecha de consulta: agosto 2010].

```
 <xsl:template match="hypervinculo">
  <a href="{enlace}" id="{identificador}">
   <xsl:value-of select="texto" />
  </a>
 </xsl:template>
 <xsl:template match="etiqueta">
  <label id="{identificador}">
   <xsl:value-of select="texto" />
  </label>
 </xsl:template>
 <xsl:template match="contrasena">
    <xsl:choose>
     <xsl:when test="soloLectura = 'true'">
        <input id="{identificador}" name="{nombre}" type="{tipo}"
value="{valor}" size="{tamano}" maxlength="{maxlongitud}"
readonly="{soloLectura}"/>
      </xsl:when>
      <xsl:when test="desactivar = 'true'">
        <input id="{identificador}" name="{nombre}" type="{tipo}"
value="{valor}" size="{tamano}" maxlength="{maxlongitud}"
disabled="{desactivar}"/>
      </xsl:when>
      <xsl:otherwise>
        <input id="{identificador}" name="{nombre}" type="{tipo}"
value="{valor}" size="{tamano}" maxlength="{maxlongitud}"/>
      </xsl:otherwise>
     </xsl:choose>
 </xsl:template>
 <xsl:template match="saltolinea">
  <br />
 </xsl:template>
 <xsl:template match="campotexto">
    <xsl:choose>
     <xsl:when test="soloLectura = 'true'">
        <input id="{identificador}" name="{nombre}" type="{tipo}"
value="{valor}" size="{tamano}" maxlength="{maxlongitud}"
readonly="{soloLectura}"/>
      </xsl:when>
      <xsl:when test="desactivar = 'true'">
        <input id="{identificador}" name="{nombre}" type="{tipo}"
value="{valor}" size="{tamano}" maxlength="{maxlongitud}"
disabled="{desactivar}"/>
      </xsl:when>
      <xsl:otherwise>
        <input id="{identificador}" name="{nombre}" type="{tipo}"
value="{valor}" size="{tamano}" maxlength="{maxlongitud}"/>
      </xsl:otherwise>
     </xsl:choose>
  </xsl:template>
          </xsl:stylesheet>
```

2.3.3 JavaScript

JavaScript es un lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas.

Una página web dinámica es aquella que incorpora efectos como texto que aparece y desaparece, animaciones, acciones que se activan al pulsar botones y ventanas con mensajes de aviso al usuario.

Técnicamente, JavaScript es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos. En otras palabras, los programas escritos con JavaScript se pueden probar directamente en cualquier navegador sin necesidad de procesos intermedios.[10]

```
function cargarMensaje() {

        ajax();
        formu.action = "/Registro/Conexion";
        formu.submit();

}

function redireccionar(){

        formu.action = "/Registro/Ingresar";
        formu.submit();

}

function verificarDatos(){

      formu.action = "/Registro/ValidarCampos";
      formu.submit();

}
```

2.3.4 CSS

Hojas de Estilo en Cascada (Cascading Style Sheets), es un mecanismo simple que describe cómo se va a mostrar un documento en la pantalla, o cómo se va a imprimir, o incluso cómo va a ser pronunciada la información presente en ese documento a

[10] Librosweb.es, Introduccion a JavaScript, http://www.librosweb.es/javascript/capitulo1.html, [fecha de consulta: agosto 2010].

través de un dispositivo de lectura. Esta forma de descripción de estilos ofrece a los desarrolladores el control total sobre estilo y formato de sus documentos[11].

Ejemplo:

```
/* CSS Document */
<style type="text/css">
BODY{
   color: #003399;
   background-color: #d8da3d;
}
h2{
    font-family: "Courier New", Courier, monospace;
    color: #003399;
   font-size:30px;
}
.titulo{
    font-family: "Courier New", Courier, monospace;
    color: #003399;
   font-size:20px;
    text-align:center;
}
</style>
```

2.3.5 XHTML

Acrónimo en inglés de eXtensible Hypertext Markup Language. Es el lenguaje marcado nuevo pensado para sustituir al HTML como estándar para las páginas web.

[11] W3C.es, Guia Breve de CSS, http://www.w3c.es/Divulgacion/GuiasBreves/HojasEstilo, [fecha de consulta: agosto 2010].

XHTML es la interpretación XML de HTML por lo que tiene básicamente las mismas funcionalidades pero cumple las especificaciones más estrictas de XML.[12]

Ejemplo:

```
<html>
  <head>
    <title>Pagina Principal</title>
  </head>
  <body>
    <h2>
      <center>Formulario</center>
    </h2>
    <form action="" method="" name="">
      <label>DATOS PERSONALES:</label>
      <br />
      <label>NOMBRES:</label>
      <input name="nombres" type="text" value="Escriba su nombre" size="30" maxlength="50" />
      <br />
      <label>APELLIDOS:</label>
      <input name="apellidos" type="text" value="escriba sus apellidos" size="30" maxlength="50" />
      <br />
      <label>DIRECCION:</label>
      <input name="direccion" type="text" value="escriba su direccion" size="30" maxlength="50" />
      <br />
      <label>Usuario:</label>
      <input name="user" type="text" value="escriba usuario" size="30" maxlength="50" />
```

[12] Hooping.net, glossary/xhtml, 2008, http://www.hooping.net/glossary/xhtml-137.aspx, [fecha de consulta_ agosto 2010].

```
        <br />
        <label>Contrasena</label>
        <input name="password" type="password" value="escriba
    contrase&ntilde;a" size="30" maxlength="50" />
        <br />
        <input name="guardarDatos" type="submit" value="Guardar" />
        <input name="cancelarDatos" type="button" value="Cancelar" />
      </form>
    </body>
  </html>
```

2.3.6 Objeto XMLHttpRequest.

XMLHttpRequest es el objeto primario que nos permitirá la comunicación con el servidor de forma asíncrona, originalmente fue implementada para *Internet Explorer* y basada en un **ActiveX** y ahora los otros navegadores también la soportan.[13]

Lo primero es detectar el tipo de navegador y crear una instancia del objeto XMLHttpRequest para poder utilizarlo posteriormente, para ello implementamos la siguiente función en *JavaScript*.

```
function ajax(){

var xmlhttprequest = false;

      if (window.XMLHttpRequest) {
    // Si es Mozilla, Safari etc
    xmlhttprequest = new XMLHttpRequest ();

  } else if (window.ActiveXObject) {
    // pero si es IE
    try{
      xmlhttprequest = new ActiveXObject ("Msxml2.XMLHTTP");
    }catch (e) {
      // en caso que sea una versión antigua
      try {
        xmlhttprequest = new ActiveXObject ("Microsoft.XMLHTTP");
```

[13] Unijimpe, Introduccion a AJAX, 25 diciembre 2007, http://blog.unijimpe.net/introduccion-a-ajax/, [fecha de consulta: agosto 2010].

```
            }
            catch (e)
            {
            }
        }
    }
return xmlhttprequest;
}
```

2.3.7 JAVA.

Son clases denominadas **OB.java** que extienden de la clase **HttpServlet** las mismas que nos permitirán acceder a la base de Datos. Estas son el resultado de la transformación de una plantilla llamada **FormularioObjeto.xsl** la misma que se la utiliza para todas las aplicaciones, Un archivo **OB.xml** que contiene las propiedades de entrada y salida de datos.

RegistroOB.java

```
/*
 *  Clase de los objetos del Formulario.
 */
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.*;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class RegistroOB extends HttpServlet {
   PrintWriter pagina;
   Connection canal = null;
   ResultSet tabla= null;
   Statement instruccion;

   public void init(ServletConfig conf) throws ServletException {
       super.init(conf);
   }

   public void doGet(HttpServletRequest request, HttpServletResponse
response)     throws ServletException, IOException{
       processRequest(request, response);
   }
```

```
    public void doPost(HttpServletRequest request, HttpServletResponse
response)     throws ServletException, IOException{
        processRequest(request, response);
    }

    public void processRequest(HttpServletRequest request,
HttpServletResponse response)     throws ServletException, IOException{
        response.setContentType("text/html");
        pagina = response.getWriter();

        //Conexion a la base de datos
        String nombreBD = "registro";
        String usuarioBD = "root";
        String contrasenaBD = "";
        String servidor = "jdbc:mysql://localhost:3306/".concat(nombreBD);
        try { Class.forName("com.mysql.jdbc.Driver");
            canal =  DriverManager.getConnection(servidor, usuarioBD,
contrasenaBD);
            instruccion =   (Statement) canal.createStatement();
        } catch(java.lang.ClassNotFoundException e){} catch(SQLException e) {};
        // Elementos a cargar en la base de datos.
        String user = request.getParameter("USER");
        String password = request.getParameter("PASSWORD");
        // Sentencias para la base de datos:
        pagina.close();
    }
}
```

FormularioObjetos.xsl

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
version="1.0">
  <xsl:output method="text"></xsl:output>
  <xsl:variable name="className"
select="/formularioObjetos/@clase"></xsl:variable>
  <!--  ********************************************************************************
  **   Generar la clase servelt con los  objetos  del formulario
  **   que se utilizan para realizar las operaciones en la BD.
  ********************************************************************************
-->
  <xsl:template match="/formularioObjetos">
    /*
     *  Clase de los objetos del Formulario.
     */
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.*;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
```

```
import java.sql.SQLException;
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
   <xsl:value-of select="concat('public class ', $className, ' extends
HttpServlet {')"></xsl:value-of>
   PrintWriter pagina;
   Connection canal = null;
   ResultSet tabla= null;
   Statement instruccion;

   public void init(ServletConfig conf) throws ServletException {
       super.init(conf);
   }

   public void doGet(HttpServletRequest request, HttpServletResponse
response)     throws ServletException, IOException{
       processRequest(request, response);
   }

   public void doPost(HttpServletRequest request, HttpServletResponse
response)     throws ServletException, IOException{
       processRequest(request, response);
   }

   public void processRequest(HttpServletRequest request,
HttpServletResponse response)     throws ServletException, IOException{
       response.setContentType("text/html");
       pagina = response.getWriter();

        //Conexion a la base de datos
   <xsl:value-of select="concat('String nombreBD = &amp;quot;', @nombreBD,
'&amp;quot;;&amp;#xA;')"></xsl:value-of>
   <xsl:value-of select="concat('String usuarioBD = &amp;quot;', @usuarioBD,
'&amp;quot;;&amp;#xA;')"></xsl:value-of>
   <xsl:value-of select="concat('String contrasenaBD = &amp;quot;',
@contrasenaBD, '&amp;quot;;&amp;#xA;')"></xsl:value-of>
   String servidor = "jdbc:mysql://localhost:3306/".concat(nombreBD);
   try { Class.forName("com.mysql.jdbc.Driver");
      canal =  DriverManager.getConnection(servidor, usuarioBD,
contrasenaBD);
       instruccion =   (Statement) canal.createStatement();
   } catch(java.lang.ClassNotFoundException e){} catch(SQLException e) {};

// Elementos a cargar en la base de datos.
   <xsl:apply-templates select="propiedad"
mode="generateRequest"></xsl:apply-templates>
   <xsl:text>// Sentencias para la base de datos:

       pagina.close();
```

```
    }
  }</xsl:text>
   </xsl:template>
   <!--  ************************************************************************
   **     Generar los objetos del formulario
   **     con su correspondiente variable.
   ************************************************************************
  -->
   <xsl:template match="propiedad" mode="generateRequest">
    <xsl:value-of select="concat('    ', @tipo, ' ')"></xsl:value-of>
    <xsl:value-of select="concat(' ', @nombre, ' = ')"></xsl:value-of>
    <xsl:text>request.getParameter("</xsl:text>
    <xsl:value-of select="concat(@nombre, '&amp;quot;);&amp;#xA;')"></xsl:value-of>
   </xsl:template>
</xsl:stylesheet>
```

RegistroOB.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<formularioObjetos clase="RegistroOB" nombreBD="registro" usuarioBD="root"
contrasenaBD="">
      <propiedad nombre="USER" tipo="String" />
      <propiedad nombre="PASSWORD" tipo="String" />
</formularioObjetos>
```

2.4 Razones para utilizar Ajax.

Existen algunas razones para utilizar AJAX[14]

2.4.1 Basado en los estándares abiertos.

Ajax está formado por las tecnologías JavaScript, html, xml, css, y XMLHTTPRequest Object, siendo este último el único que “no es” estándar pero es soportado por los navegadores más utilizados de internet como son los basados en Mozilla, Internet Explorer, Safari y Opera.

2.4.2 Usabilidad.

Permite a las páginas hacer una pequeña petición de datos al servidor y recibirla sin necesidad de cargarla página entera. El incremento de las actualizaciones “on the fly” elimina el tener que refrescar el navegador, algo bastante apreciado a la hora de operar en una aplicación web.

[14] Jorge A. Saavedra Gutierre, Razones para usar AJAX/El mundo informatico, 22 junio 2007, http://jorgesaavedra.wordpress.com/2007/06/22/10-razones-para-usar-ajax/, [fecha de consulta: agosto 2010].

2.4.3 Valido en cualquier plataforma y navegador.

Internet Explorer, los basados en Mozilla y Firefox son los que se llevan la palma en el mercado de internet y además son los navegadores en los que es más fácil programar aplicaciones Web AJAX, pero ahora es posible construir aplicaciones web basadas en AJAX para que funcionen en los navegadores más modernos. Es una de las razonas más importantes por las que AJAX se ha vuelto tan popular. Aunque si bien muchos desarrolladores sabían que era posible usarse años atrás con Internet Explorer, no era viable realizarse. Ahora ya es posible su avance gracias a Mozilla y Firefox.

2.4.4 Beneficia las aplicaciones web

AJAX es la cara del presente en las aplicaciones web – las aplicaciones web conllevan ciertos beneficios sobre las aplicaciones sobre escritorio (aplicaciones que dependan de un sistema operativo, librerías, lo que entendemos por programas compilados).

Esto incluyo un menor coste de creación, facilidad de soporte y mantenimiento, menores tiempos a la hora de desarrollarlas, y sin necesidad de instalaciones; éstas son algunas de los beneficios que han llevado a las empresas y usuarios el adoptar aplicaciones web desde mediados de los 90. AJAX solo ayudará a las aplicaciones web a mejorar y conseguir un mejor resultado de cara al usuario final.

2.4.5 No es difícil su utilización.

Porque AJAX está basada en los estándares que han sido utilizados durante muchos años, muchos desarrolladores web han tenido que utilizar las tecnologías que las aplicaciones AJAX requieren. Esto significa que no es un gran esfuerzo el aprendizaje de los desarrolladores el pasar de un simple código HTML y aplicaciones web a una potente aplicación AJAX. También significa que los desarrolladores pueden actualizar poco a poco las interfaces de usuario hacia unas interfaces con AJAX; no necesita una re-escritura de la aplicación entera, se puede hacer incrementalmente.

2.4.6 Compatible con flash.

Muchos desarrolladores tienen serias dudas sobre usar Flash o AJAX. Definitivamente hay ventajas y desventajas en ambas tecnologías según la situación que se dé pero también hay muchas posibilidades y muy buenas para que ambas funcionen en conjunto.

2.4.7 Adoptados por los "grandes" de la tecnología web

La difusión de AJAX en los líderes de la industria de internet prueba que el mercado acepta y valida el uso de esta tecnología. Todo el mundo está migrando hacia AJAX incluyendo Google, Yahoo, Amazon, Microsoft (por nombrar unos pocos). Google Maps fue lo que captó la atención de los desarrolladores web. Cuando empezaron a investigar como google era capaz de llevar esa increíble herramienta dentro de un navegador sin necesidad de ningún tipo de plug-in, encontraron que AJAX estaba detrás del tema.

2.4.8 Web 2.0.

El movimiento Web 2.0 está cada vez más en auge y dando quebraderos de cabeza de muchos programadores, usuarios, y vendedores. Esto está ayudando la adopción de AJAX. Las interfaces de AJAX son un componente clave de muchas de las aplicaciones Web 2.0, como puede ser BackPack (un organizador de disco online en entorno Web) y Google Maps. Afortunadamente gracias bombo que se le está dando, acelerará la adopción de AJAX y la los beneficios de su uso lo mantendrá en escena. Una de las claves principales de Web 2.0 es el usar la red como plataforma para el desarrollo de aplicaciones, en vez de simples páginas web. Siendo importante la iteración de los usuarios con la aplicación en sí.

2.4.9 Es independiente del tipo de tecnología de servidor que se utilice.

Así como AJAX funciona en cualquier navegador, es perfectamente compatible con cualquier tipo de servidor estándar y lenguaje de programación Web. PHP, ASP. ASP.Net, Perl, JSP, Cold Fusion. El ser completamente compatible el desarrollo en éstas tecnologías ha ayudado a AJAX a que vaya cada vez más en auge.

2.4.10 Mejora la estética de la Web.

Con AJAX se puede interactuar la imaginación del desarrollador con la usabilidad de una aplicación web de forma que se pueda realizar una aplicación que si no estuviera dentro de un navegador, podría pasar por una aplicación normal de escritorio.

HERRAMIENTAS DE DESARROLLO.

3.1 NetBeans IDE 7.1 RC2.

El IDE Netbeans es un entorno premiado de desarrollo integrado disponible para Windows, Mac, Linux y Solaris. El proyecto Netbeans consiste en un IDE de código abierto y una plataforma de aplicaciones que permiten a los desarrolladores crear rápidamente aplicaciones web, empresariales, de escritorio y aplicaciones móviles utilizando la plataforma Java, así como PHP, JavaScript y Ajax, Groovy y Grails, y C / C + +.[15]

NetBeans IDE 7.1 introduce soporte para JavaFX 2.0, así como herramientas para la depuración visual de Swing y las interfaces de usuario JavaFX. Otros puntos destacados incluyen el apoyo Git integrado en el IDE, las nuevas características de depuración de PHP, Java EE y diversas mejoras en Maven, y mucho más.

3.2 JDBC.

JDBC (Java Database Connectivity) es un API para trabajar con bases de datos desde Java, independientemente de la base de datos a la que accedemos.

A la hora de conectarnos a una base de datos usando JDBC usamos un driver intermedio, que no es más que una clase ofrecida por el vendedor que implementa la interfaz Driver. Cuando se crea una instancia de una de estas clases Driver, esta se registra con el DriverManager (gestor de drivers) que es la encargada de decidir qué driver se ha de utilizar para acceder a tal o cual BBDD. El driver para trabajar con bases de datos MySQL, por ejemplo, es com.mysql.jdbc.Driver.

[15] NetBeans IDE, NetBeans IDE 7.1.2, modificado 2012, http://www.netbeans.org/community/releases/71/, [fecha de consulta: 20 mayo 2012].

A partir de la clase DriverManager, a través del método estático getConnection(String url, String usuario, String password), obtenemos el objeto Connection que representa la conexión a la base de datos a través del Driver que el DriverManager ha seleccionado. Si la base de datos permitiera acceso anónimo, se podría usar una versión de getConnection que sólo requiere un parámetro String con la URL.[16]

```
public static void conectar(String nameBD) throws SQLException{
   //permite realizar la conexion
   String url1= url.concat(nameBD);
   try {
      Class.forName("com.mysql.jdbc.Driver");
      conn = DriverManager.getConnection(url1, login, password);
      if (conn != null) {
         System.out.println("Conexión a base de datos "+url1+" ... Ok");
      }
   }catch(SQLException ex) {
      System.out.println("Hubo un problema al intentar conectarse con la base de datos
"+url1+ex.getMessage());
   }catch(ClassNotFoundException ex) {
      System.out.println(ex);
   }
   //permite la iteración con la base de dato
}
```

3.3 LIBRERIAS.

3.3.1 JDOM.

JDOM es una API desarrollada específicamente para Java y da soporte al **tratamiento de XML**: parseo, búsquedas, modificación, generación y serialización. Es un modelo similar a DOM, pero no está creado ni modelado sobre DOM. Se trata de un modelo alternativo. La principal diferencia es que DOM fue desarrollado para que fuera independiente del lenguaje, mientras que JDOM está **creado y optimizado específicamente para Java**.

Al igual que DOM, JDOM genera un **árbol de nodos** al parsear un documento XML. En cuanto a los tipos de nodos, son similares a los de DOM, aunque algunos cambia el nombre ligeramente. La jerarquía de clases también se ve algo modificada.

[16] Mundogeek.net, JDBC, 27 enero 2007, http://mundogeek.net/archivos/2007/01/27/jdbc/, [fecha de consulta: enero 2011].

Mientras que en DOM todo hereda de la clase *Node*, en JDOM casi todo hereda de la clase *Content*, que se encuentran en el paquete *org.jdom*.[17]

Codigo de Ejemplo:

```
public void readXML(File f,Element e) throws IOException,JDOMException  {

    if(f.exists()){
       SAXBuilder sax = new SAXBuilder();
       Document doc = sax.build(f);
       System.out.println(doc);
       Element raiz = doc.getRootElement();
       System.out.println(doc.getContent());
       raiz.addContent(e);
        XMLOutputter dd = new
        XMLOutputter(Format.getPrettyFormat().setEncoding("ISO-8859-1"));
          FileOutputStream out = new FileOutputStream(f);
          dd.output(doc, out);
          dd.output(doc, System.out);
        }
        else{
          System.out.println("el archivo no existe");
          createXML();
        }
  }
```

[17] latascadexela.es, Java y XML: JDOM, 8 de agosto 2008, http://www.latascadexela.es/2008/08/java-y-xml-jdom.html, [fecha de consulta: julio 2009].

3.3.2 SERVLET.

3.3.2.1 Que es un Servlet.

Un servlet es un programa que se ejecuta en el contenedor web de un servidor de aplicaciones. Los clientes pueden utilizarlo utilizando protocolo HTTP.

Un servlet acepta peticiones de un cliente, procesa la información relativa a la petición realizada por el cliente y le devuelve a este los resultados que podrán ser mostrados mediante applets, paginas HTML, etc.[18]

3.3.2.2 Caracteristicas de un servlet.

Al estar escritos en Java, son independientes de la plataforma.

Consumen menos recursos porque solo son cargados la primera vez que se solicitan sus servicios.

Son más rápidos q los CGI y que los scripts porque, por un lado se pre compilan y, por otro, no se tiene que generar un nuevo proceso cada vez que se invocan.

Son seguros y portables debido: a que se ejecutan bajo la máquina virtual de Java. Al mecanismo de excepciones de Java, y al so de administrador de seguridad de Java.

No requieren soporte para Java en el explorador del cliente, ya que operan en el dominio del servidor y envían los resultados en HTML. No obstante, se pueden utilizar otras interfaces de cliente como aplicaciones java o applets.

3.3.2.3 Estructura de un servlet.

Un servlet no es más que un objeto de alguna de las clases de la API Java Servlet que implemente la interfaz Servlet, como son **GenericServlet** y **HttpServlet.** Cuando se implementa un servicio genérico normalmente se utiliza la clase **GenericServlet.** En cambio, la clase **HttpServlet** es la idónea para servicios específicos HTTP.

[18] comp.unanleon.edu.ni, java_c8, http://www.comp.unanleon.edu.ni/u/aaltamirano/java_c8.pdf, [fecha de consulta: agosto 2010].

Código de Ejemplo:

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.sql.*;
public class Conexion extends HttpServlet {
   PrintWriter pagina;
   Connection canal = null;
   ResultSet tabla= null;
   Statement instruccion;
   String q = null;
   public void init(ServletConfig conf)
   throws ServletException {
   super.init(conf);
  }
  public void doGet(HttpServletRequest req, HttpServletResponse res)
   throws ServletException, IOException
  {
   res.setContentType("text/html");
   PrintWriter out = res.getWriter();
   out.println("<html>");
   out.println("<body>");
   out.println("<h1>Hola Mundo</h1>");
   out.println("</body>");
   out.println("</html>");
  }
  public void doPost(HttpServletRequest req, HttpServletResponse res)
   throws ServletException, IOException
  {
```

```
    res.setContentType("text/html");
    PrintWriter out = res.getWriter();
    out.println("<html>");
    out.println("<body>");
    //out.println("<h1>Hola Mundo</h1>");
     try { Class.forName("com.mysql.jdbc.Driver");
     System.out.println("entramos a la conex:   ");
            canal=
DriverManager.getConnection("jdbc:mysql://localhost:3306/prueba", "root", "");
            instruccion =   (Statement) canal.createStatement();
            out.println("<h1>Hola Mundo</h1>");
     } catch(java.lang.ClassNotFoundException e){
         System.out.println("del java lang:   "+e.getMessage());
     } catch(SQLException e) {
          System.out.println("del sql:   "+e.getMessage());
     }
    out.println("</body>");
    out.println("</html>");
  }
}
```

3.3.3 QUAQUA LOOK AND FEEL

Es un Look & Feel (Skins) que permite cambiar la apariencia de las ventanas swing en java.

Biblioteca de interfaz de usuario para las aplicaciones Java que deseen adherirse firmemente a las directrices de interfaz de Apple Humanos para Mac OS X.[19]

[19] Werner Randelshofer, Quaqua Look and Feel, http://www.randelshofer.ch/quaqua/, [fecha de consulta: agosto del 2010].

Para aplicar estos temas a nuestras aplicaciones, simplemente hay que descargar el .jar del Look & Feel que deseamos y ubicarlo en la carpeta de nuestro proyecto para después agregarlo a la librería del mismo. Luego, en el constructor de cada una de las clases que componen la vista de cada FrameView o JFrame agregar las siguientes líneas:

```
JFrame.setDefaultLookAndFeelDecorated(true);
UIManager.setLookAndFeel("NombreDelLook&Feel");
```

Figura 5: Interfaz QUAQUA

3.4 Tomcat.

3.4.1 Definición.

Apache Tomcat (o Jakarta Tomcat) es un software desarrollado con Java (con lo cual puede funcionar en cualquier sistema operativo, con su máquina virtual java correspondiente) que sirve como servidor web con soporte de servlets y JSPs.[20]

3.4.2 Entorno.

Tomcat es un servidor Web con soporte para servlets y JSPs. No es un servidor de aplicaciones, como JBoss o JOnAS. Trae incluido el compilador Jasper, que compila JSPs convirtiéndolas en servlets. El motor de servlets de Tomcat a menudo se presenta en combinación con el servidor Web Apache. A partir de la versión 4.0, Tomcat utiliza el contenedor de servlets Catalina.

[20] Ajpdsoft, Definicion Tomcat, http://www.ajpdsoft.com/modules.php?name=Encyclopedia&op=content&tid=769, [fecha de consulta: agosto 2010].

Tomcat puede funcionar como servidor Web por sí mismo. Al principio de su desarrollo existió la percepción de que la utilización de Tomcat de forma autónoma era sólo recomendable para entornos de desarrollo y entornos con mínimos requisitos de velocidad y gestión de transacciones. Actualmente ya no existe esa percepción y Tomcat es usado como servidor Web independiente en entornos con alto nivel de tráfico y alta disponibilidad.

El hecho de que Tomcat fue escrito en Java, hace posible que funcione en cualquier sistema operativo que disponga de la máquina virtual Java (también se puede utilizar con XAMPP).[21]

3.4.3 Estructura de Directorios.

Estos son algunos de los directorios de Tomcat: [22]

- **/ Bin** - arranque, parada, y otros scripts. El *.sh archivos (para los sistemas Unix) son copias funcionales de la *.bat archivos (para sistemas Windows). Desde la línea de comandos de Win32 carece de algunas funciones, hay algunos archivos adicionales de aquí.
- **/ Conf** - Los archivos de configuración y DTD relacionados. El archivo más importante aquí es server.xml. Es el archivo de configuración principal para el contenedor.
- **/ Logs** - Los archivos de registro están aquí de forma predeterminada.
- **/ Webapps** - Aquí es donde las aplicaciones web van.

[21] Centro de Desarrollo Territorial Holguín UCI, Servidor Tomcat, Junio 2011, http://www.ecured.cu/index.php/Servidor_Tomcat, [fecha de consulta: 20 mayo 2012].
[22] ***Apache Software Foundation***, Apache Tomcat 7, modificado 31 Marzo 2012, http://tomcat.apache.org/tomcat-7.0-doc/introduction.html, [fecha de consulta: 20 mayo 2012].

D. RESULTADOS.

1. DESARROLLO DE LA PROPUESTA ALTERNATIVA.

En el desarrollo de la propuesta se fue analizando cada uno de los objetos a investigar los cuales los describimos en la siguiente tabla:

OBJETO DE INVESTIGACIÓN	PROPUESTA ALTERNATIVA
Construir un Framework para el desarrollo de aplicaciones Web 2.0 con acceso a base de datos MySql.	Se construye un IDE que permita a los usuarios-programadores realizar sus aplicaciones web de una manera dinámica, logrando de esta manera utilizar el Framework TOOLWEB con toda su funcionalidad para el cual fue construido.
Implementar el patrón Modelo Vista Controlador en el diseño del Framework.	Se implemento el patrón de diseño MVC de la siguiente manera, Para la vista, diseñar las distintas interfaces gráficas correspondientes para la creación de cada uno de los componentes básicos que intervienen en el proceso de desarrollo de las aplicaciones web; Para el modelo, se desarrolla las plantillas xsl para la generación y creación de las paginas html y los servlets. El controlador se encuentra en el paquete LogicaProgramacion, en el cual se utiliza la librería JDOM que nos permite realizar la construcción y generación de código para los archivos .html y .java; Además dentro de este paquete se realiza la construcción de los archivos JavaScript y CSS que son parte de las aplicaciones web desarrolladas con TOOLWEB.
Diseñar el Framework utilizando la tecnología AJAX que es parte de la Web 2.0.	Dentro del desarrollo de TOOLWEB se creó una clase AdministrarJavaScript, la misma que nos permite crear el archivo JavaScript y la generación de código de la función Ajax que contiene el Objeto XMLHTTPRequest que

	permite la comunicación asincrónica con el servidor, característica principal de AJAX.
Diseñar los componentes básicos que ofrecerá el Framework.	Los componentes básicos que se pueden desarrollar con la herramienta TOOLWEB se encuentran distribuidos en el panel denominado PALETA (Fig. 17), donde cada uno tiene su interfaz gráfica, en la cual se ingresan sus respectivos atributos, estos se los podrá visualizar en la ventana principal del IDE; la información de los componentes con sus atributos se visualiza en la pestaña CodigoXML, el código fuente en la pestaña CodigoHTML y el componente grafico en el panel de Vista Previa.
Permitir que las aplicaciones Web basadas en el Framework tengan el acceso a la base de Datos MySql.	Dentro de TOOLWEB se podrá establecer la conexión a la base de datos ingresando el usuario y contraseña para luego poder crear y eliminar Bases de Datos, crear y añadir tablas con sus respectivos campos necesarios, y poder visualizar las Bases de Datos con sus respectivas tablas.
Las aplicaciones se almacenaran en un archivo XML.	Para almacenar la información de las aplicaciones Web desarrolladas en TOOLWEB en un archivo XML, se crea una clase java (SaveProyectXML.java) que crea el archivo Proyecto_nombredelproyecto.xml, el mismo que tiene la información de la aplicación web creada con sus directorios y subdirectorios con sus respectivos archivos de cada aplicación web creada.
Implementación del Framework.	Implementar el Framework TOOLWEB a los estudiantes de la carrera de Ingeniería en Sistemas de la Universidad Nacional de Loja, como una herramienta de desarrollo de aplicaciones web.

Crear una aplicación de Ejemplo basado en el Framework desarrollado.	Se creó una aplicación web denominada Registro utilizando la herramienta TOOLWEB, quedando demostrado el funcionamiento del Framework. (ver Anexo 4).

Tabla 1: Evaluación del objeto de Investigación

Luego de haber estudiado los objetos de investigación se obtuvieron soluciones necesarias para el desarrollo de esta herramienta denominada TOOLWEB. En la construcción de este Framework se hizo un plan de actividades las mismas que se desarrollaron de la siguiente manera:

La información obtenida en el Análisis de requisitos, ha contribuido para la creación del modelo del dominio y el prototipo de pantallas, procediendo a la elaboración de los casos de uso con sus descripciones respectivas.

Siguiendo en el desarrollo tenemos la fase de Diseño en la cual utilizamos la información obtenida en la tarea de Análisis, la misma que aporta en esta fase para realizar el modelado del sistema, es decir; los diagramas de secuencia, clases, componentes y despliegue.

Una vez terminado el diseño del sistema se procede a la programación para la creación del Framework TOOLWEB utilizando el lenguaje de programación Java y la herramienta de desarrollo NetBeans IDE 7.1 RC2.

Realizar un plan de validación con la finalidad de tener información relacionada con el funcionamiento, para ello realizamos las pruebas de funcionalidad, aceptabilidad y usabilidad; brindando la capacitación sobre la utilización del Framework TOOLWEB a los alumnos de sexto y octavo módulo de la carrera de Ingeniería en Sistemas de la Universidad Nacional de Loja.

2. VALORACION TECNICA ECONOMICO – AMBIENTAL.

Se necesitó la participación de los siguientes recursos humanos:

- Desarrolladores
- Alumnos de los módulos de la carrera de Ingeniería en Sistemas
- Asesores

RECURSOS HUMANOS				
HUMANOS	**CANTIDAD**	**Valor de Hora**	**Nª de horas**	**Valor total**
Desarrolladores	2	0.00	800	0.00
Alumnos de la Carrera Ing. Sistemas	80	0.00	12	0.00
Asesores	2	20.00	50	1000.00
			SUBTOTAL:	**1000.00**

Tabla 2: Recursos Humanos

RECURSOS MATERIALES

RECURSOS MATERIALES			
Material	**Cantidad**	**Unidad**	**Valor total**
Hojas de papel bond (Resmas)	5	3,75	18.75
Carpetas	20	0.25	5.00
Perforadora	1	8,00	8,00
Borrador	4	0.25	1.00
Lápices	8	0.25	2.00
Esferos	8	0.25	2.00
Calculadora	1	5,00	5,00
Clips (caja)	1	0.50	0.50
Cds en blanco	20	0.50	10.00
Cartuchos de Tinta Negra	15	5.50	55.00
Cartuchos de Tinta Color	10	6,00	60,00
Anillados	6	1,50	9,00
Empastados	6	8,00	48,00
		SUBTOTAL:	**222,25**

Tabla 3: Recursos Materiales

SERVICIOS BÁSICOS

SERVICIOS BASICOS	
SERVICIO	**VALOR TOTAL**
AGUA	100,00
LUZ	400,00
TELEFONO	576,00
INTERNET	900,00
TRANSPORTE	2160,00
SUBTOTAL:	**4136,00**

Tabla 4: Servicios Básicos

RECURSOS TÉCNICOS Y TECNOLOGICOS

RECURSOS TECNICOS Y TECNOLOGICOS			
HARDWARE			
EQUIPO	**CANTIDAD**	**VALOR UNITARIO**	**VALOR TOTAL**
Memory Flash	4	18,00	72,00
Computador MacBook Pro	1	2233.80	2233.80
Laptop Hp pavilion Dv4-1212la	1	788.40	788.40
SOFTWARE			
Microsoft Office 2007	1	130,00	130,00
Netbeans IDE 7.1RC2	2	Gratuito	0,00
Java jdk1.7	2	Gratuito	0,00
MySql 5.5	2	Gratuito	0,00
Tomcat 7.0	2	Gratuito	0,00
Enterprise Architect	1	150,00	150,00
		SUBTOTAL:	**3374,20**

Tabla 5: Recursos Técnicos

RECURSOS ECONOMICÓS

RECURSOS	VALOR TOTAL
HUMANOS	1000,00
MATERIALES	222,25
SERVICIOS BASICOS	4136,00
RECURSOS TÉCNICOS Y TECNOLOGICOS	352,00
IMPREVISTOS	500,00
TOTAL:	**6.210,25**

Tabla 6: Recursos Económicos

E. DISCUSION.

1. Modelado del Sistema.

1.1 Requerimientos Funcionales.

El Framework TOOLWEB permite:

REF. REQUERIMIENTO	DESCRIPCION	CATEGORIA
RF001	Crear aplicaciones web.	visible
RF002	Mostrar el contenido de aplicación creada en un árbol.	visible
RF003	Crear archivos Xml, JavaScript y Css en las aplicaciones Web	oculto
RF004	Mostrar el código respectivo de los archivos creados.	visible
RF005	Crear los formularios Web utilizando los componentes básicos de HTML.	oculto
RF006	Observar una vista previa del formulario de la aplicación web.	visible
RF007	Genera el archivo HTML del código de los formularios.	oculto
RF008	Generar el archivo Java del código del servlet.	oculto
RF009	Navegar entre las paginas HTML de la aplicación.	visible
RF010	Crea un archivo servlet.	oculto
RF011	Visualizar el código del archivo del servlet.	visible
RF012	Editar los archivos Xml, JavaScript, Css, Html, Servlet.	visible
RF013	Crear la base de datos MySql.	visible
RF014	Que las aplicaciones tengan acceso a la base de datos MySql.	oculto
RF015	Generar un archivo xml con los datos de	visible

	la conexión a la base de datos MySql.	
RF016	Crear las tablas de la base datos con sus respectivos campos.	visible
RF017	Visualizar las tablas de la base de datos MySql con sus correspondientes campos y datos.	visible
RF018	Eliminar la base de datos MySql.	visible
RF019	Guardar el nombre y ubicación de las aplicaciones web con sus respectivos archivos en un archivo XML.	oculto
RF020	Imprimir el código de los archivos de la aplicación web.	visible
RF021	Tener acciones de un editor como copiar, cortar, pegar, deshacer, rehacer y seleccionar todo al editarlo.	visible
RF022	Guardar los cambios realizados en los editores de texto.	oculto
RF023	Validar los errores de sintaxis dentro de los editores de texto.	oculto
RF024	Abrir aplicaciones web con su contenido.	visible
RF025	Abrir archivos Xml, Xsl, JavaScript, Java, Css, Html.	visible

Tabla 7: Requerimientos Funcionales

1.2 Requerimientos no Funcionales.

REFERENCIA	ATRIBUTOS	DESCRIPCION
RF001	Tiempo de Respuesta	La creación de la aplicación web dependerá de las características del ordenador.
RF002	Interfaz Grafica	El Framework TOOLWEB estará orientado al manejo de ventanas, cuadros de diálogos.
RF003	Facilidad de Uso	El Framework TOOLWEB cuenta con ventanas amigables, mensajes de aviso, que permitirán un fácil manejo del mismo.
RF004	Plataforma del Sistema Operativo	Multiplataforma
RF005	Plataforma de Lenguajes de Programación	Java
RF006	Mensajes	El framework TOOLWEB presentara mensajes de confirmación de las acciones realizadas.
RF007	Servidores	Se utiliza servidor web Apache Tomcat para la navegación de las aplicaciones

Tabla 8: Requerimientos No Funcionales

2. Modelado del Dominio.

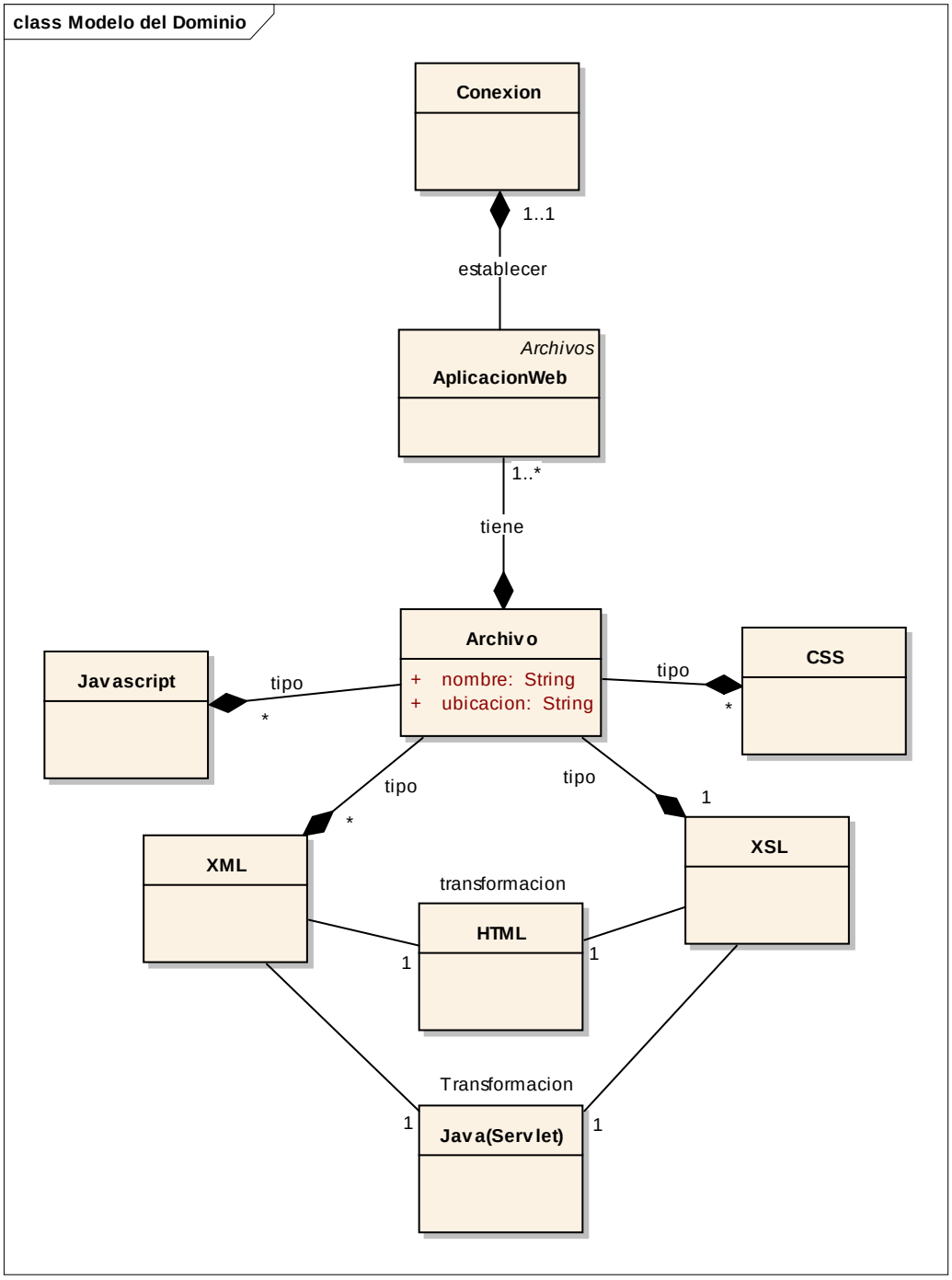


Figura 6: Modelo del dominio

3. Modelamiento de Casos de Uso del Sistema.

3.1 Diagramas de Casos de Uso.

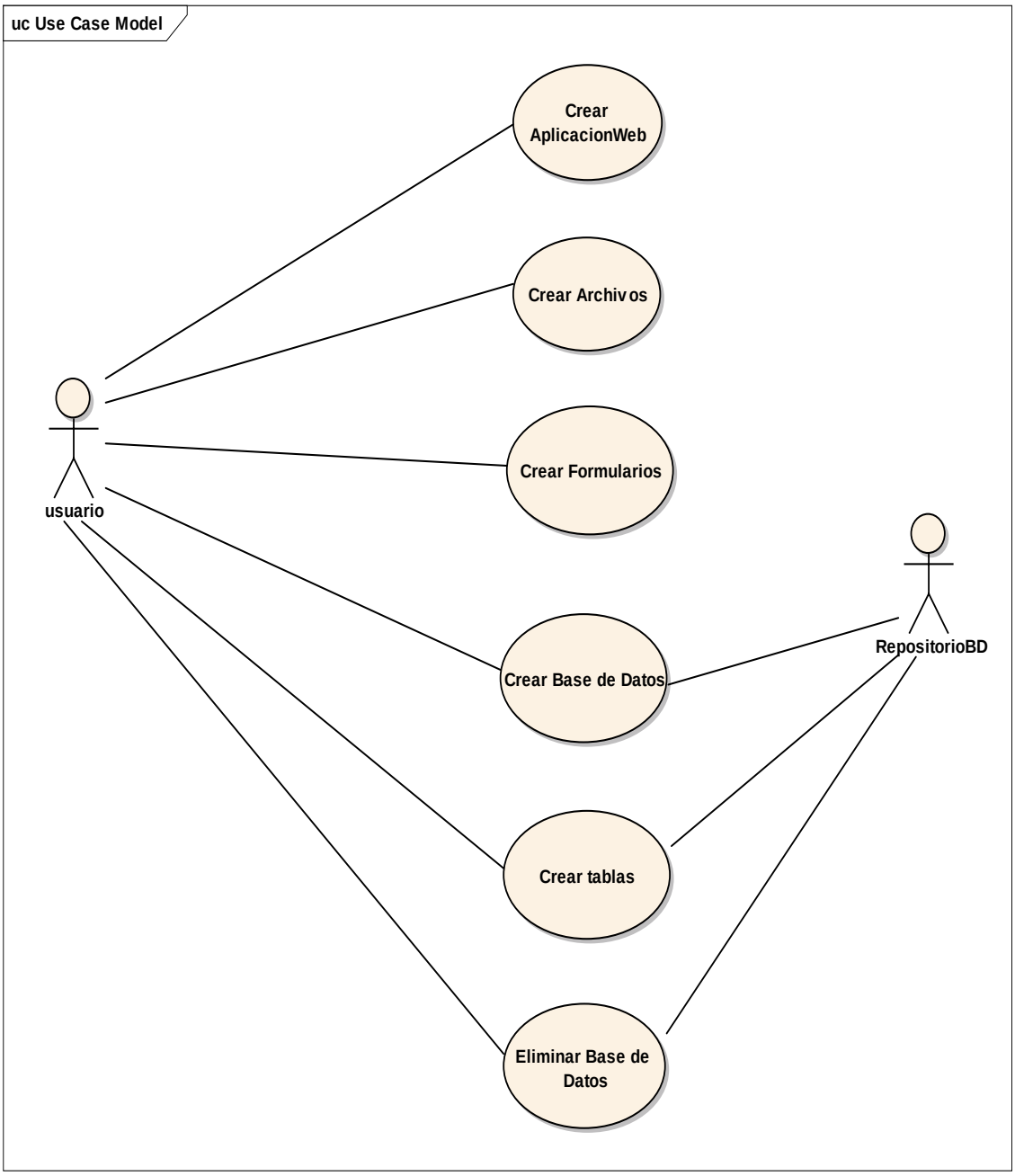


Figura 7: Diagramas de Caso de Uso

3.2 Descripción de Casos de Uso.

CREAR APLICACIÓN WEB

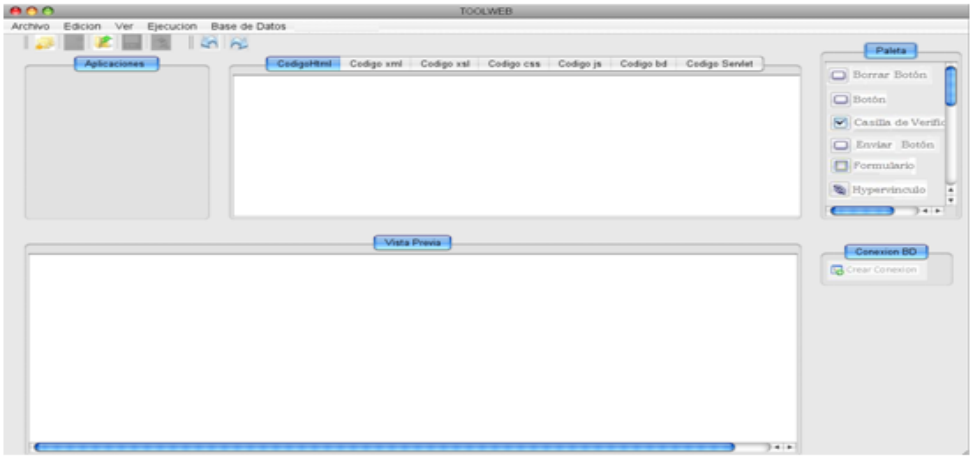


Figura 8: Pantalla Ventana Principal

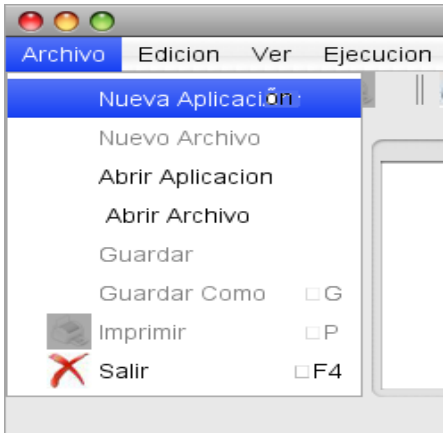


Figura 9: Pantalla Menú Archivo opción Nueva Aplicación.

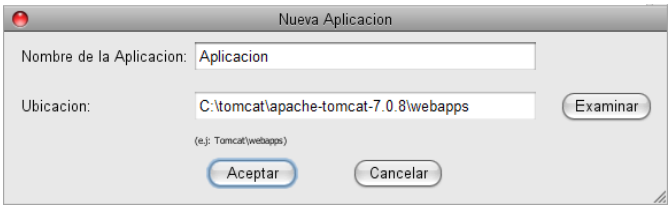


Figura 10: Pantalla Ventana Nueva Aplicación

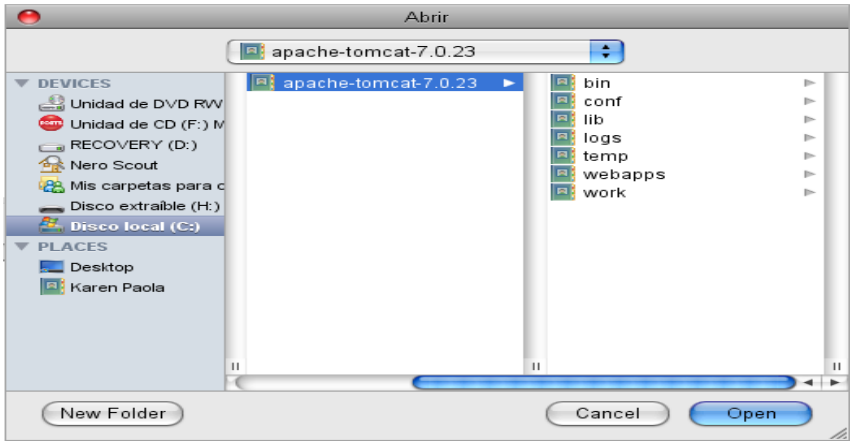


Figura 11: Pantalla Ventana Abrir

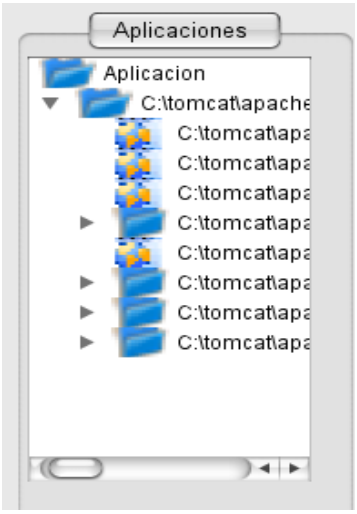


Figura 12: Pantalla Estructura del árbol de las aplicaciones

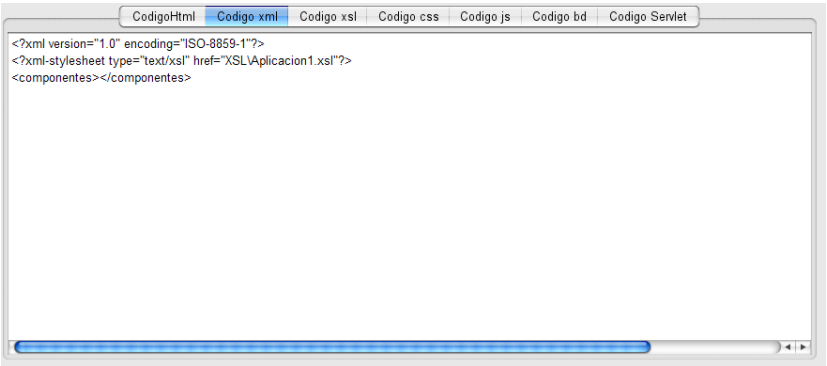


Figura 13: Pantalla Panel Código del Archivo xml

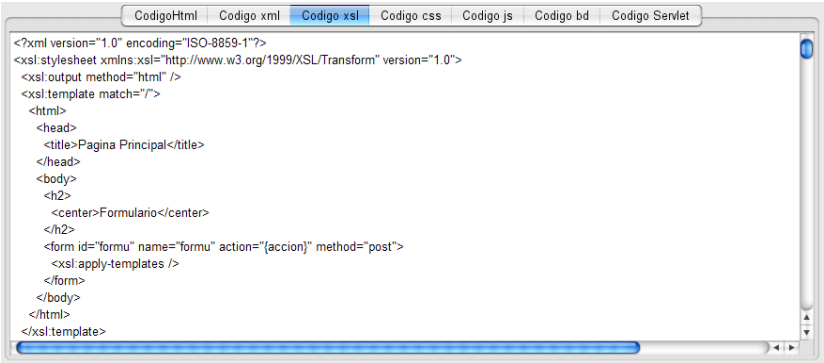


Figura 14: Pantalla Panel Código del Archivo xsl

Nombre:	Crear Aplicación Web
Actores:	Usuario
Propósito:	Permitir crear, editar una aplicación Web.
Visión General:	El usuario podrá crear y editar una aplicación.
Tipo:	Primario, esencial.
Referencia:	RF001, RF002, RF003, RF004, RF013, RF021, RF022, RF023, RF024, RF025, RF026, RF027.

Curso Típico de Eventos

1. El usuario elige la opción **Nueva Aplicación** del menú **Archivo** en la ventana **TOOLWEB**.
2. El sistema presenta la ventana **NUEVA APLICACIÓN.**
3. El usuario ingresa el nombre en el campo de texto **Nombre de la Aplicación**
4. El usuario selecciona la opción **Examinar** en la ventana **NUEVA APLICACIÓN.**
5. El sistema presenta la ventana **ABRIR**.
6. El usuario elige el directorio TOMCAT/WEBAPPS para crear la aplicación.
7. El usuario selecciona la opción **Abrir.**
8. El sistema muestra la dirección del directorio TOMCAT/WEBAPPS, en el campo de texto **Ubicación** en la ventana **NUEVA APLICACIÓN.**
9. El usuario selecciona la opción **Aceptar**.

10. El sistema valida los datos introducidos para la creación de la aplicación web.
11. El sistema crea la aplicación web con todo su contenido y con los archivos iniciales de XML/XSL para la página Html y los archivos XML/XSL para la creación del archivo JAVA.
12. El sistema crea el archivo **Proyecto.xml,** donde está contenido el nombre y directorio de los archivos de la aplicación recién creada.
13. El sistema presenta en un árbol el contenido de la aplicación web HTML, XML, XSL, JavaScript, CSS, WEB-INF, META-INF con sus respectivos archivos, en el panel **Aplicación** en el panel **Área de Trabajo** en la ventana **TOOLWEB.**
14. El sistema presenta los archivos XML/XSL en los editores del panel **Código XML, Código XSL,** respectivamente en el panel **Área de Trabajo** a la ventana **TOOLWEB.**

Curso Alterno de Eventos

A. El usuario selecciona la opción **Nueva Aplicación** de la barra de herramientas de la ventana **NUEVA APLICACIÓN**.

A1. El use case vuelve al paso 2.

B. El usuario elige la opción **Abrir Aplicación** del menú **Archivo** en la ventana **TOOLWEB**.

B1. El sistema presenta la ventana **ABRIR**.

B2. El usuario elige la aplicación que desea abrir en la ventana **ABRIR.**

B3. El usuario selecciona la opción **Abrir** en la ventana **ABRIR.**

B4. El sistema presenta el contenido de la aplicación web HTML, XML, XSL, JavaScript, CSS, WEB-INF, META-INF con sus respectivos archivos, en el árbol **Aplicación** en el panel **Área de Trabajo** en la ventana **TOOLWEB.**

B5. El usuario procede a editar la aplicación web.

C. El usuario elige la opción **Cancelar** de la ventana Abrir

C5. El sistema cierra la ventana **Abrir.**

C6. El sistema presenta la ventana **NUEVA APLICACIÓN.**

D. El sistema presenta mensaje de error **La aplicación ya existe.**

D5. El usuario selecciona la opción aceptar del mensaje.

D6. El sistema muestra la ventana **NUEVA APLICACIÓN.**

D7. El usuario cambia el nombre de la aplicación web.

D8. El use case vuelve al paso 5. **E.** El usuario selecciona la opción **Salir** en el menú **Archivo** de la ventana **TOOLWEB**. E1. El use case termina

Tabla 9: Descripción de casos de Usos Crear Aplicación Web

CREAR ARCHIVOS

Figura 15: Pantalla Menú Archivo opción Nuevo Archivo

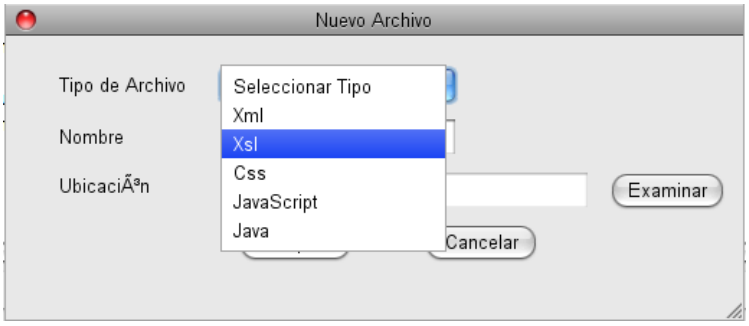


Figura 16: Pantalla Nuevo Archivo

Nombre:	Crear Archivos
Actores:	Usuario
Propósito:	Crear archivos XML, CSS, JavaScript, Java dentro de la aplicación web.
Visión General	El usuario crea la aplicación web para proceder a crear los diferentes tipos de archivos necesarios para su aplicación.
Tipo:	Primario Esencial.
Referencia:	RF003, RF004, RF013, RF023, RF024, RF025, RF026.

Curso Típico de Eventos
1. El usuario elige la opción **Nuevo Archivo** del menú Archivo de la ventana **TOOLWEB.** **2.** El sistema presenta la ventana **NUEVO ARCHIVO.** **3.** El usuario elige uno de los tipos de archivos (XML, CSS, JavaScript, Java) del menú desplegable **Tipo de Archivo.** **4.** El sistema muestra la dirección donde se creara el archivo en el campo de texto **Ubicación**. **5.** El usuario ingresa el nombre del archivo en el campo de texto **Nombre.** **6.** El usuario selecciona la opción **Aceptar.** **7.** El sistema valida los datos introducidos para la creación del archivo. **8.** El sistema crea el archivo respectivo, en el caso de que sea un archivo xml también creara un archivo java. **9.** El sistema actualiza el archivo **Proyecto.xml** con el nombre y directorio del archivo recientemente creado. **10.** El sistema actualiza árbol con el contenido de la aplicación web HTML, XML, XSL, JavaScript, CSS, JAVA, WEB-INF, META-INF con el archivo creado anteriormente en el panel **Aplicación** en el panel **Área de Trabajo** de la ventana **TOOLWEB.** **11.** El sistema muestra código del archivo creado en el editor del panel **Código** del archivo respectivo, en el panel **Área de Trabajo** en la ventana **TOOLWEB.**
Curso Alterno de Eventos

A. El usuario selecciona la opción **Nuevo Archivo** de la barra de herramientas de la ventana **TOOLWEB.**

A1. El use case vuelve al paso 2.

B. El usuario selecciona la opción **Examinar** de la ventana **NUEVO ARCHIVO.**

B4. El sistema presenta la ventana **ABRIR.**

B5. El usuario elige el directorio donde desea crear el archivo**.**

B6. El usuario selecciona la opción **Abrir.**

B7. El sistema muestra la dirección elegida en el campo de texto **Ubicación** de la Ventana **NUEVO ARCHIVO.**

B8. El use case vuelve al paso 6.

C. El sistema presenta mensaje de error **Archivo ya existe.**

C7. El sistema muestra la ventana **NUEVO ARCHIVO.**

C8. El usuario procede a cambiar el nombre del archivo en el campo de texto **Nombre.**

C9. El use case vuelve al paso 6.

D. EL usuario selecciona la opción **Cancelar** de la Ventana **NUEVO ARCHIVO**.

D6. El sistema cierra la ventana **NUEVO ARCHIVO.**

D7. El sistema regresa a la ventana **TOOLWEB.**

Tabla 10: Descripción de casos de Usos Crear Archivos

CREAR FORMULARIOS

Figura 17: Pantalla Panel Paleta.

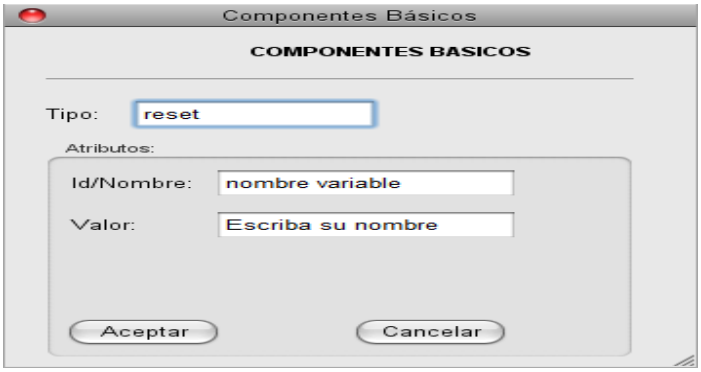


Figura 18: Pantalla Ventana Componentes Básicos (Borrar Botón)

Figura 19: Pantalla Ventana Componentes Básicos (Botón)

Figura 20: Pantalla Ventana Componentes Básicos (Casilla Verificación)

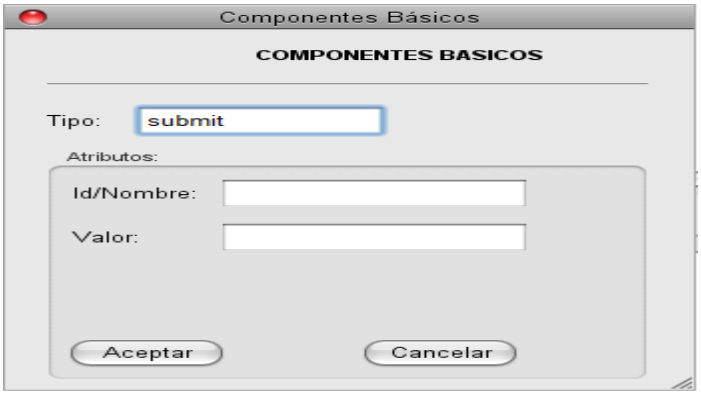


Figura 21: Pantalla Ventana Componentes Básicos (Enviar Botón)

Figura 22: Pantalla Ventana Formulario

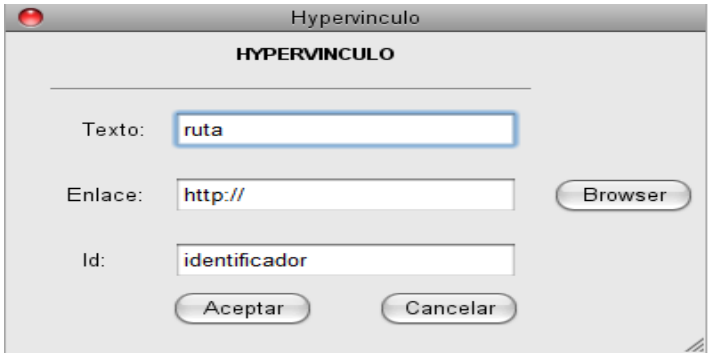


Figura 23: Pantalla Ventana Hypervinculo

Figura 24: Pantalla Ventana Componentes Básicos (Campo de Imagen)

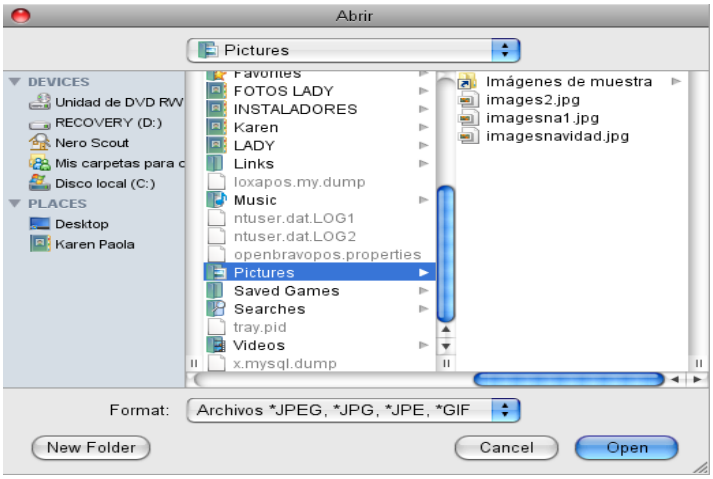


Figura 25: Pantalla Ventana Imagen

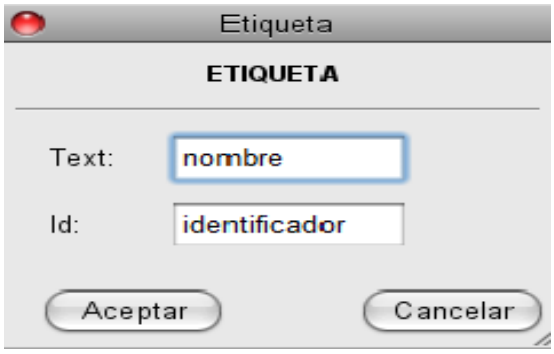


Figura 26: Pantalla Ventana Etiqueta

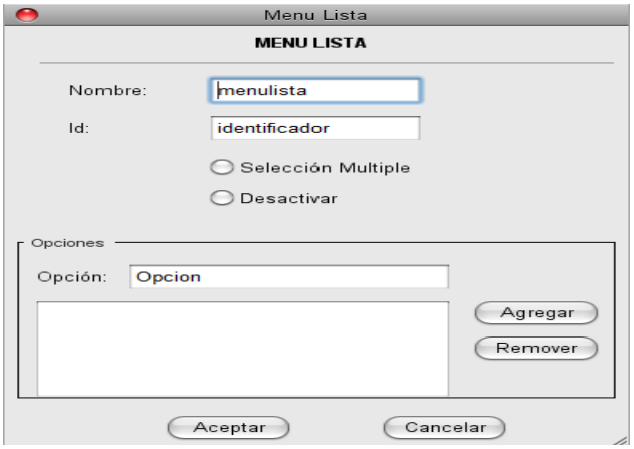


Figura 27: Pantalla Ventana Menú Lista

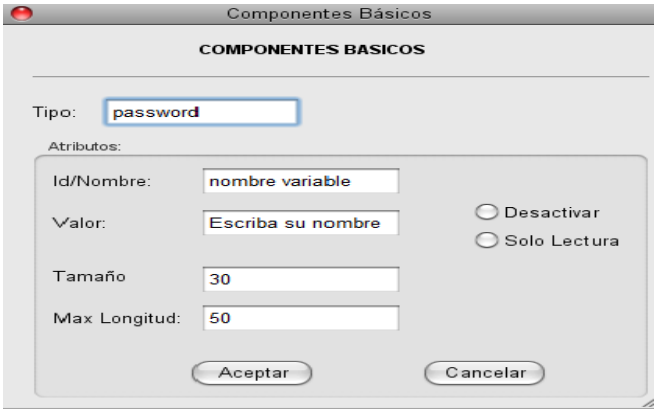


Figura 28: Pantalla Ventana Contraseña

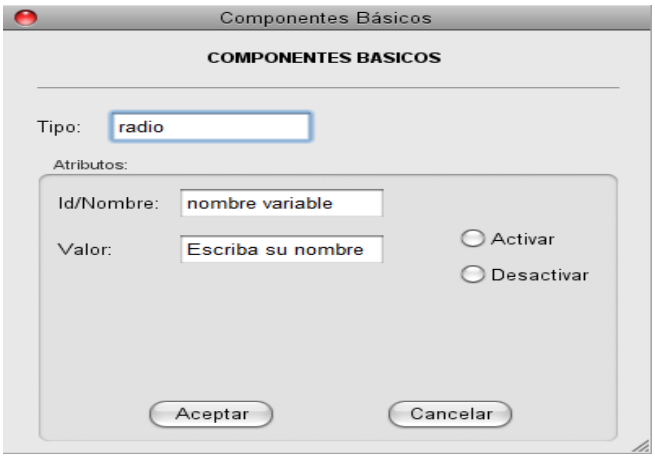


Figura 29: Pantalla Ventana Componentes Básicos (Radio Botón)

Figura 30: Pantalla Ventana Grupo de Radios

Figura 31: Pantalla Ventana Tabla

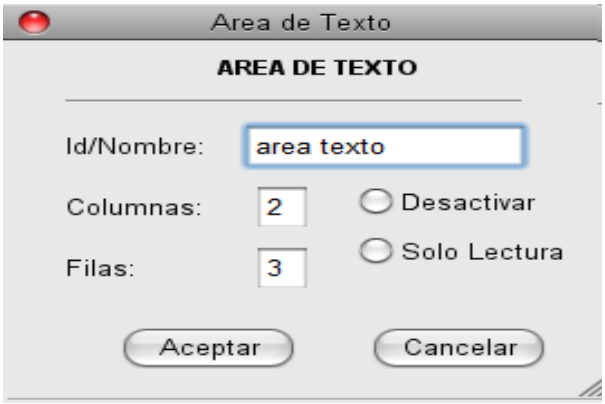


Figura 32: Pantalla Ventana Área de Texto

Figura 33: Pantalla Ventana Componentes Básicos (Campo de Texto)

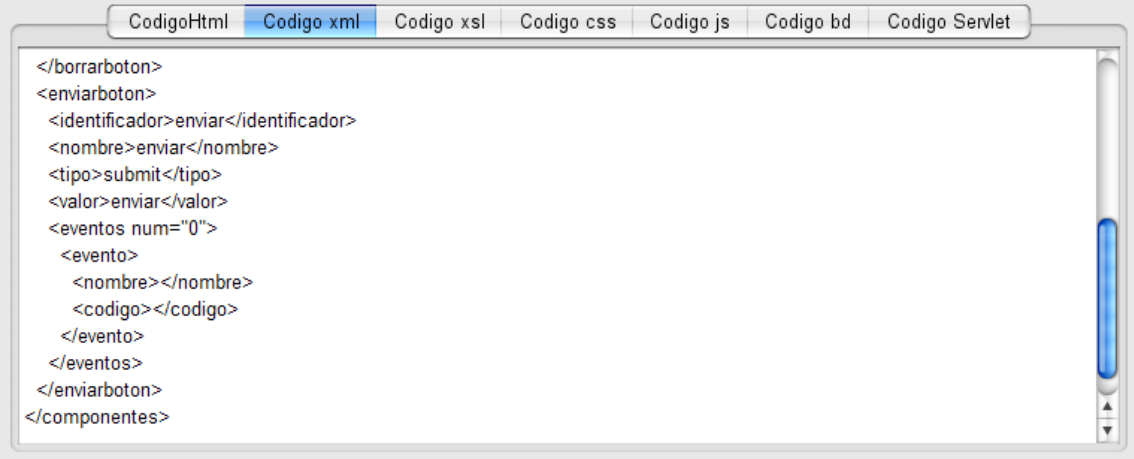


Figura 34: Pantalla TOOLWEB Panel Código xml

Figura 35: Pantalla TOOLWEB Panel Código Servlet

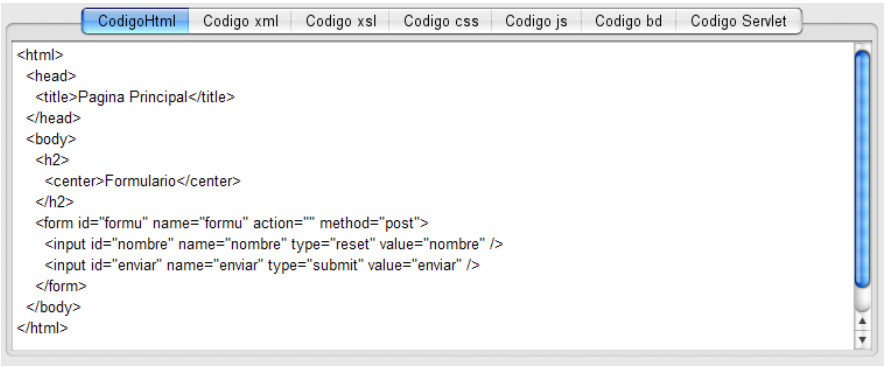


Figura 36: Pantalla TOOLWEB Panel Código Html

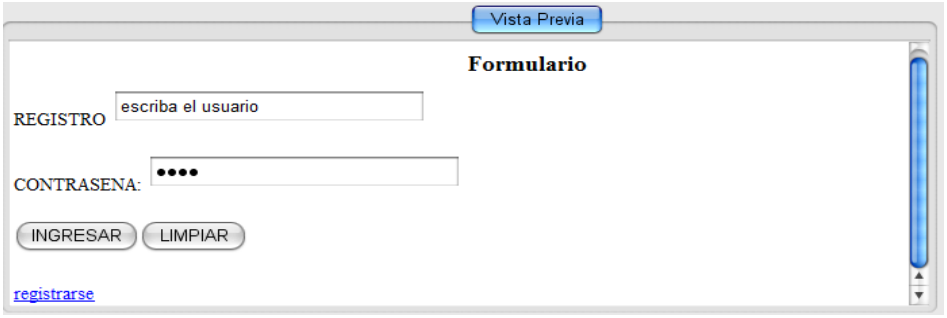


Figura 37: Pantalla Panel Vista previa de la página HTML

Nombre:	Crear Formularios
Actores:	Usuario
Propósito:	Crear un formulario con componentes necesarios para la aplicación web.
Visión General:	El usuario ha creado su aplicación web y debe agregar los componentes para la creación de las paginas HTML.
Tipo:	Primario Esencial
Referencia:	RF005, RF007, RF008, RF009, RF010, RF011, RF021, RF022, RF023, RF024, RF025.
Curso Típico de Eventos:	

1. El usuario elige el(os) componente(s) deseado del **Panel Paleta** en la ventana **TOOLWEB.**
2. El sistema presenta la ventana **COMPONENTES BASICOS**.
3. El usuario ingresa los datos del componente, atributos requeridos para su creación en la ventana **COMPONENTES**.
4. El usuario selecciona la opción **Aceptar.**
5. El sistema valida los datos introducidos para la creación del componente.
6. El sistema presenta el código de creación del componente en el archivo XML en el editor del panel **Código XML**, en el panel **Área de Trabajo.**
7. El usuario elige la opción **Vista Previa** del menú **Ver**.
8. El sistema crea el archivo html.
9. El sistema realiza la transformación del archivo XML y XSL en el archivo Html.

10. El sistema realiza la transformación del archivo de propiedades XML y XSL creando un archivo java con los respectivos parámetros de los componentes de entrada/salida. **11.** El sistema actualiza árbol con el contenido de la aplicación web HTML, XML, XSL, JavaScript, CSS, WEB-INF, META-INF con el archivo creado anteriormente en el panel **Aplicación** en el panel **Área de Trabajo** de la ventana **TOOLWEB.** **12.** El sistema presenta el código del archivo java en el panel del editor **Código Servlet** en el panel **Área de Trabajo** de la ventana **TOOLWEB.** **13.** El sistema presenta el código del archivo html en el panel del editor **Código HTML** en el panel **Área de Trabajo** de la ventana **TOOLWEB.** **14.** La página HTML se visualiza en el panel **Vista Previa**.
Curso Normal de Eventos:
A. EL usuario selecciona el botón **Cancelar** de la Ventana **COMPONENTE BASICOS**. A4. El sistema cierra la ventana **COMPONENTE BASICOS**. A5. El sistema regresa a la ventana **TOOLWEB.**

Tabla 11: Descripción de casos de Usos Crear Formularios

CREAR BASE DE DATOS

Figura 38: Pantalla Ventana Establecer Conexión

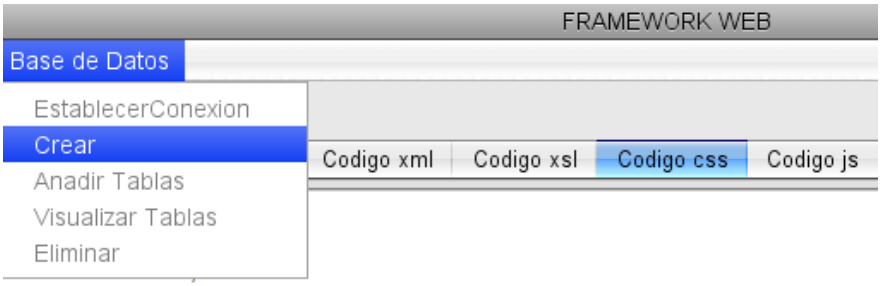


Figura 39: Pantalla TOOLWEB Menú base de Datos opción Crear

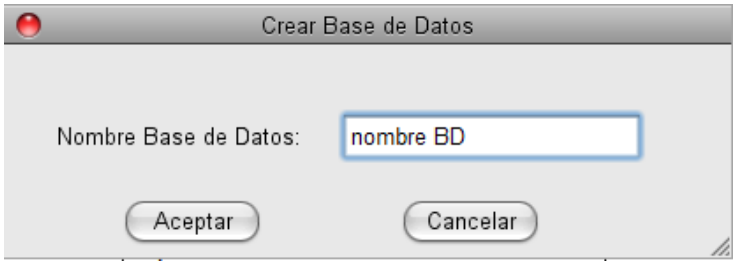


Figura 40: Pantalla Ventana crear Base de Datos

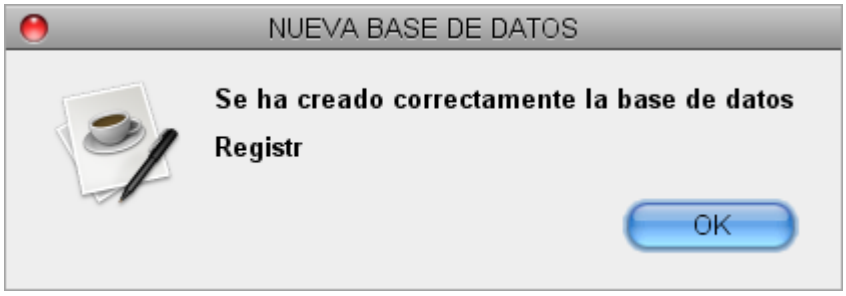


Figura 41: Pantalla Ventana Nueva Base de Datos creada

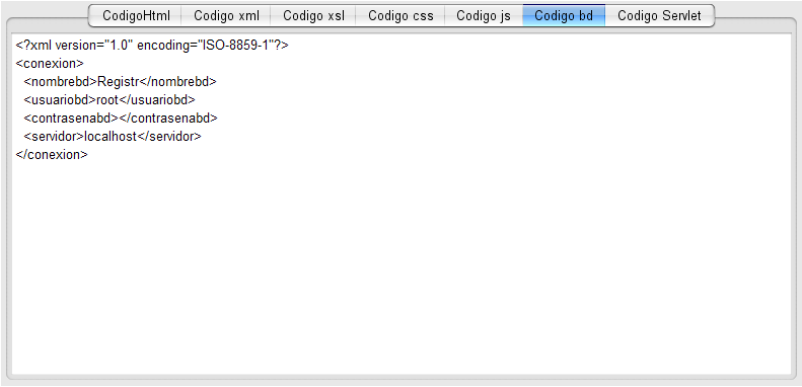


Figura 42: Pantalla TOOLWEB Código de la Base de Datos

Nombre:	Crear Base de Datos
Actores:	Usuario
Propósito:	Permitir crear, una base de datos y la conexión de la BD.
Visión General:	El usuario tiene su aplicación web para proceder hacer la conexión a la base de datos ingresando el usuario y contraseña del motor de base de datos MySql, estableciendo de esta manera la conexión a la base de datos.

Tipo:	Primario Esencial
Referencia:	RF016, RF017.
Curso Típico de Eventos:	
1. El usuario elige la opción **Crear** del menú **Base de Datos** en la ventana **TOOLWEB.** **2.** El sistema muestra la ventana **CREAR BASE DE DATOS**. **3.** El usuario ingresa el nombre de la base de datos en el campo de texto **Nombre Base de Datos**. **4.** El usuario selecciona la opción **Aceptar**. **5.** El sistema valida el dato introducido en el campo de texto **Nombre Base de Datos**. **6.** El sistema presenta un mensaje de información **"Se ha creado correctamente la base de datos."** en la ventana Nueva Base de Datos**.** **7.** El usuario selecciona la opción **Aceptar** el mensaje, quedando creada la base de datos. **8.** El sistema crea un archivo xml llamado **Datos-BD,** con los datos de la conexión BD, como el nombre de la base de datos, usuario, contraseña. **9.** El sistema muestra el código del archivo **Datos-BD,** en el panel del editor **Código BD** en el panel **Área de Trabajo** de la ventana **TOOLWEB.** **10.** El sistema adiciona la información de los atributos de la base de datos: nombre, usuario y contraseña al archivo de propiedades.xml.	
Curso Normal de Eventos:	
A. El usuario selecciona la opción **Crear Conexión** de la **Conexión BD** del panel **Área de Trabajo** en la ventana **TOOLWEB.** A1. El sistema presenta un mensaje de información **"Desea crear Base de Datos"** A2. El usuario selecciona la opción **Si** del mensaje de información. A3. El use case vuelve al paso 2. **B.** EL usuario selecciona la opción **Cancelar** de la Ventana **CREAR BASE DE DATOS.** C2. El sistema cierra la ventana **CREAR BASE DE DATOS.** C3. El sistema regresa a la ventana **TOOLWEB.**	

Tabla 12: Descripción de casos de Usos Crear Base de Datos

CREAR TABLAS

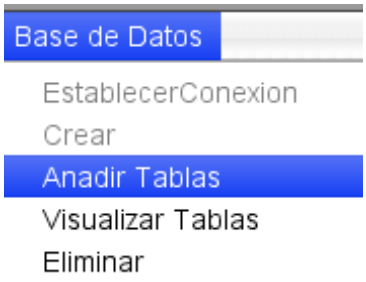


Figura 43: Pantalla TOOLWEB Menú Base de Datos opción Añadir Tablas

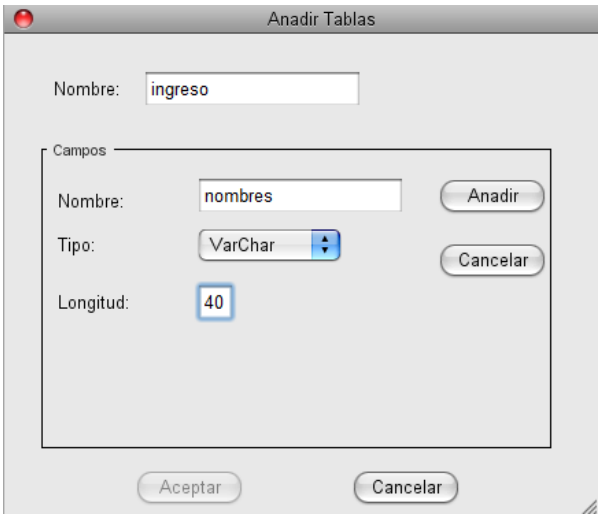


Figura 44: Pantalla Ventana Añadir Tablas

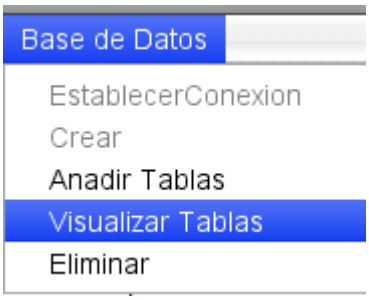


Figura 45: Ventana TOOLWEB Menú Base de Datos opción Visualizar Tablas

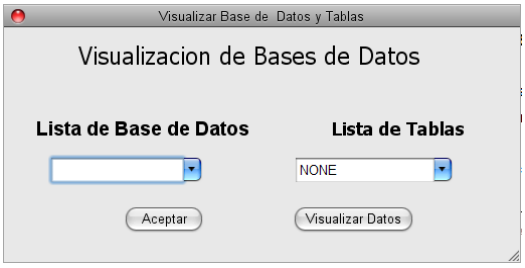


Figura 46: Pantalla Ventana Visualizar Base de Datos y Tablas

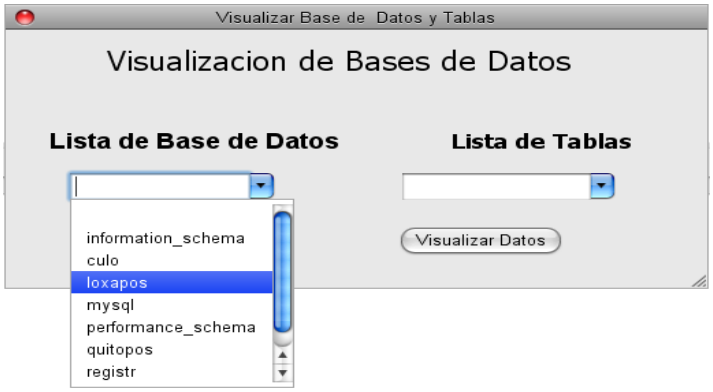


Figura 47: Pantalla Visualización de la lista de Base de Datos

Figura 48: Pantalla Visualización de la lista de tablas de la Base de Datos

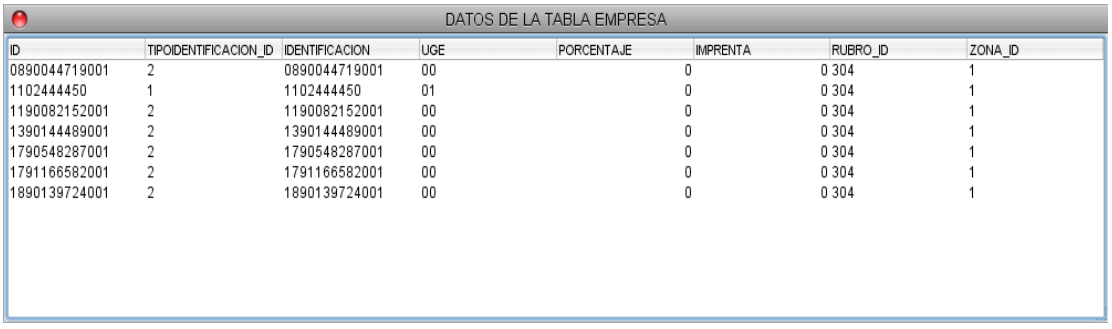


Figura 49: Ventana Datos de la tabla seleccionada

Nombre:	Crear Tablas
Actores:	Usuario
Propósito:	Permitir añadir tablas en una base de datos con sus respectivos campos.
Visión General:	El usuario tiene creada la base de datos.
Tipo:	Primario Esencial
Referencia:	RF018, RF019.

Curso Típico de Eventos:

1. El usuario elige la opción **Añadir Tablas** del menú **Base de Datos** en la ventana **TOOLWEB.**
2. El sistema muestra la ventana **AÑADIR TABLAS**.
3. El usuario ingresa el nombre de la tabla en el campo de texto **Nombre**.
4. El usuario selecciona la opción **Añadir Campos**.
5. El sistema presenta el panel **Campos**.
6. El usuario ingresa el nombre del campo de la tabla en el campo de texto **Nombre** en el panel **Campos**.
7. El usuario selecciona el tipo del campo de la tabla en la lista desplegable **Tipo** en el panel **Campos**.
8. El usuario selecciona la opción **Añadir** en el panel **Campos**.
9. El usuario selecciona la opción **Aceptar**.
10. El sistema valida los datos introducidos.
11. El sistema crea la tabla de datos en la base de datos con los respectivos campos.
12. El usuario elige la opción **Visualizar Tablas** del menú **Base de Datos** en la ventana **TOOLWEB.**
13. El sistema presenta la ventana **VISUALIZAR BASE DE DATOS Y TABLAS.**
14. El usuario elige una base de datos de la lista desplegable **ListaBD.**
15. El sistema carga las tablas de la base de datos respectiva en la lista desplegable **ListaBD.**
16. El usuario elige una tabla de la base de datos seleccionada anteriormente, de la lista desplegable **ListaTablas.**

17. El usuario selecciona la opción **Visualizar Datos.** **18.** El sistema presenta los registros de la tabla seleccionada en la ventana **DATOS DE LA TABLA.** **19.** El usuario selecciona la opción **Aceptar** de la ventana **VISUALIZAR BASE DE DATOS Y TABLAS.**
Curso Normal de Eventos:
A. EL usuario selecciona la opción **Cancelar** en el panel **Campos** de la ventana **AÑADIR TABLAS**. A5. El sistema oculta el panel **Campos** de la ventana **AÑADIR TABLAS**. A6. El use case vuelve al paso 9. **B.** EL usuario selecciona la opción **Cancelar** de la ventana **AÑADIR TABLAS**. B2. El cierra la ventana **AÑADIR TABLAS**. B3. El sistema regresa a la ventana **TOOLWEB.**

Tabla 13: Descripción de casos de Usos Crear Tablas

ELIMINAR BASE DE DATOS

Figura 50: Pantalla Ventana TOOLWEB Menú Base de Datos opción Eliminar

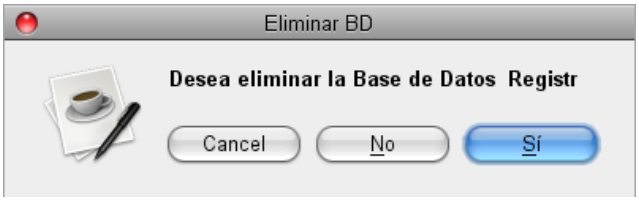


Figura 51: Pantalla Ventana Eliminar Base de Datos

Nombre:	Eliminar Base de Datos
Actores:	Usuario
Propósito:	Permitir eliminar una base de datos.
Visión General:	El usuario tiene creada la base de datos.
Tipo:	Primario Esencial
Referencia:	RF023.

Curso Típico de Eventos:

1. El usuario elige la opción **Eliminar** del menú **Base de Datos** en la ventana **TOOLWEB.**
2. El sistema muestra un mensaje de confirmación **"Desea Eliminar la base de datos _nombreBD".**
3. El usuario selecciona la opción **Si** del mensaje **Eliminar BD**.
4. El sistema elimina la base de datos de la aplicación web.
5. El sistema presenta un mensaje de confirmación **"La base de datos ha sido correctamente eliminada".**
6. El usuario selecciona la opción **Aceptar** del mensaje de confirmación.

Curso Normal de Eventos:

A. El usuario selecciona la opción **No** del mensaje de confirmación **"Desea Eliminar la base de datos _nombre".**

A2. El sistema no eliminar la base de datos.

A3. El sistema presenta la ventana **TOOLWEB.**

B. El usuario selecciona la opción **Cancel** del mensaje de confirmación **"Desea Eliminar la base de datos _nombre".**

B2. El sistema cierra la ventana del mensaje de confirmación**.**

B3. El sistema presenta la ventana **TOOLWEB.**

Tabla 14: Descripción de casos de Usos Eliminar Base de Datos

4. Modelo de la Interacción.

4.1 Diagramas de Secuencia.

CREAR APLICACIÓN WEB

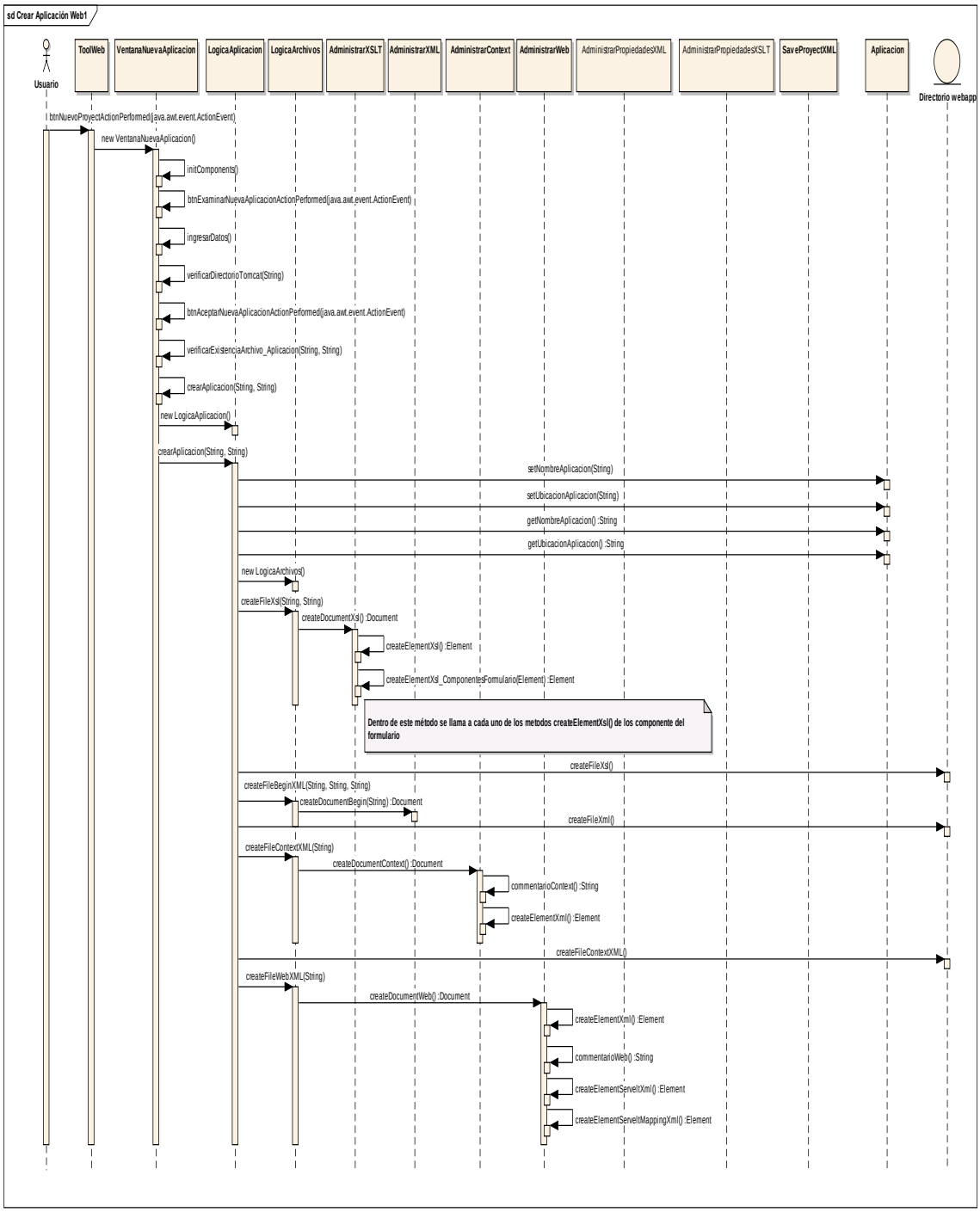


Figura 52a: Diagramas de Secuencia Crear Aplicación Web

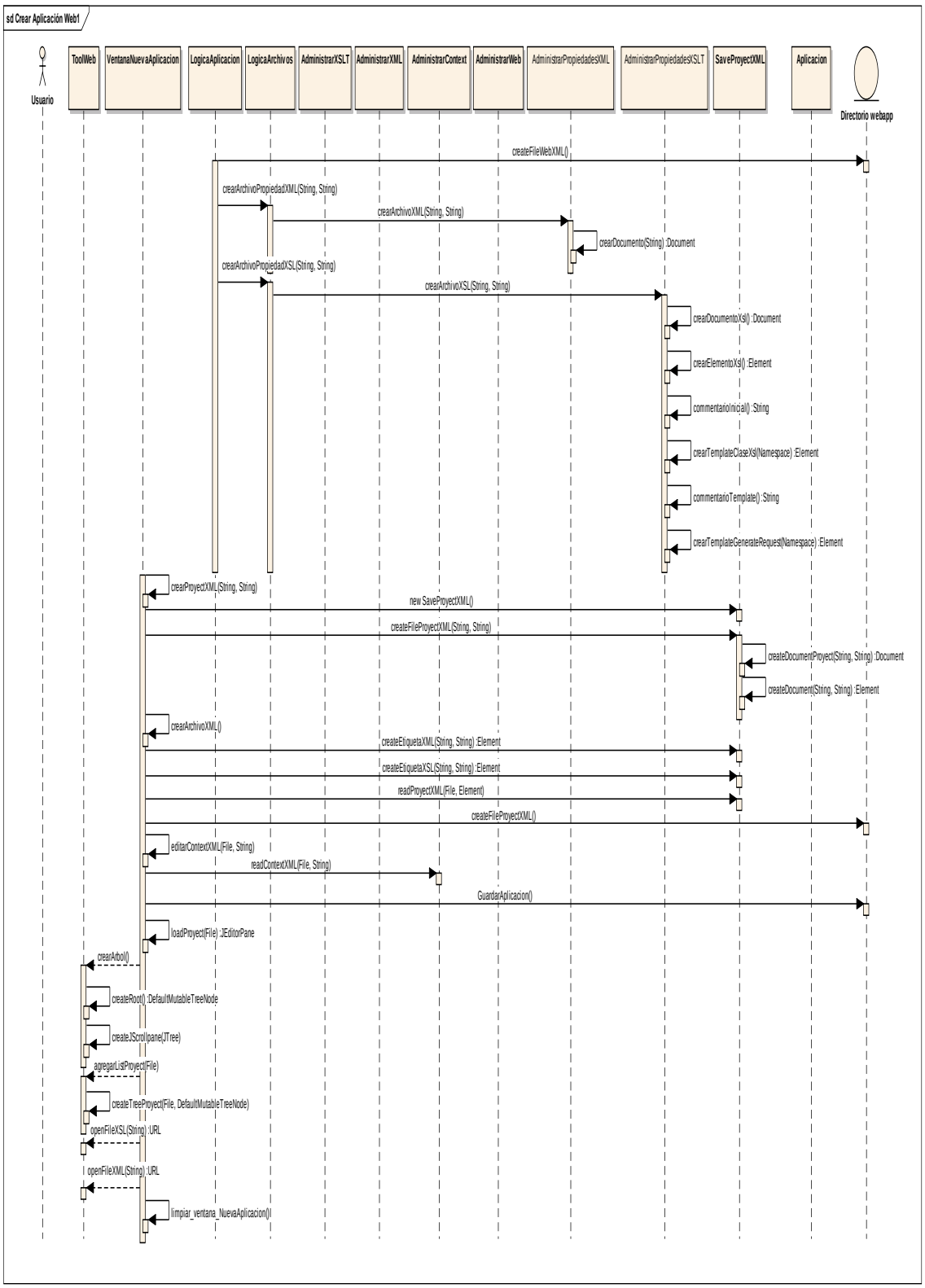


Figura 52b: Diagramas de Secuencia Crear Aplicación Web

CREAR ARCHIVOS

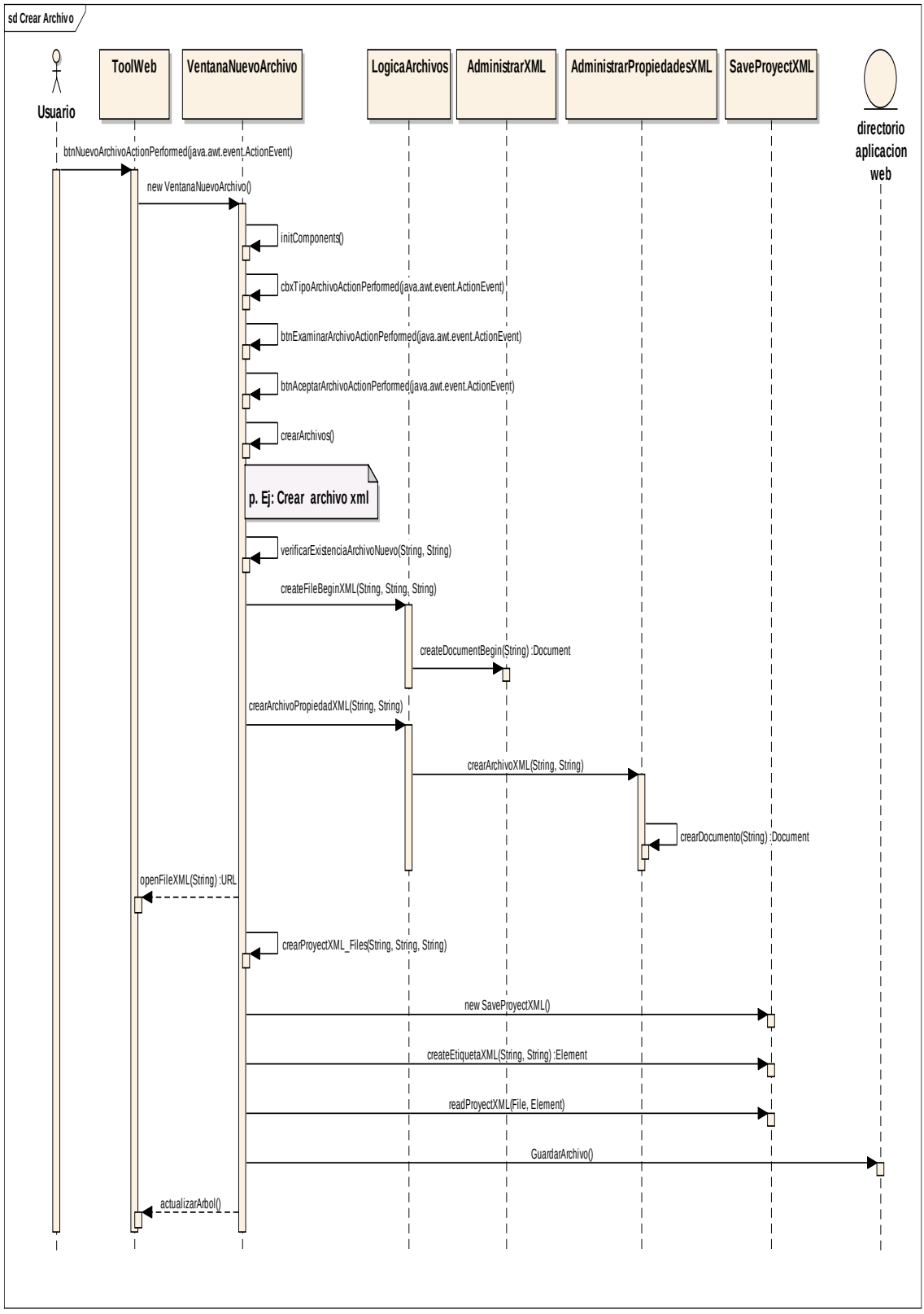


Figura 53: Diagramas de Secuencia Crear Archivos

CREAR FORMULARIOS

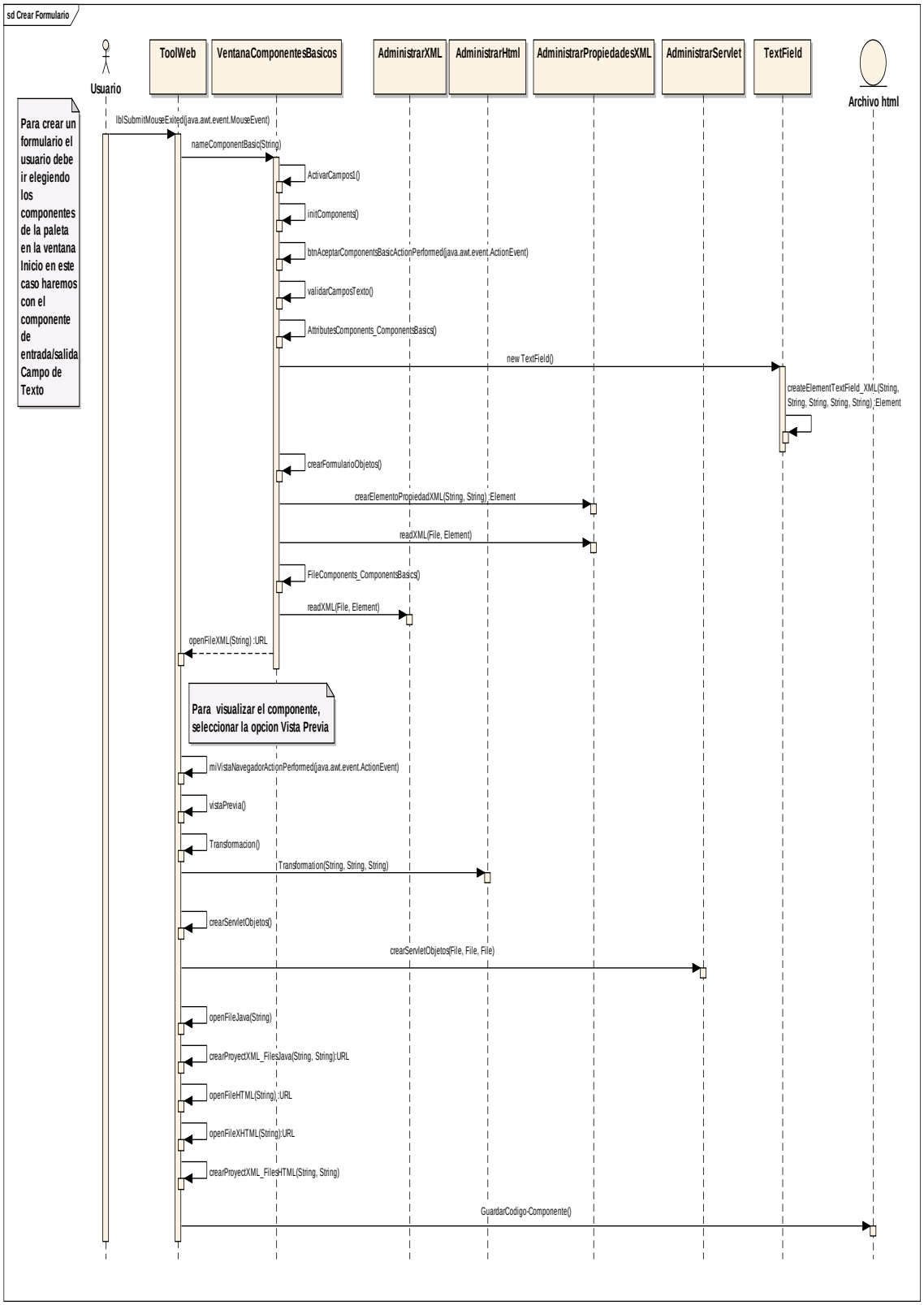


Figura 54: Diagramas de Secuencia Crear Formularios

CREAR BASE DE DATOS

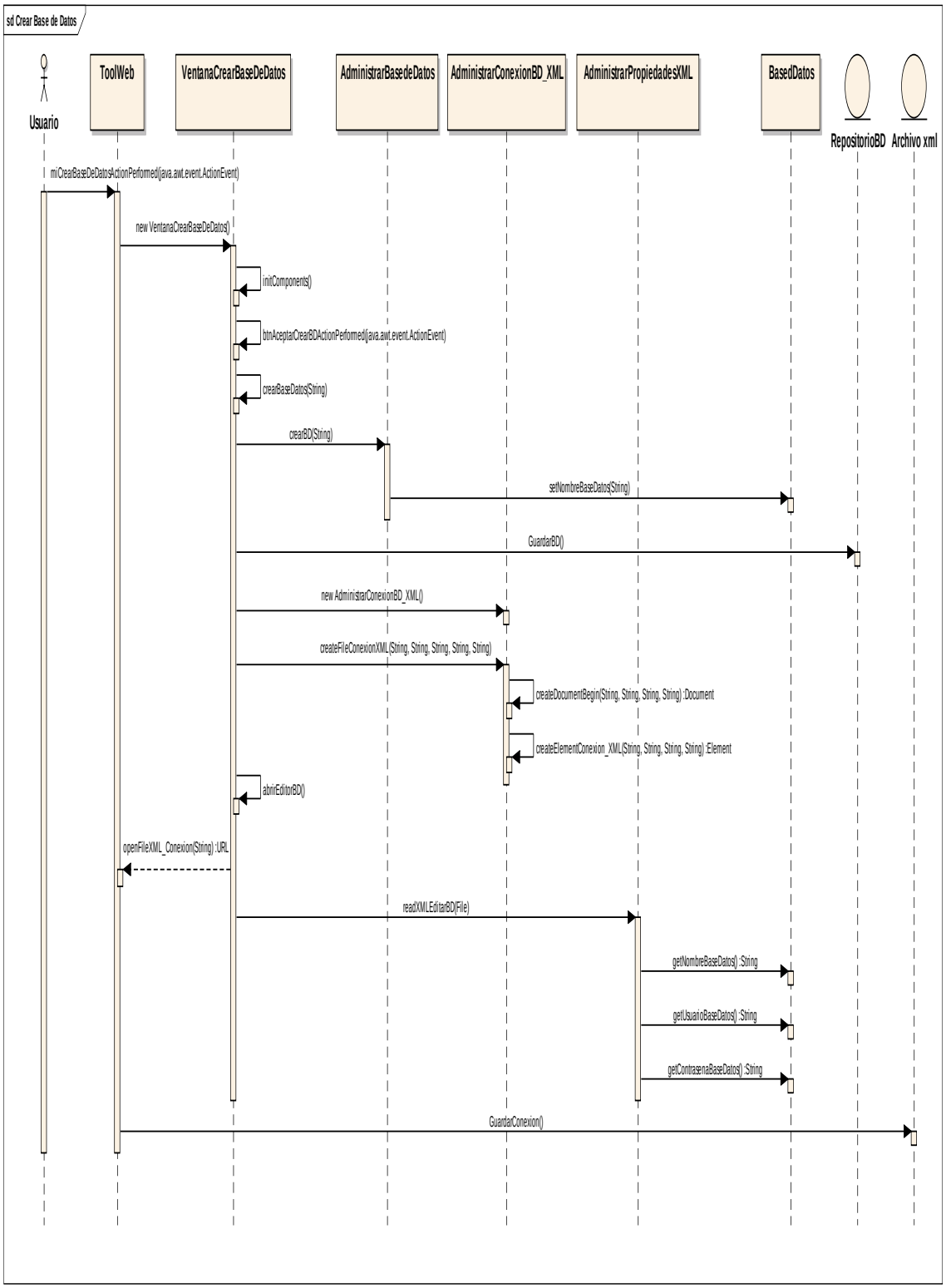


Figura 55: Diagramas de Secuencia Crear Base de Datos

CREAR TABLAS

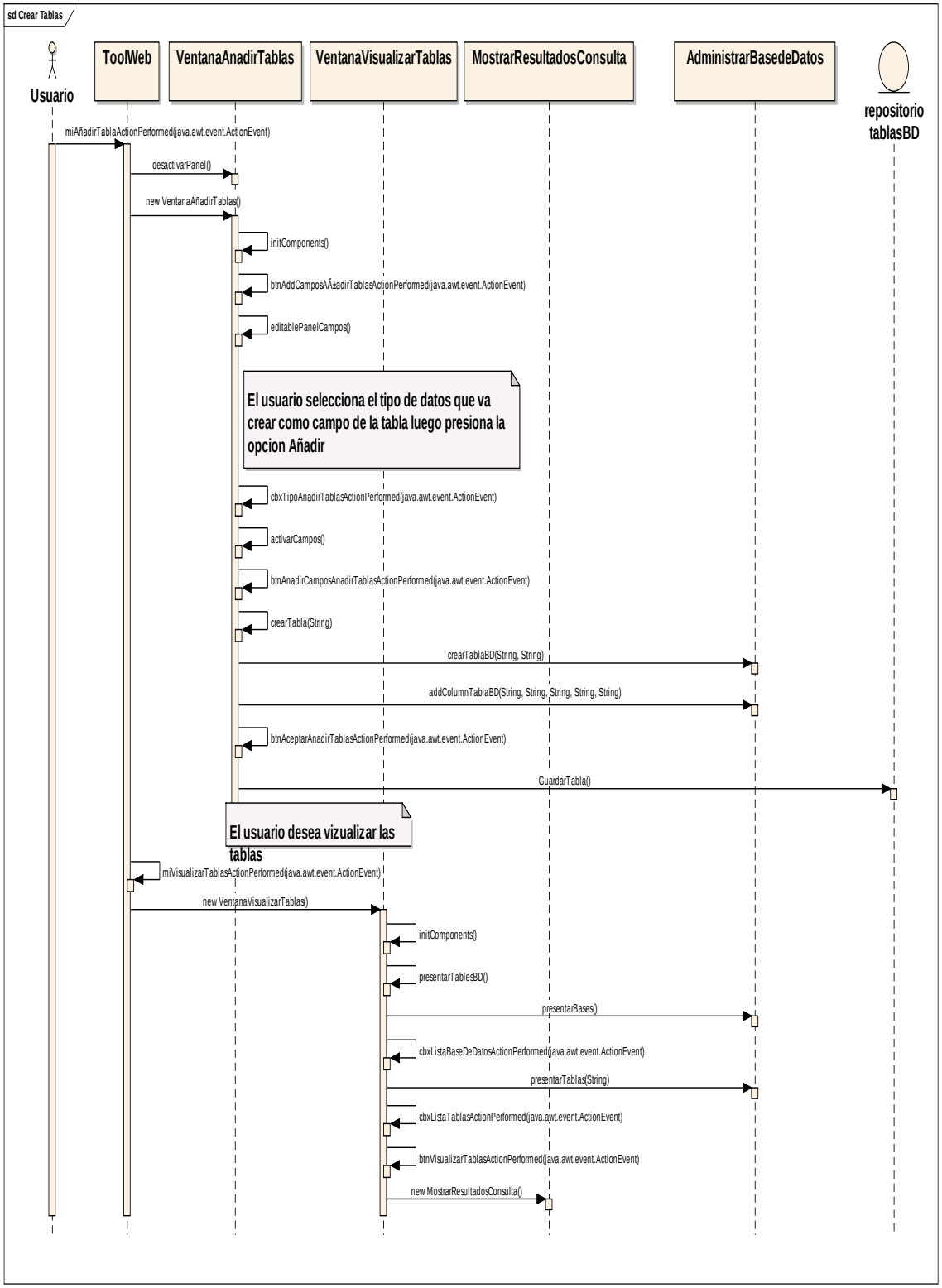


Figura 56: Diagramas de Secuencia Crear Tablas

ELIMINAR BASE DE DATOS

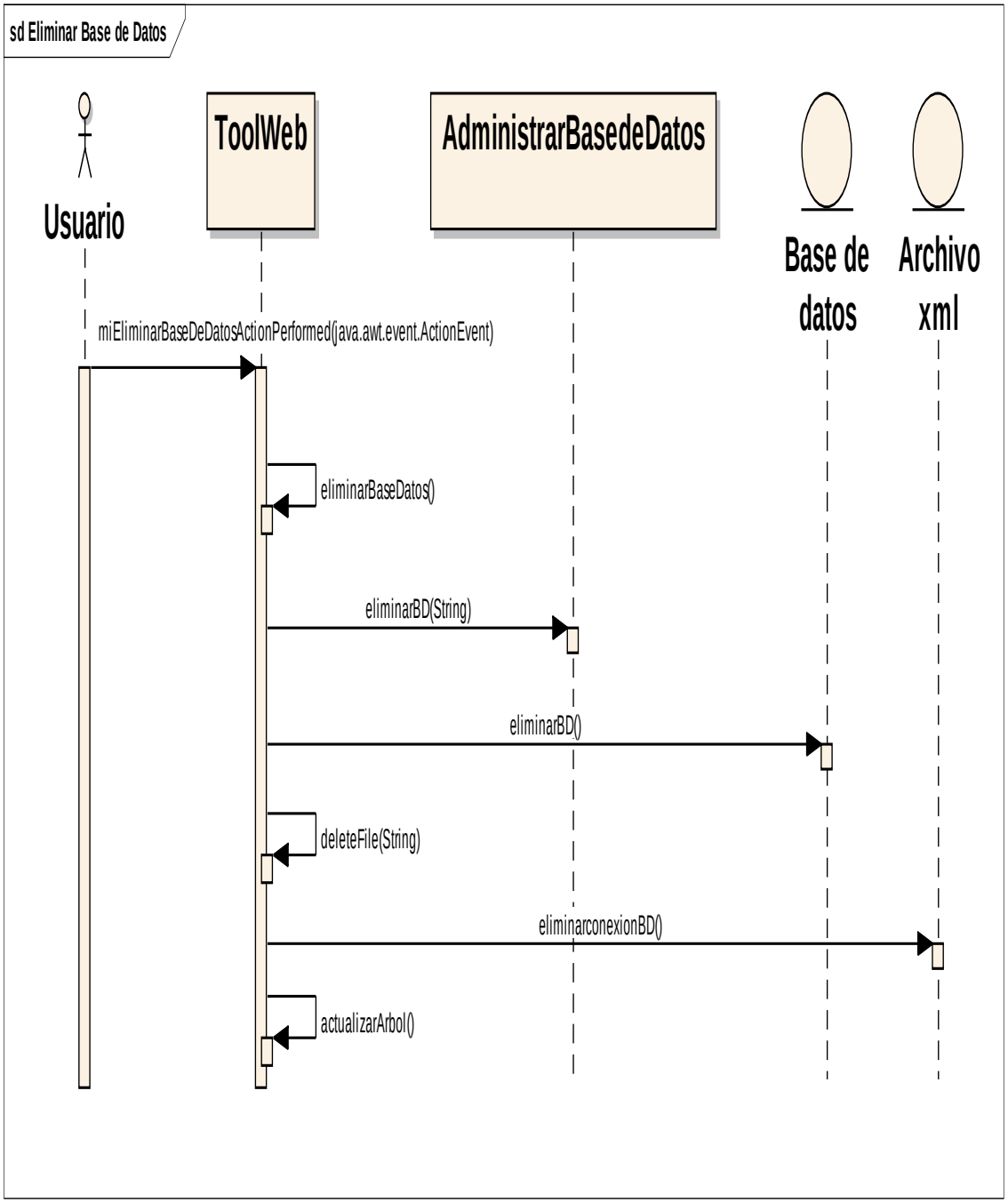


Figura 57: Diagramas de Secuencia Eliminar Base de Datos

4.2 Diagramas de Clases.

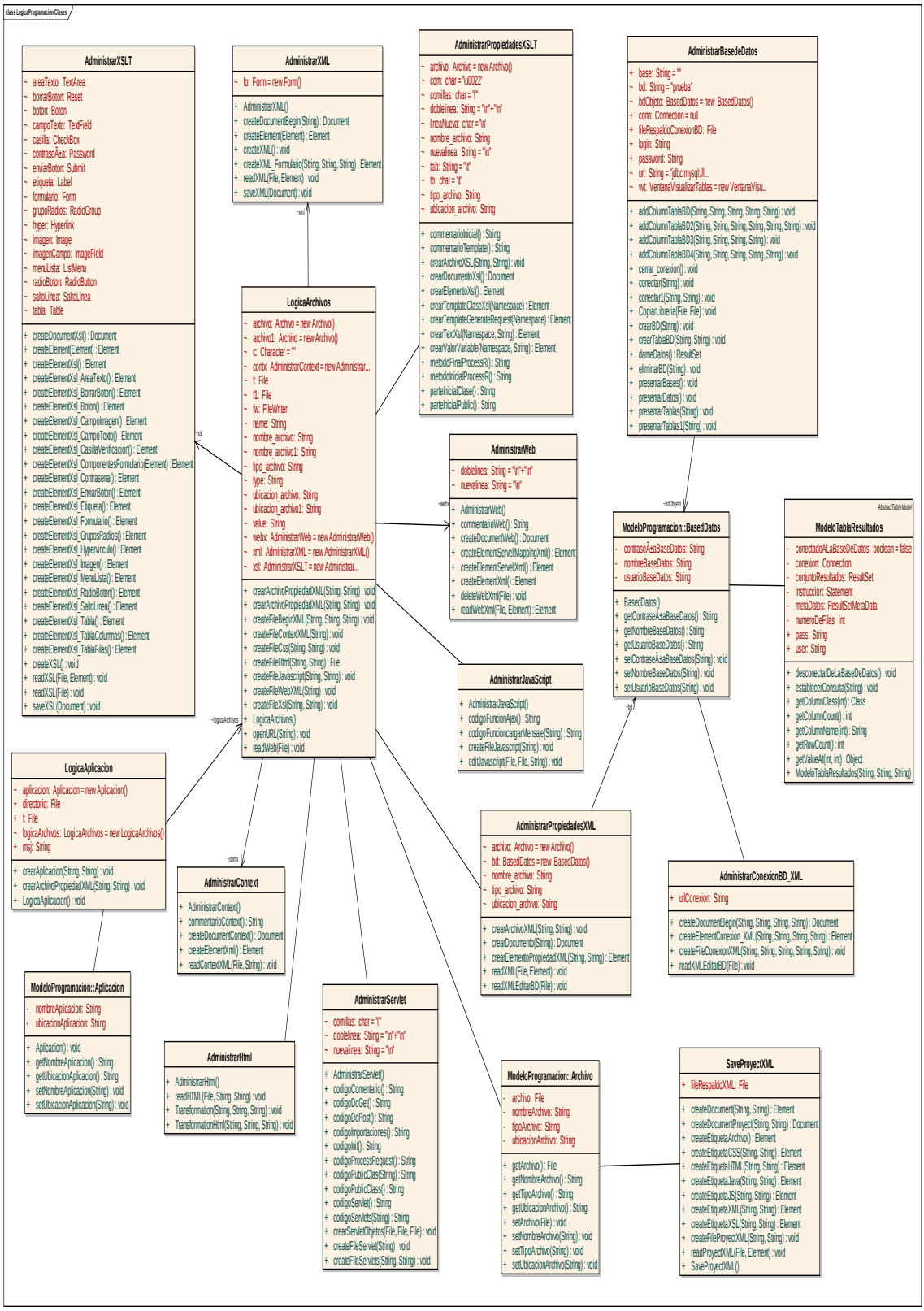


Figura 58: Diagramas de Clases

4.3 Diagramas de Paquetes.

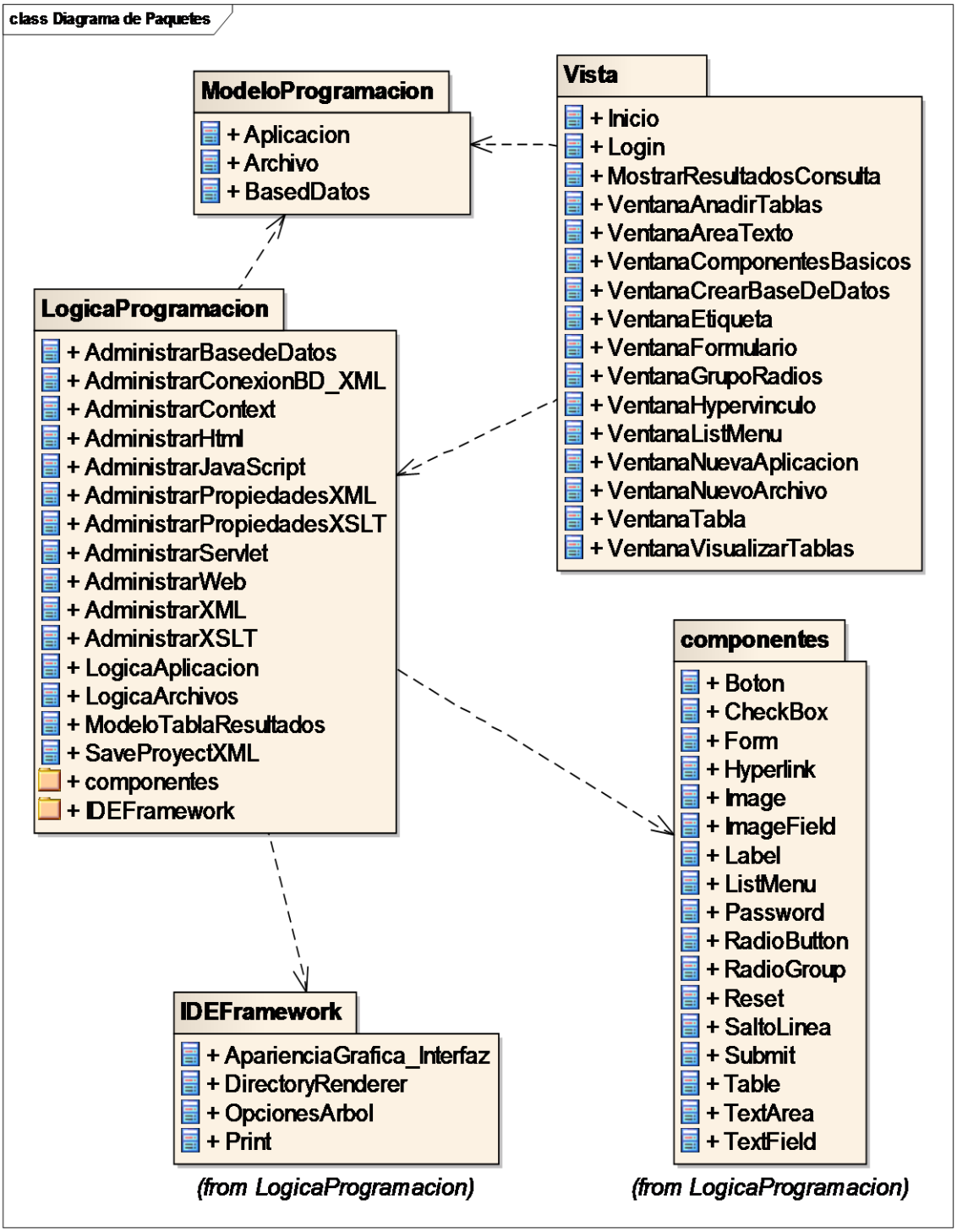


Figura 59: Diagramas de Paquetes

4.4 Diagramas de Componentes.

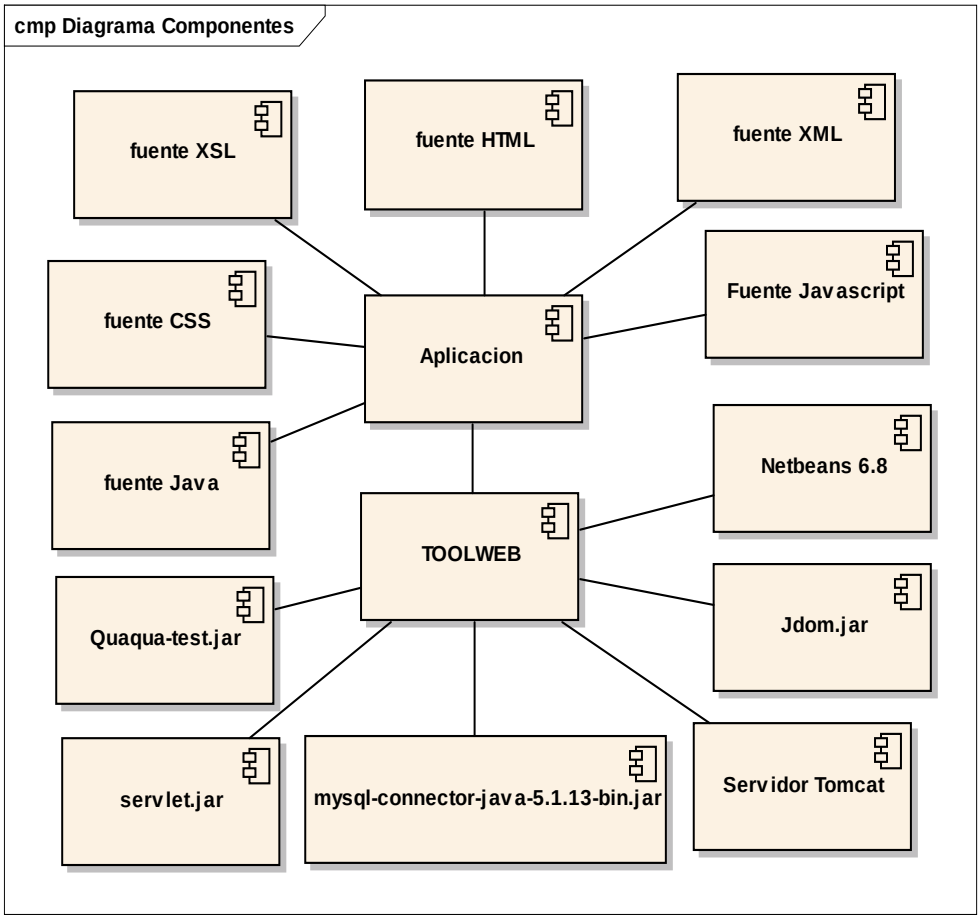


Figura 60: Diagramas de Componentes

COMPONENTE	FUNCIÓN
Aplicación	Directorio creado con los archivos xml, xsl, javascript, css, html.
Fuente HTML	Código generado de los componentes
Fuente XML	Código generado de los componentes con sus atributos
Fuente XSL	Código para generar componentes
Fuente CSS	Código para dar estilo a la página web
Fuente JAVASCRIPT	Código de las funciones para el comportamiento de la página web
Fuente JAVA	Código generado clase servlet con la conexión

	a la base de datos y con los parámetros del formulario.
Quaqua-test.jar	Librería para la interfaz grafica
Servlet.jar	Librería para hacer conexión con los servlets en el servidor
Jdom.jar	Librería para generar el código de los archivos xml, xsl y html
Netbeans 6.8	Herramienta que se utilizó para construir el Framework TOOLWEB
Servidor Tomcat	Servidor de las aplicaciones web
mysql-connector-java-5.1.13-bin.jar	Librería que permite la administración de la Base de Datos MySql.

Tabla 15: Diagramas de Componentes

4.5 Diagramas de Despliegue.

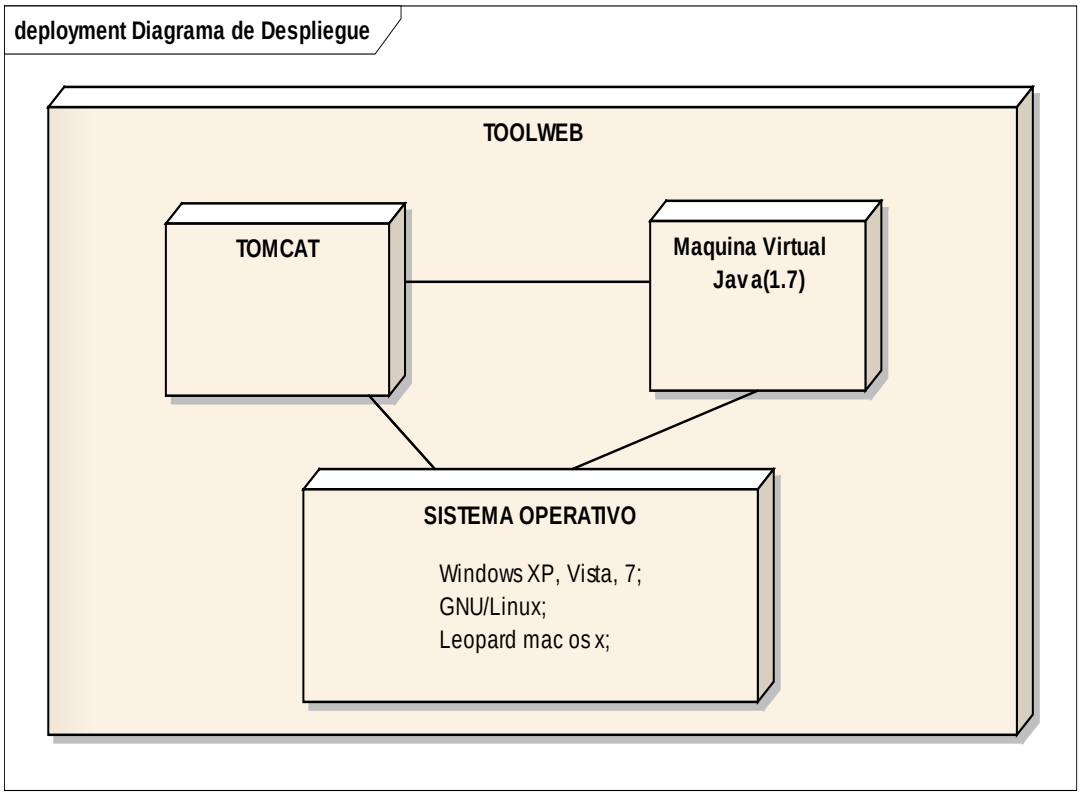


Figura 61: Diagramas de Despliegue

5. Plan de Validación.

Para realizar el plan de validación es necesario tomar en cuenta los siguientes factores:

Recursos Humanos:

Determinar una muestra de usuarios, para ello hemos creído conveniente hacerlo con los alumnos de sexto y octavo módulo de la carrera de Ingeniería en Sistemas de la Universidad Nacional de Loja, los mismos que ya tienen un conocimiento acerca de las Aplicaciones Web, de esta manera nos ayudara a mejorar nuestra herramienta.

Tiempo:

El tiempo estimado para realizar las pruebas se determinó considerando las funciones, procesos y tecnologías que intervienen en nuestra herramienta TOOLWEB y la disponibilidad de los módulos de sexto y octavo de la carrera de Ingeniería en Sistemas quedando determinado el tiempo de 18 horas para la ejecución de las mismas.

Recodificación

Luego de realizar las pruebas, y de haber obtenido la información necesaria se realizara una recodificación si se cree conveniente para el mejoramiento la funcionalidad del Framework.

6. Pruebas de Validación.

6.1 Tipos de Pruebas.

6.1.1 Pruebas de Funcionalidad.

Dentro de estas pruebas se realizaron los siguientes procesos:

- ✓ Ejecución del Framework: el tiempo estimado en ejecutarse es de 10 a 15 s dependiendo de las características del ordenador.

- ✓ Creación de las aplicaciones: se realizan de una manera rápida y sencilla, esto incluye la creación de directorios y archivos estos últimos se crean con código fuente necesarios para su funcionalidad.

- ✓ Creación de componentes: el usuario cuenta con una paleta de componentes básicos y necesarios para cada aplicación, este proceso se realiza de forma rápida y dinámica.

- ✓ Vista Previa: el usuario cuenta con una opción de visualizar el diseño de su página web.

- ✓ Ejecución de la Aplicación Web: esta opción realiza un proceso interno en el cual se crean los archivos JavaScript, java, css, generando el código fuente respectivo, necesarios para su aplicación.

- ✓ Establecer Conexión y Creación de Bases de Datos: la herramienta permite validar el usuario y contraseña del motor de base de datos MySql, para luego poder crear las bases de datos.

- ✓ Crear y Visualizar Tablas: luego que se crea la base de datos se puede añadir tablas con sus respectivos campos para luego poder visualizarlas.

- ✓ Eliminar Base de Datos: el usuario podrá eliminar la base de datos que fue creada con la herramienta.

6.1.2 Pruebas de Aceptación.

Luego de que se dio la capacitación adecuada sobre la funcionalidad de nuestro Framework, la aceptación por parte de los usuarios es satisfactoria debido a que es una herramienta nueva la cual la podrá ser utilizada para realizar sus aplicaciones web de una forma dinámica.

La herramienta TOOLWEB tuvo una aceptación en cuanto a los siguientes aspectos:

- ✓ Interfaz Gráfica: las pantallas son de forma amigable y dinámica para realizar los procesos que realiza la herramienta.

- ✓ Generación de Código: proceso que contribuye al desarrollo de las aplicaciones web en su parte inicial.

6.1.3 Pruebas de Usabilidad.

Con respecto a estas pruebas según los usuarios; el Framework es una herramienta que tiene una interfaz gráfica de fácil utilización; las aplicaciones, los componentes y las bases de datos se crean de manera rápida.

TOOLWEB tendrá una usabilidad para los desarrolladores de aplicaciones web.

6.2 Resultados de Validación.

Para la validación de la herramienta TOOLWEB se dio una capacitación que se inició en el mes de Marzo del 2012, a los alumnos de los módulos de sexto y octavo de la carrera de Ingeniería en Sistemas, tomando como muestra del grupo a los alumnos que disponían de su ordenador portátil siendo un total del 55% del alumnado mencionado anteriormente, los mismos que estuvieron a cargo por los docentes: Ing. Wilman Chamba e Ing. Hernán Torres, respectivamente.

Según los resultados de las pruebas realizadas podemos decir que la herramienta de desarrollo de aplicaciones web es válida y eficiente para que pueda ser utilizada por los usuarios. (Ver Anexo 3).

F. CONCLUSIONES.

Podemos considerar que se ha logrado cumplir con todos los objetivos planteados en el presente trabajo de desarrollo de tesis concluyendo de la siguiente manera:

- ✓ Para llevar a cabo este trabajo se recopilo información exhaustiva de procesos de desarrollo, áreas de conocimiento de ingeniería de software, metodologías ágiles y prácticas de desarrollo dentro de la informática.

- ✓ Obtenida la información necesaria se logró realizar la construcción del Framework al mismo que se lo ha denominado TOOLWEB. Esta herramienta se encuentra implementado por el patrón de diseño Modelo Vista Controlador (MVC), cumpliendo de esta manera con uno de los objetivos planteados.

- ✓ Las aplicaciones generadas con TOOLWEB implementan el patrón de diseño Modelo Vista Controlador (MVC), las misma que se encuentran estructuradas de la siguiente manera: el modelo son dos archivos **xsl**, uno para en el cual está toda la información de los componentes de interfaz gráfica GUI que se pueden construir y en el otro la información de los componentes que serán de entrada y salida de datos la vista son los archivos **html** y **css** que se crean para la aplicación; y el controlador son los archivos **javascript** y java (**servlet**) que permiten las operaciones con los datos.

- ✓ Dentro de las aplicaciones web desarrolladas con la herramienta TOOLWEB se puede establecer una conexión Asincrónica con el Objeto XMLHttpRequest que se crea dentro de un archivo JavaScript en un método denominado Ajax permitiendo la comunicación con el servidor Tomcat.

- ✓ En la herramienta TOOLWEB se puede crear las bases de datos con sus respectivas tablas y campos necesarios que el usuario cree conveniente para que puedan acceder desde sus aplicaciones.

- ✓ Se desarrolló una aplicación de ejemplo denominada Registro que permite validar los usuarios registrados y en el caso de no constar en la base de datos

tiene la opción de registrarse, de esta manera demostramos la funcionalidad de la herramienta TOOLWEB.

- ✓ Esta herramienta es una alternativa de desarrollo para los usuarios que están encaminados en la generación de aplicaciones web 2-0. Utilizar Ajax en las aplicaciones web nos ha permitido ahorrar tiempo en el proceso de envío y recepción de datos creando paginas dinámicas en el lado del servidor Tomcat.

G. RECOMENDACIONES

Terminado el siguiente trabajo de tesis podemos dar las siguientes recomendaciones:

- ✓ Para que la herramienta funciones de manera adecuada se debe seguir un orden en la creación de las aplicaciones web, el cual está especificado en el manual del usuario-programador.

- ✓ Para que la herramienta TOOLWEB funcione en distintas plataformas (Windows, Linux, Mac OSX), es necesario tener la versión de java 1.7, realizar configuraciones previas para cada uno de ellos; por ejemplo en Windows se debe configurar las variables de entorno: PATH se modifica añadiendo la ruta del directorio de java jdk/bin.

- ✓ Que los usuarios-programadores tengan conocimientos sobre la web 2.0, los mismos que les ayudara con un mejor manejo de la herramienta. Además queda abierta la posibilidad para que los desarrolladores puedan implementar nuevas funcionalidades como: ampliar la lista de componentes que se puedan utilizar en las aplicaciones web; permitir el acceso a otras base de datos que ayuden al mejoramiento de la misma, quedando como nueva alternativa de desarrollo de tesis.

- ✓ Utilizar el patrón de Diseño de software Modelo Vista Controlador (MVC), ya que permite a los programadores separar las aplicaciones en tres capas, ayudando a tener una mejor distribución de las mismas, facilitando realizar cambios en cada una de estas capas.

H. BIBLIOGRAFIA Y REFERENCIAS.

RECURSOS DE INTERNET

1. Marius Zaharia, Introduccion a XSL/centro de desarrolladores, 22 de agosto 2005, http://www.adobe.com/es/devnet/dreamweaver/articles/xsl_overview.html, [fecha de consulta: agosto 2010].

2. ADR Infor S.L., Estructura y despliegue de aplicaciones web, [internet], 2004, < http://www.adrformacion.com/curso/javaservidor/leccion3/estructura_servidor_web.htm>, [Fecha de consulta 10 noviembre 2009].

3. Ajpdsoft, Definición Tomcat, http://www.ajpdsoft.com/modules.php?name=Encyclopedia&op=content&tid=769, [fecha de consulta: agosto 2010].

4. ALEGSA, Definición de Formulario web - ¿qué es Formulario web?,[internet], < http://www.alegsa.com.ar/Dic/formulario%20web.php>, [Fecha de consulta octubre 2008].

5. Unijimpe, Introducción a AJAX, 25 diciembre 2007, http://blog.unijimpe.net/introduccion-a-ajax/, [fecha de consulta: agosto 2010].

6. ZonaClic, ¿Qué es XML?, [internet], http://clic.xtec.cat/es/jclic/xml.htm, [fecha de consulta: agosto del 2010].

7. comp.unanleon.edu.ni, java_c8, http://www.comp.unanleon.edu.ni/u/aaltamirano/java_c8.pdf, [fecha de consulta: agosto 2010].

8. Bulmaro Noguera, Qué es AJAX, [internet], 5 de julio 2011, http://culturacion.com/2011/07/que-es-ajax/, [fecha de consulta: noviembre del 2011].

9. Centro de Desarrollo Territorial Holguín UCI, Servidor Tomcat, Junio 2011, http://www.ecured.cu/index.php/Servidor_Tomcat, [fecha de consulta: 20 mayo 2012].

10. Hooping.net, glossary/xhtml, 2008, http://www.hooping.net/glossary/xhtml-137.aspx, [fecha de consulta_ agosto 2010].

11. José María de Pereda (JMPereda), ¿Que es una Aplicacion Web?, [internet], agosto 24, 2007, <http://jmpereda.wordpress.com/2007/08/24/definiendo-la-plantilla/>, [Fecha de Consulta: 15 octubre 2008].

12. Jorge A. Saavedra Gutierre, Razones para usar AJAX/El mundo informatico, 22 junio 2007, http://jorgesaavedra.wordpress.com/2007/06/22/10-razones-para-usar-ajax/, [fecha de consulta: agosto 2010].

13. latascadexela.es, Java y XML: JDOM, 8 de agosto 2008, http://www.latascadexela.es/2008/08/java-y-xml-jdom.html, [fecha de consulta: julio 2009].

14. Librosweb.es, Introducción a JavaScript, http://www.librosweb.es/javascript/capitulo1.html, [fecha de consulta: agosto 2010].

15. Christian Van Der Henst S. para Maestros del web, ¿Qué es la web 2.0?, [internet], Noviembre 2005, http://www.maestrosdelweb.com/editorial/web2/, [fecha de consulta agosto 2008].

16. Mundogeek.net, JDBC, 27 enero 2007, http://mundogeek.net/archivos/2007/01/27/jdbc/, [fecha de consulta: enero 2011].

17. NetBeans IDE, NetBeans IDE 7.1.2, modificado 2012, http://www.netbeans.org/community/releases/71/, [fecha de consulta: 20 mayo 2012].

18. O'Reilly, Java and XSLT, septiembre 2001, http://repository.mdp.ac.id/ebook/oreilly-books/OReilly.Java.&%20XSLT.pdf, [fecha de consulta: mayo del 2012].

19. Srbyte, Cambia las apariencias de tus aplicaciones java, 2 octubre 2008, http://www.srbyte.com/2008/10/cambia-la-apariencia-de-tus.html, [fecha de consulta: agosto del 2010].

20. *Apache Software Foundation*, Apache Tomcat 7, [internet], 31 de Marzo 2012, http://tomcat.apache.org/tomcat-7.0-doc/,[fecha de consulta: mayo del 2012].

21. W3C.es, Guia Breve de CSS, http://www.w3c.es/Divulgacion/GuiasBreves/HojasEstilo, [fecha de consulta: agosto 2010].

I. ANEXOS

ANEXO 1 ANTEPROYECTO

UNIVERSIDAD NACIONAL DE LOJA

Área de la Energía, las Industrias y los Recursos Naturales No Renovables

Tema:

"Construcción de un Framework para desarrollo de aplicaciones Web 2.0 con acceso a base de datos MySql".

Autores:

Oscar Rafael Guamán Gutiérrez

Lady Jackeline Rosillo Cumbicus

Fecha:

28 de Noviembre 2007

Loja - Ecuador

2007

DISEÑO DEL ANTEPROYECTO DE INVESTIGACIÓN

1. PLANTEAMIENTO DEL PROBLEMA

TEMA:

Construcción de un Framework para desarrollo de aplicaciones Web 2.0 con acceso a base de datos.

SITUACION PROBLEMÁTICA

Antecedentes

Desde hace muchos años el avance tecnológico ha tomado un importante interés, para las diferentes empresas comerciales, industriales y de servicios, es así que la sociedad se ha servido de ella para obtener información con el fin de poder tener un mejor nivel de conocimiento, emergiéndose en un marco de competitividad que le ha permitido satisfacer las necesidades en el entorno social que se desenvuelve.

En la actualidad el avance tecnológico está proporcionando un mejor desenvolvimiento en las actividades que se realizan dentro de las empresas, las mismas que contribuyen para un mejor desarrollo de nuestra sociedad.

Así mismo se han realizado herramientas que brindan un mejor aprendizaje para el desarrollo de aplicaciones. Lo que ha permitido que gran parte de la sociedad se beneficie de esta tecnología y en mayor parte las instituciones educativas y con mayor frecuencia en las de nivel superior.

La Universidad Nacional de Loja dentro de sus innovaciones tecnológicas requiere de nuevas herramientas para brindar una mejor comprensión y aprendizaje a los estudiantes de la carrera de Ingeniería en Sistemas. Consideramos que para satisfacer estas innovaciones es importante la creación de un Framework que permita desarrollar aplicaciones Web con el acceso a base de datos, con esto se pretende alcanzar que los estudiantes se incentiven en la investigación de las tecnologías utilizadas y en los nuevos avances tecnológicos y por ende obtendrán un mejor aprendizaje.

Se considera a los Frameworks como los Generadores de Aplicaciones que relacionan directamente con un dominio especifico, es decir con una familia de problemas relacionados, es por eso que un "Software Framework" es un diseño re-usable de un sistema (o subsistema); expresado por un conjunto de clases abstractas y de modo que sus instancias colaboran para un tipo específico de software. Todos los frameworks de software son diseños orientados a objetos"

Para desarrollar un framework se debe pasar por tres etapas que son:

Análisis del dominio: que procura descubrir los requisitos del dominio y los posibles requerimientos futuros.

Diseño del framework: define las abstracciones de éste. Se modelan los puntos calientes y los puntos congelados (diagrama UML), y la extensión y la flexibilidad propuesta en el análisis del dominio se esboza en líneas generales.

Instantiación del framework: los puntos calientes del framework son implementados, generando un software del sistema. Es importante observar que cada uno de estas aplicaciones tendrá los puntos congelados del framework en común.

Las fases de desarrollo del framework son comparados con las tradicionales fases del diseño orientado a objetos, según se puede apreciar en la siguiente figura:

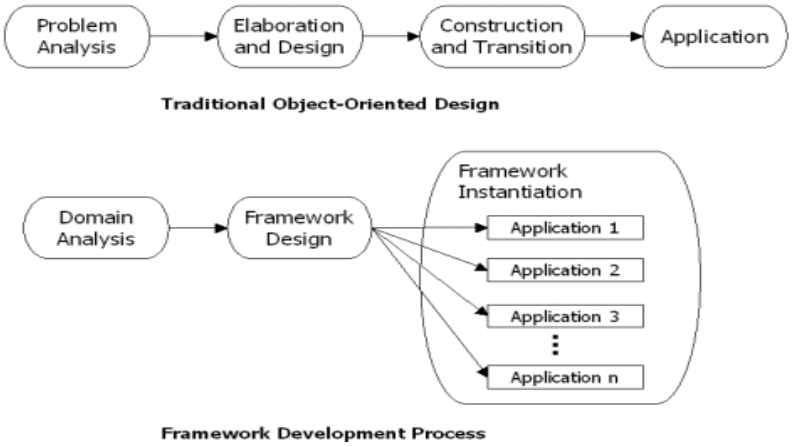


Figura 1. Proceso de Desarrollo del Frameworks[1]

El desarrollo tradicional Orientado al Objeto se diferencia del desarrollo del framework. En el desarrollo tradicional Orientado al Objeto, la fase de análisis del problema, también llamada inicio, estudia solamente los requisitos de un solo problema. En cambio, en el desarrollo del framework captura los requisitos para un dominio entero. Además, el resultado final del desarrollo orientado al objeto tradicional es una aplicación que es completamente ejecutable, mientras que muchas aplicaciones resultan a partir de la fase de "instantiación" del desarrollo del framework. La fase del "instantiación" abarca las fases de construcción y de transición del desarrollo tradicional. Así, la construcción y las fases separadas de la transición están presentes en cada uno de las instancias del framework. Para cada una de las instancias del framework hay un esfuerzo de implementación o puesta en práctica introducido por estas fases.

Problemática

Entender el proceso de desarrollo de aplicaciones Web en la mayoría de las herramientas de desarrollo es aun complejo de llevarlos a cabo satisfactoriamente y con el fin de reducir este proceso es cuando se propone realizar el framework que permitirá reducir el tiempo en la elaboración de aplicaciones con un kit de herramientas, además de esto otro propósito importante es ayudar al estudiante universitario a entender y aprender a utilizar las herramientas que permiten desarrollar aplicaciones Web utilizando la tecnología AJAX que es parte de la Web2.0 teniendo el acceso a la base de datos MySql; dando cumplimiento así a nuestros objetivos planteados al emprender en este proyecto de investigación.

Pretendemos utilizar herramientas que en nuestro transcurso universitario no constaban en un pensum de estudio, por ende no las conocemos pero que permiten resolver de una manera muy eficiente y que hoy por hoy son muy utilizadas para dar soluciones a innumerables problemas, creemos que el framework permitirá a los estudiantes universitarios utilizarlo al mismo e investigar sobre nuevas tecnologías utilizadas para su desarrollo, que es lo que pretendemos demostrar con nuestro proyecto de investigación.

En el trascurso de nuestra formación universitaria adquirimos conocimientos que nos servirá de base para el desarrollo de nuestro proyecto planteado, con este trabajo de investigación aportaremos a la enseñanza como material de estudio para que en la

unidad de aplicaciones Web sea utilizada nuestra herramienta con el fin de desarrollar pequeñas aplicaciones demostrativas a nuestros compañeros universitarios.

Para entender un poco más sobre los framework comenzaremos por definirlo:

Un framework en el contexto de la programación es un set de funciones o código genérico que realiza tareas comunes y frecuentes en todo tipo de aplicaciones (creación de objetos, conexión a base de datos, limpieza de strings, etc.). Esto brinda una base sólida sobre la cual desarrollar aplicaciones concretas y permite obviar los componentes más triviales y genéricos del desarrollo.

En general, los frameworks son construidos en base a lenguajes orientados a objetos. Esto permite una mejor modularización de los componentes y óptima reutilización de código. Además, en la mayoría de los casos un framework implementará uno o más patrones de diseño de software que aseguren la escalabilidad del producto.

Un patrón de diseño es una metodología probada para resolver problemas comunes en el diseño de aplicaciones. Una convención, que facilita la comprensión de la arquitectura de la aplicación.

Hace ya mucho tiempo existen frameworks que facilitan el desarrollo de aplicaciones en diversos ambientes y lenguajes de programación. En el mundo de Java existe Struts, un ambiente muy estable y poderoso, y la iniciativa .NET de Microsoft es una especie de súper-framework inspirado en Java. Sin embargo, la complejidad de estos ambientes ha relegado el uso de frameworks al desarrollo de aplicaciones grandes y costosas.

En el ámbito del desarrollo para la Web, los patrones de diseño más utilizados son aquellos que se centran en separar la presentación (páginas html, css) de la lógica o backend. Esto porque un típico equipo de desarrollo consiste en programadores por un lado y diseñadores por el otro. Separando efectivamente las tareas de cada uno mediante una arquitectura estándar comprendida por todos -un patrón de diseño- facilita enormemente el trabajo del equipo; de estos patrones, el más popular es MVC (Modelo Vista Controlador), muy conocido en el mundo de Java.

Como su nombre lo dice, MVC consiste en separar lo mejor posible las capas de Modelo (los objetos que interactúan con la base de datos y efectúan los procesos pesados o "lógica de negocios"), la Vista (la presentación final de los datos procesados al cliente, comúnmente en formato HTML) y el Controlador (la capa que se encarga de recibir el input del usuario, delegar el trabajo a los Modelos apropiados e invocar las Vistas que correspondan).

Un frameworks es la forma en que un desarrollador decide solucionar sus proyectos, un ambiente de trabajo. En esencia un framework es la aplicación rigurosa de los principios básicos que hacen la diferencia entre un programa bien construido y uno malo: economía, modularización, separación de tareas.

Otro punto a considerar en el tratamiento de la problemática es la diferencia que hay que considerar en la siguiente frase "desarrollo del generador de Aplicaciones versus Desarrollo de aplicaciones", esto significa que los frameworks generan aplicaciones no por defecto sino que personalizando los requisitos particulares. Ellos en sí mismo no son aplicaciones sino que son construcciones más complejas. Es importante tener presente que el desarrollo de un framework será por lo menos tan costoso como el solo desarrollo de la aplicación. Uno debe analizar cuidadosamente la necesidad de flexibilizar un framework, evaluar los requisitos que se deben resolver para un cliente o futuro usuario, de lo contrario un framework no utilizable será creado innecesariamente.

Nuestro proyecto pretende dotar al estudiante universitario de una herramienta en la que pueda entender fácilmente la utilización de productos integrados para obtener mejor rendimiento en la elaboración de una pequeña aplicación con lo que queremos demostrar que podemos crear este tipo de herramientas que benefician mucho el aprendizaje e incentivar al estudiante a crecer en sus conocimientos.

PROBLEMA DE INVESTIGACIÓN

ENUNCIADO DE LA SITUACION PROBLEMÁTICA:

> **"**Las herramientas que se pueden utilizar en la actualidad para el desarrollo de aplicaciones Web no son en su mayoría de fácil uso y no tienen una interacción dinámica con el usuario**"**

Delimitación

El framework se lo realizará con la finalidad de que los estudiantes mejoren el aprendizaje en la materia de desarrollo Web en la carrera de Ingeniería en Sistemas de la Universidad Nacional de Loja. Debido a que hoy en día se ha visto conveniente la creación de una herramienta que permita desarrollar aplicaciones Web que tengan una mejor interfaz dinámica con el usuario, permitiendo el acceso a la base de datos. La información necesaria para nuestra investigación se la encuentra disponible en la Web y en los usuarios que utilizarán el Frameworks tales como los estudiantes y docentes de la carrera.

JUSTIFICACIÓN

Justificación Académica.

Durante mucho tiempo la principal preocupación de la U.N.L, ha sido proporcionar a sus alumnos, conocimientos importantes que permita investigar la realidad de su entorno y lo relacionado con el cultivo de la ciencia, a fin de obtener un producto académico satisfactorio con profesionales idóneos.

Es por tal motivo que como estudiantes de la carrera de Ingeniería en Sistemas y gracias a la formación académica que hemos obtenido, nos encontramos en total capacidad para realizar el tema de tesis "Construcción de un Framework para desarrollo de aplicaciones Web 2.0 con acceso a base de datos".

La realización del presente proyecto es un requisito fundamental para obtener el título de Ingenieros en Sistemas.

Justificación Técnica

Para el desarrollo de este proyecto se ha creído conveniente la utilización de la tecnología **Ajax** acrónimo de *Asynchronous JavaScript And XML* la misma que es parte de la Web2.0, esta tecnología será utilizada para el desarrollo de nuestro Frameworks; debido a que es una técnica de desarrollo Web para crear aplicaciones interactivas, además provee una mayor usabilidad, es válido en cualquier plataforma y navegador, beneficia las aplicaciones Web, no es difícil su utilización, es independiente del tipo de tecnología de servidor que se utilice y mejora la estética de la Web.

Para la fase de Diseño se utilizará la herramienta Enterprise Architect, la misma que nos permitirá la creación de los diferentes diagramas necesarios de una manera más sencilla y rápida. Además para la programación del framework se utilizara la herramienta de desarrollo NetBeans 5.5.

Justificación Operativa

El presente proyecto se realizara porque existe la información básica y necesaria para su desarrollo, además se aplicaran los conocimientos teóricos y prácticos adquiridos durante la formación académica, también nos permitirá obtener un mayor conocimiento y experiencia en el campo informático.

Una de las partes de esenciales para el desarrollo de nuestro trabajo informático es la disponibilidad de la información en la Web, permitiéndonos conocer la estructura y construcción de un Framework. Con el diseño de este proyecto se pretende lograr un mejor rendimiento dinámico en el desarrollo de aplicaciones Web.

Justificación Económica

Debido a que las herramientas seleccionadas para este proyecto se encuentran disponibles, por consiguiente generaría un mínimo gasto económico.

Para el desarrollo de la documentación del análisis, diseño, desarrollo e implementación se cuenta con el material necesario que implica gastos mínimos.

OBJETIVOS

Objetivo General

Construir un Framework para el desarrollo de aplicaciones Web 2.0 con acceso a base de datos Mysql.

Objetivos Específicos

- Implementar el patrón Modelo Vista Controlador en el diseño del Framework.
- Diseñar el Framework utilizando la tecnología AJAX que es parte de la Web 2.0.
- Diseñar los componentes básicos que ofrecerá el Framework.
- Permitir que las aplicaciones Web basadas en el Framework tengan el acceso a la base de Datos Mysql.
- Las aplicaciones se almacenaran en un archivo XML.
- Implementación del Framework.
- Crear una aplicación de Ejemplo basado en el Framework desarrollado.

2. MARCO TEÓRICO

Capítulo I. La Web2.0

1.1. Generalidades

1.2. La Web como plataforma

1.3. Tecnología que conforma la Web2.0

1.3.1 Ajax (Generalidades).

1.3.1.1 JavaScript

1.3.1.2 Xml

1.3.1.3 Html

1.3.1.4 CSS

1.4. Blogs

1.5. Wikis

Capítulo II: La Tecnología AJAX

2.1. Definición

2.2. El funcionamiento de AJAX.

2.3. Razones para utilizar AJAX

Capítulo III: Frameworks

3.1. Introducción

3.2. ¿Qué es un Framework?

3.3. ¿Qué es un Framework Web?

3.4. Arquitectura básica de la Web

3.5. Evolución de los Frameworks Web

3.5.1 La Web estática.

3.5.2 La interfaz CGI

3.5.3. Los Servidores de Aplicaciones Web

3.5.4. Frameworks Modelo I y Modelo II

DESARROLLO

Capítulo I. La Web2.0

1.1. Generalidades

El concepto de 'Web 2.0'[2] comenzó con una sesión de *'brainstorming'* realizada entre O'Reilly y MediaLive International. Dale Dougherty, pionero de la web y vicepresidente de O'Reilly, observaron que lejos de 'estrellarse', la web era más importante que nunca, con apasionantes nuevas aplicaciones y con sitios web apareciendo con sorprendente regularidad. Lo que es más, las compañías que habían sobrevivido al desastre parecían tener algunas cosas en común. ¿Podría ser que el derrumbamiento de las punto-com supusiera algún tipo de giro crucial para la web, de tal forma que una llamada a la acción tal como 'Web 2.0' pudiera tener sentido? Estuvimos de acuerdo en que lo hizo, y así nació la conferencia de la Web 2.0.

Tras año y medio, el término 'Web 2.0' ha arraigado claramente, con más de 9,5 millones de menciones en Google. Pero todavía existe un enorme de desacuerdo sobre qué significa Web 2.0, existiendo algunas críticas que afirman que se trata simplemente de una palabra de moda, fruto del marketing, y sin sentido, en tanto que otros la aceptan como un nuevo paradigma.

En una reunión inicial de *brainstorming*, formularon una interpretación de Web 2.0 según un ejemplo[2]:

Web 1.0		**Web 2.0**
Doble click	-->	Google AdSense
Ofoto	-->	Flickr
Akamai	-->	BitTorrent
mp3.com	-->	Napster
Britannica Online	-->	Wikipedia
personal websites	-->	blogging

evite	-->	upcoming.org and EVDB
domain name speculation	-->	search engine optimization
page views	-->	cost per click
screen scraping	-->	web services
publishing	-->	participation
content management systems	-->	wikis
directories (taxonomy)	-->	tagging ('folksonomy')
stickiness	-->	syndication

La lista crecía y crecía. ¿Pero qué era lo que nos permitía asociar una aplicación o enfoque a 'Web 1.0' y otro a 'Web 2.0'? (La pregunta es particularmente apremiante, porque el 'meme'de la Web 2.0 ha llegado a ser tan extenso, que las compañías están usando el término como una nueva palabra de moda fruto del marketing, sin comprender realmente lo que significa. Este asunto es particularmente difícil porque muchas de esas *startups* adictas a las palabras de moda no son en absoluto Web 2.0, mientras que algunas de las aplicaciones que asociamos a la Web 2.0, como Napster y BitTorrent, ¡no son, en sentido estricto, ni siquiera aplicaciones web!). Comenzamos por intentar extraer los principios que se deducen de una forma u otra de las historias de éxito de la Web 1.0 y por lo más interesante de las nuevas aplicaciones.

1.2. La Web como plataforma

Como muchos conceptos importantes, Web 2.0 no tiene una clara frontera, sino más bien, un núcleo gravitacional. Usted puede visualizar Web 2.0 como un sistema de principios y prácticas que conforman un verdadero sistema solar de sitios que muestran algunos o todos esos principios, a una distancia variable de ese núcleo.

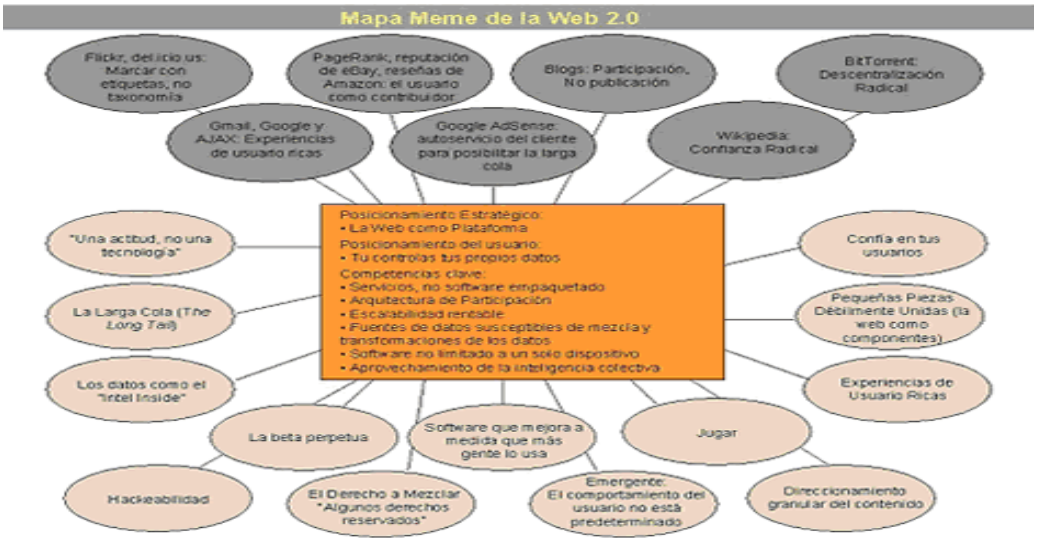


Figura: mapa de la Meme de la Web2.0[2]

La figura anterior muestra un 'mapa meme' de la Web 2.0 que fue desarrollado en una sesión de *brainstorming* durante el FOO Camp, una conferencia en O'Reilly Media. Es fundamentalmente trabajo en curso, pero manifiesta las muchas ideas que irradian desde el núcleo de la Web 2.0.

Por ejemplo, en la primera conferencia de la Web 2.0, en octubre de 2004, John Battelle enumeraron un conjunto preliminar de principios. El primero de dichos principios era 'la web como plataforma.'[2] Aunque esto era también un eslogan para lograr apoyos a favor del querido Netscape de la Web 1.0, que resultó derrotado tras una encendida batalla con Microsoft. Más aún, dos de nuestros ejemplos iniciales de Web 1.0, DoubleClick y Akamai, eran ambos pioneros en tratar a la web como una plataforma. La gente no piensa habitualmente en ello como 'web services', pero de hecho, la publicidad en sitios web fue el primer *'web service'* ampliamente desplegado, y el primer '*mashup*'(por utilizar otro término que ha ganado adeptos recientemente) que se desplegó extensamente. Cada *banner* actúa como elemento que facilita la cooperación transparente entre dos *websites,* proporcionando una página integrada a un lector en otro ordenador. Akamai también trata la red como plataforma, y en un nivel inferior de la pila, construyendo una red transparente de copia en caché y de distribución de contenidos que alivie la congestión del ancho de banda.

No obstante, estos pioneros proporcionaron contrastes útiles porque competidores posteriores han llevado su solución del mismo problema aún más lejos, entendiendo de forma más profunda la naturaleza de la nueva plataforma. DoubleClick y Akamai fueron pioneros de la Web 2.0, sin embargo podemos también ver cómo es posible materializar más posibilidades adoptando patrones de diseño adicionales de Web 2.0.

Vamos profundizar por un momento en cada uno de estos tres casos, desentrañando algunos de los elementos diferenciadores esenciales.

Netscape frente a Google

Si Netscape era el abanderado de la Web 1.0, Google es ciertamente el abanderado de la Web 2.0, aunque sólo sea porque sus respectivas salidas a bolsa fueron acontecimientos determinantes para cada era. Vamos comenzar con una comparación de estas dos compañías y de su posicionamiento.

Netscape ideó el concepto de 'la web como plataforma'[2] en términos del viejo paradigma del software: su buque insignia era el navegador web, una aplicación de escritorio, y su estrategia era utilizar su dominio en el mercado de los navegadores para crear un mercado de productos de servidor de gama alta. El control sobre los estándares para visualizar el contenido y las aplicaciones en el navegador, en teoría, dio a Netscape la clase de poder de mercado del que disfrutó Microsoft en el mercado de los PCs. Al igual que el 'carro sin caballos' posicionó al automóvil como extensión de lo conocido, Netscape promovió un '*webtop*' para sustituir al escritorio (el 'desktop'), y planeó poblar ese *webtop* con las actualizaciones de información y *applets* insertados en el *webtop* por los proveedores de información que comprarían los servidores de Netscape.

Google, por el contrario, comenzó su vida como una aplicación web nativa, nunca vendida o empaquetada, sino siempre entregada como un servicio, con clientes pagando, directamente o indirectamente, por el uso de ese servicio. Ninguna de las rémoras de la vieja industria del software está presente. No hay programación de las actualizaciones de las versiones del software, sencillamente mejora continua. Ninguna licencia o venta, sencillamente uso. Ningún tipo de portabilidad a diferentes plataformas de forma que los clientes puedan ejecutar el software en su propio equipo, sencillamente, una colección masiva de PCs escalables en los que corren sistemas

operativos de software abierto junto con aplicaciones y utilidades de su propia cosecha que nunca nadie de fuera de la compañía consigue ver.

En el fondo, Google requiere una capacidad que Netscape nunca necesitó: gestión de la base de datos. Google no es sencillamente una colección de herramientas software, es una base de datos especializada. Sin los datos, las herramientas son inútiles; sin el software, los datos son inmanejables. El licenciamiento del software y el control sobre las APIs (la palanca de poder en la era anterior) es irrelevante porque el software no necesita ser distribuido sino ejecutado, y también porque sin la capacidad de recoger y de gestionar los datos, el software es de poca utilidad. De hecho, el valor del software es proporcional a la escala y al dinamismo de los datos que ayuda a gestionar.

El servicio de Google no es un servidor (aunque es ofrecido por una colección masiva de servidores de Internet) ni un navegador (aunque es experimentado por el usuario a través del navegador)[2]. Ni siquiera su servicio insignia, el de búsqueda, almacena el contenido que permite encontrar a los usuarios. Como una llamada telefónica, que no tiene lugar en los teléfonos de los extremos de la llamada sino en la red que hay entre medias, Google tiene lugar en el espacio que se encuentra entre el navegador y el motor de búsqueda y el servidor de contenido destino, como un habilitador o intermediario entre el usuario y su experiencia *online*.

Aunque Netscape y Google se podrían describir como compañías de software, está claro que Netscape perteneció al mismo mundo del software que Lotus, Microsoft, Oracle, SAP, y otras compañías que surgieron durante la revolución del software de los años 80, mientras que los amigos de Google son aplicaciones de Internet como eBay, Amazon, Napster, y sí, DoubleClick y Akamai.

DoubleClick frente a Overture y AdSense

Como Google, DoubleClick es un verdadero hijo de la era del Internet[2]. Ofrece software como un servicio, tiene una competencia básica de gestión de datos, y, según lo mencionado anteriormente, era un pionero en web services mucho antes de que los web services tuvieran un nombre. Sin embargo, finalmente DoubleClick se vio limitado por su modelo de negocio. Apoyó en los años 90 el concepto de que la web trataba de publicación, no participación; que los publicistas, no los consumidores, deben ser los que deciden; que el tamaño importaba, y que Internet cada vez estaba

más dominada por los sitios web situados en la cima según las estadísticas de MediaMetrix y otras compañías que valoraban los anuncios de la web.

Como consecuencia, DoubleClick cita orgulloso en su web 'más de 2000 implementaciones exitosas' de su software. ¡Yahoo! Search Marketing (antes Overture) y Google AdSense, por el contrario, ya dan cada uno servicio a centenares de millares de publicistas.

El éxito de Overture y de Google fue fruto de la comprensión de lo que Chris Anderson cita como ' *the long tail* ' (literalmente 'la larga cola'), el poder colectivo de los sitios web pequeños que conforman la gran mayoría del contenido de la web. Las ofertas de DoubleClick requieren un contrato formal de venta, limitando su mercado a unos pocos miles de sitios web grandes. Overture y Google se las ingeniaron para permitir la colocación del anuncio prácticamente en cualquier página web. Lo que es más, evitaron los formatos de publicidad preferidos por los publicistas y las agencias de publicidad como *banners* y *popups* (ventanas emergentes), en favor de los anuncios de texto, mínimamente intrusivos, sensibles al contexto y amigables para el consumidor.

La lección de la Web 2.0: [2] **hacer uso del autoservicio del cliente y de la gestión de datos algorítmica para llegar a toda la web, a los extremos y no sólo al centro, a 'la larga cola' ('*the long tail* ') y no sólo a la cabeza.**

Como es de esperar, otras historias de éxito de la Web 2.0 demuestran este mismo comportamiento. eBay permite las transacciones ocasionales de tan solo algunos dólares entre simples individuos, actuando como un intermediario automatizado. Napster (aunque cerrado por razones legales) construyó su red no mediante la construcción de una base de datos centralizada de canciones, sino arquitecturando un sistema en el que cada individuo que descargaba algo también se convertía en un servidor (esto es, alguien del que otros se descargaban algo), y así creció la red.

Akamai frente a BitTorrent

Como DoubleClick, Akamai está optimizado para hacer negocios con la cabeza, no con la cola, con el centro, no con los extremos. Mientras que sirve a las necesidades

de los individuos en el extremo de la web facilitando su acceso a los sitios web de mucha demanda en el centro, obtiene sus ganancias de esos sitios centrales.

BitTorrent, como otros pioneros en el movimiento del P2P, adopta el enfoque radical de la descentralización del Internet. Cada cliente es también un servidor; los archivos están subdivididos en fragmentos que se pueden servir desde múltiples localizaciones, aprovechando de forma transparente la red de los individuos que se están descargando archivos para proporcionar tanto ancho de banda como datos a otros usuarios. De hecho, cuanto más popular es el archivo, más rápidamente se descarga, puesto que hay más usuarios que proporcionan ancho de banda y fragmentos del archivo completo.

BitTorrent demuestra así un principio dominante de la Web 2.0: **el servicio mejora automáticamente cuanta más gente lo use**. [2] Mientras que Akamai debe agregar servidores para mejorar el servicio, cada consumidor de BitTorrent aporta sus propios recursos al grupo. Hay una 'arquitectura implícita de participación', una ética de cooperación inherente, en la que el servicio actúa sobre todo como intermediario inteligente, conectando los extremos entre sí y aprovechando las posibilidades que ofrecen los propios usuarios.

La plataforma supera a la aplicación en todo momento

En cada una de sus últimas confrontaciones con los rivales, Microsoft ha jugado con éxito la carta de la plataforma, triunfando incluso sobre las aplicaciones más dominantes. Windows permitió que Microsoft desplazara el Lotus 1-2-3 con Excel, WordPerfect con Word, y Netscape Navigator con Internet Explorer.

Esta vez, sin embargo, el choque no es entre una plataforma y una aplicación, sino entre dos plataformas, cada una con un modelo de negocio radicalmente distinto: en un lado, un solo suministrador de software, cuya base masivamente instalada y sus firmemente integrados sistema operativo y APIs le proporcionan el control sobre el paradigma de programación; en el otro, un sistema sin un dueño, agrupado mediante una serie de protocolos, estándares abiertos y acuerdos de cooperación.

Windows representa la cumbre del control propietario mediante el software basado en APIs. Netscape intentó arrebatar el control a Microsoft usando las mismas técnicas

que el propio Microsoft había utilizado contra otros rivales, y falló. Pero Apache, que se aferró a los estándares abiertos de la web, ha prosperado. La batalla ya no es desigual, una plataforma contra una sola aplicación, sino plataforma frente a plataforma, siendo más bien la pregunta qué plataforma, y más profundamente, qué arquitectura, y qué modelo de negocio, se ajustan más a la oportunidad que se presenta por delante.

Windows era una solución brillante a los problemas de la era inicial del PC. Igualó las condiciones para todos los desarrolladores de aplicaciones, solucionando una multitud de problemas que previamente habían asediado a la industria. Pero una sola aproximación monolítica, controlada por un solo proveedor, ya no es una solución, es un problema. Los sistemas orientados hacia las comunicaciones, algo que ciertamente es Internet como plataforma, requieren interoperabilidad. A menos que un proveedor pueda controlar ambos extremos de cada interacción, las posibilidades de conseguir usuarios cautivos mediante el software basado en APIs son limitadas.

Cualquier proveedor de la Web 2.0 que intente asegurar los beneficios de su aplicación mediante el control de la plataforma, por definición, no estará contribuyendo al fortalecimiento de la plataforma.

Esto no quiere decir que no haya oportunidades para asegurar beneficios y conseguir una ventaja competitiva, pero creemos que no se deben obtener mediante el control sobre el software basado en APIs y los protocolos. Hay un nuevo juego en marcha. Las compañías que tendrán éxito en la era de la Web 2.0 serán las que entiendan las reglas de ese juego, en vez de intentar volver a las reglas de la era del software de PC.

1.3. Tecnología que conforma la Web2.0

1.3.1. Ajax

Generalidades.

Ajax está basado es una nueva tecnología web que cubre los beneficios de los abordajes basados en pantallas y los abordajes basados en páginas. Permitiéndole más funcionalidad sofisticada utilizando estándares web fáciles-de-implementar, Ajax provee una alternativa real para crear aplicaciones web poderosas.

Jesse James Garrett de Adaptive Path acuñó el término "Ajax" en febrero de 2005, pero la tecnología que está detrás no es nueva.

Los desarrolladores que construyen interfaces Ajax aprovechan las mismas herramientas que los abordajes basados en páginas: XHTML, CSS, y JavaScript. ¿Entonces por qué Ajax repentinamente es un tópico popular?

Una razón es que varias compañías grandes incluyendo Google han creado asombrosas aplicaciones utilizando la tecnología: Google Maps, Google Gmail y Google Suggest están todos construidos utilizando Ajax.

Otra razón es la continua adopción de navegadores que cumplen con los estándares que soportan la tecnología Ajax, más notablemente Firefox, Safari, Opera e Internet Explorer 6.

JavaScript Asíncrono y XML (AJAX) no es una tecnología por sí misma, es un término que describe un nuevo modo de utilizar conjuntamente varias tecnologías existentes. Esto incluye: HTML o XHTML, CSS, JavaScript, DOM, XML, XSLT, y el objeto XMLHttpRequest. Cuando estas tecnologías se combinan en un modelo AJAX, es posible lograr aplicaciones web capaces de actualizarse continuamente sin tener que volver a cargar la página completa. Esto crea aplicaciones más rápidas y con mejor respuesta a las acciones del usuario

1.3.1.1. JavaScript

JavaScript[3], es un lenguaje de programación de páginas web de lado del cliente, esto significa, que cuando estamos viendo una página que utiliza JavaScript, hemos descargado el código JavaScript a nuestro navegador y nuestro navegador lo está ejecutando de acuerdo con las acciones realizadas en la página.

Gracias a que se ejecuta en el navegador, JavaScript, nos permite responder de manera rápida y eficaz a las acciones del usuario, creando de esta manera aplicaciones interactivas.

Pero el hecho de ejecutarse en el navegador, da lugar a otros problemas, por ejemplo, el código ejecutado es enviado al navegador, por lo que puede ser obtenido por nuestros usuarios, de manera que no podremos utilizar JavaScript para proteger secciones con contraseña.

Además, JavaScript, es ejecutado por el navegador y cada fabricante de este tipo de software interpreta el mismo código JavaScript a su manera, por lo que tendremos que tener mucho cuidado si queremos que todo el mundo vea correctamente nuestra página.

Un ejemplo

Un clásico ejemplo de JavaScript es el típico popup o las alertas, en este caso vamos a hacer una alerta que dará el clasico 'hola mundo':

```
000 <script>
001 alert('Hola mundo');
002 </script>
```

Este código producirá un resultado cómo este.[3]

Cómo usarlo

Por norma general, podremos utilizar JavaScript en nuestras páginas HTML, incluyendo el código entre las etiquetas <script> y </script> cómo hemos visto en el ejemplo anterior, pero existen otras formas de usarlo cómo los ficheros js o en los eventos.

1.3.1.2. Xml

XML[4] significa *extensible markup language*, o lenguaje de anotación extensible.[11] Ya conocemos el lenguaje HTML (*hypertext markup language*), lenguaje de anotación para página webs que permite navegación tipo hipertexto; sin embargo, XML no es sólo un lenguaje, es una forma de especificar lenguajes, de ahí lo de extensible. Todo lenguaje que se exprese de una forma determinada puede ser XML. Por lo tanto, XML *no es un lenguaje para hacer mejores páginas web*, sino un lenguaje para información auto-descrita, o al menos, auto-descrita si las etiquetas están bien puestas.

XML se inició como un subconjunto de SGML (*structured generalized markup language*), un standard ISO para documentos estructurados que es sumamente complejo para poder servir documentos en la web. XML es algo así como SGML

simplificado, de forma que una aplicación no necesita comprender SGML completo para interpretar un documento, sino sólo el subconjunto que se defina. Los editores SGML, sin embargo, pueden comprender XML.

Por tanto, no debe uno pensarse que XML es para crear páginas web, o algo parecido a las páginas web. XML es un lenguaje que cambia el paradigma de programación: de basada en funciones u objetos a la *programación basada en el documento*. XML se puede usar para cambiar totalmente el paradigma de publicación; de un programa que recibe unas entradas y produce unas salidas, se pasa a un documento que genera otro documento, o bien programas que toman documentos y producen otros documentos. Por eso, también, y, en general, salvo en entornos de servicios web, lo normal es que el XML se use en el servidor, y se sirva otro tipo de documentos, HTML, por ejemplo, que se obtienen a base de una serie de transformaciones. Precisamente, esto hace que los documentos XML se usen dentro de entornos de aplicaciones. Este entorno de aplicaciones permite publicar documentos XML, que, antes de ser enviados al cliente, sufrirán una serie de transformaciones para adaptarlo a los requisitos del mismo. Algunos ejemplos de entorno de aplicaciones son el Cocoon, un entorno basado en Java, libre, que permite no sólo publicar páginas XML, sino también incluir programas dentro de las páginas (XSP). No se caracteriza por su velocidad ni amigabilidad, pero es excelente como entorno de desarrollo (y el precio es inmejorable). Otra alternativa gratuita es el AxKit, escrito en Perl. Como alternativas de pago (y bien pagadas) están el Bea Weblogic (del que puedes leer una introducción en programacion.com, y el IBM WebSphere Transcoding Publisher.

¿Cómo se usa XML?

Para editar documentos XML, al igual que para hacerlo con HTML, se puede hace de dos formas: editándolos como cualquier otro fichero ASCII, usando, si acaso, un editor estructurado como el XEmacs, o bien usar un editor específico para XML, que entiende las particularidades del lenguaje, lo indenta como está mandado, y te cierra solito las etiquetas.

Para hacer esto hay muchas opciones, tanto en Windows como en Linux, aunque la mayoría son de pago. Por ejemplo, XMLSpy tiene un buen entorno, funciona solo para Windows, paro es relativamente inestable (al menos las versiones probadas). eXcelon Stylus permite además aplicar transformaciones, en un entorno de tres paneles

bastante fijo. También es relativamente caro. <oXygen/> es bastante económico para uso personal o académico, y tiene una versión de prueba de treinta días. Está basado en Java, y funciona tanto en Windows como en Linux. Te completa las etiquetas, y es aceptablemente rápido. Se basa también en bastantes herramientas libres, tales como Batik y FOP de Apache. Otra opción, bastante simple, es XMLShell, que permite también hacer transformaciones XSLT simples.

Los mismos entornos incluyen facilidades para validar el código XML resultante, pero esto se puede hacer también usando *analizadores XML*, de los cuales hay muchos, de bastante buena calidad, y la mayor parte de ellos gratuitos. Uno de los más conocidos y usados es el Xerces, del cual hay versiones en Java, en Perl y en C++. Es adecuadamente rápido, y además incorpora todos los últimos estándares del W3. Otra opción, que además se puede usar desde Internet, es el XParse de Jeremie, que te analiza directamente el documento y te lo presenta en forma de árbol.

La mayor parte de los validadores pueden trabajar de dos formas: de forma independiente, y usándolos como librerías desde el lenguaje de programación de la elección de uno; por ejemplo, Xerces se puede usar *stand-alone*, o bien como una librería xerces.jar, cuyos objetos se pueden instanciar o usar desde el programa de uno.

1.3.1.3. Html

El HTML[4], Hyper Text Markup Language (Lenguaje de marcación de Hipertexto) es el lenguaje de marcas de texto utilizado normalmente en la www (World Wide Web).[12] Fue creado en 1986 por el físico nuclear Tim Berners-Lee; el cual tomo dos herramientas preexistentes: El concepto de Hipertexto (Conocido también como link o ancla) el cual permite conectar dos elementos entre si y el SGML (Lenguaje Estándar de Marcación General) el cual sirve para colocar etiquetas o marcas en un texto que indique como debe verse. HTML no es propiamente un lenguaje de programación como C++, Visual Basic, etc., sino un sistema de etiquetas. HTML no presenta ningún compilador, por lo tanto algún error de sintaxis que se presente éste no lo detectará y se visualizara en la forma como éste lo entienda.

El entorno para trabajar HTML es simplemente un procesador de texto, como el que ofrecen los sistemas operativos Windows (Bloc de notas), UNIX (el editor vi o ed) o el

que ofrece MS Office (Word). El conjunto de etiquetas que se creen, se deben guardar con la extensión .htm o .html.

Estos documentos pueden ser mostrados por los visores o "browsers" de páginas Web en Internet, como Netscape Navigator, Mosaic, Opera y Microsoft Internet Explorer.

También existe el HTML Dinámico (DHTML), que es una mejora de Microsoft de la versión 4.0 de HTML que le permite crear efectos especiales como, por ejemplo, texto que vuela desde la página palabra por palabra o efectos de transición al estilo de anuncio publicitario giratorio entre página y página.

Creación de páginas web con lenguaje HTML

Para crear una página web se pueden utilizar varios programas especializados en esto, como por ejemplo, el Microsoft Front Page o el Macromedia Dreamweaver 3. Otra forma de diseñar un archivo .html, es copiar todo en el Bloc de Notas del Windows, ya que este sencillo programa cumple con un requisito mínimo que es la posibilidad de trabajar con las etiquetas con las que trabaja este lenguaje.
A continuación les mostraremos las etiquetas más comunes que deben aprenderse para hacer una página Web.

Estructura de los documentos de HTML

Si se tiene en cuenta el contenido del documento, todos los documentos de HTML bien escritos comparten una estructura en común. Un documento de HTML empieza con la etiqueta <HTML>, que es la que encerrará el documento actual. Contiene dos secciones primordiales: la cabecera y el cuerpo encerrados respectivamente por los elementos <HEAD> cabeza y <BODY> cuerpo.
La cabecera puede contener información y siempre contiene el titulo del documento encerrado por el elemento <TITLE>.
En el cuerpo se encuentra todo el contenido del documento, ya sea, texto, imágenes, sonidos, hipervínculos, etc. Un documento escrito en HTML contiene las siguientes etiquetas en el siguiente orden:

Ejemplo:

<HTML>

<HEAD>

<TITLE> Título de mi página de Internet </TITLE>

</HEAD>

<BODY>

<H1> <CENTER> Primera página </CENTER> </H1>

<HR>

Esta es mi primera página, [4] aunque todavía es muy sencilla. Como el lenguaje HTML no es difícil, pronto estaremos en condiciones de hacer cosas más interesantes.

<P> Aquí va un segundo párrafo.

</BODY>

</HTML>

1.3.1.4. CSS

CSS[5] es un lenguaje de hojas de estilos creado para controlar la presentación de los documentos electrónicos definidos con HTML y XHTML. CSS es la mejor forma de separar los contenidos y su presentación y es imprescindible para la creación de páginas web complejas.

La separación de los contenidos y su presentación presenta numerosas ventajas, ya que obliga a crear documentos HTML/XHTML bien definidos y con significado completo (también llamados *"documentos semánticos"*). Además, mejora la accesibilidad del documento, reduce la complejidad de su mantenimiento y permite visualizar el mismo documento en infinidad de dispositivos diferentes.

Si el lenguaje HTML/XHTML se utiliza para *marcar* los contenidos, es decir, para designar lo que es un párrafo, lo que es un titular o lo que es una lista de elementos, el lenguaje CSS se utiliza para definir el aspecto de todos los contenidos, es decir, el color, tamaño y tipo de letra de los párrafos de texto, la separación entre titulares y párrafos, la tabulación con la que se muestran los elementos de una lista, etc.

1.4. Blogs

Qué es un Blog?

Un **Blog**[6] es una publicación en la Web compuesta de varios artículos, normalmente breves y a veces de carácter personal. Se comenzaron a masificar aproximadamente en el 2001, pero ya existían desde el 1999 y antes, aunque no con este nombre. Originalmente eran simplemente una especie de "diario de vida público" pero ahora toman distinto carácter, y si bien la mayoría mantiene una línea bastante personal, también los hay de servicio público, tipo revista, etc.

La diferencia más importante entre un blog y un sitio Web es que el blog está pensado para varios "***postings***" o artículos breves que se presentan en orden cronológico inverso (del más nuevo al más antiguo), y que se van agregando sin una agenda predeterminada. Además los blogs son un espacio predominantemente de aficionados, de forma que los blogs permiten a sus autores o "***bloggers***" pasar de ser consumidores y observadores pasivos a ser entidades activas en la creación de contenidos y culturas.

Tipos de Blog

Existen muchos tipos de blogs; una posible clasificación es la siguiente:

- **Blog Temático**, de política, noticias, opinión, etc.: comentarios sobre la actualidad noticiosa. Lo interesante en este caso es que hay una relación directa entre quien escribe y su audiencia, que no es mediada por un editor como en los medios tradicionales. Esto es bueno y malo, en el sentido de que no hay un filtro ni control de calidad, pero al mismo tiempo hay una libertad única. Se ha dicho también que los bloggers "*jugamos a ser periodistas*".
- **Blog Personal**: diario de vida o diario de viaje, con fotos o historias.
- **Blog Colaborativo**: un blog que es escrito por varias personas, como Sushi Knights. Lo compartido es el registro de usuarios, los tags o clasificaciones del contenido y la apariencia del sitio.
- **Fotolog**: un fotolog es una especie de blog en que no se escribe mucho, sino a lo más un párrafo y lo más importante del contenido es una foto que ilustra un momento del tiempo o un acontecimiento.

- **AudioBlog o Podcast**: un blog en el que se publica un programa de audio (similar a un programa de radio), normalmente de 20-60 minutos de duración y que permite a los usuarios descargarlo a un dispositivo como un reproductor portátil de MP3s.

Tecnologías

En el aspecto técnico, la mayoría de los blogs incluye alguna forma de participación de los lectores a través de un foro de comentarios que normalmente está asociado a cada artículo. En los blogs más pequeños la falta de comentarios es un indicio de que no hay muchos visitantes, mientras que en los blogs más grandes es necesario usar algún sistema de moderación editorial o con puntajes que permita filtrar los comentarios.

Además de la herramienta base que se usa para hacer el blog, cinco tecnologías son clave en la forma en que los blogs se comunican y expanden: la posibilidad de clasificar contenido, la posibilidad de suscribirse (feeds), la posibilidad de notificar a otro blog que estoy escribiendo sobre su artículo, la posibilidad de notificar a un servidor central que mi blog tiene contenido nuevo, y la posibilidad de tener una URL estable para cada artículo.

Tags (etiquetas): son la forma de clasificar la información, normalmente son 1 o 2 palabras que se agregan a cada artículo y que permiten agruparlos después por temas como "arte", "cultura", "tecnología", etc.

Sindicación (feeds): lo más usual en este caso es que en tu blog dejas un archivo en formato XML que contiene las últimas cosas que has publicado en un formato que es legible por un ordenador, y que incluye el título del artículo, la fecha y un breve resumen. Esta información puede ser usada por los visitantes de tu sitio en usando un agregador de noticias. Un agregador de noticias es un programa para poder ver en un solo lugar todos lo nuevo que ha aparecido en los blogs a los que tú estás subscrito. Vía Web, algunos agregadores de noticias usados son: My Yahoo, Bloglines y Newsgator. Si quieres un programa para usar en tu computador, mira la comparación de lectores de noticias RSS.

Trackbacks: es normal en los blogs ir propagando la información, y por lo tanto si tu blog se hace medianamente famoso será común que otros blogs hablen sobre artículos en tu blog. Y para tí será interesante poder saber quiénes están hablando de tu blog en otros blogs. Para eso existen los trackbacks, que son un servicio que proveen los programas para crear blogs. Cuando escribes un artículo, normalmente aparece una dirección "Trackback URL" que las personas que escriban sobre tu artículo en otros blogs deben ingresar en sus sistemas para poder enviarte estas notificaciones.

Un análisis científico de la propagación de las noticias en los blogs es el artículo del 2005 de Adar y Adamic: "*Tracking information epidemics in the Blogspace*".

Pings: los programas para hacer blogs normalmente incluyen una forma de notificar a otros sitios como Technorati que has publicado algo nuevo, esto normalmente viene configurado "de fábrica" para apuntar a varios sitios que indexan blogs.

Permalinks: este es el nombre que recibe la dirección con que queda tu artículo cuando está publicado. Normalmente termina en .html y es una dirección permanente, invariante y segura en el sentido de que otras personas pueden linkear a tu artículo y no importa si lo editas o lo cambias la dirección seguirá siendo la misma.

Malas prácticas

Lo peor

- **Hablar solamente de otros Blogs**: como los programas de televisión que hablan 90% sobre la televisión, la idea es aportar algo nuevo al Web.
- **No poner resumen en el RSS**. esto restringe lo que se puede ver offline, y además no ayuda a que la gente defina qué quiere leer y qué no.
- **Poner links a través de una redirección** del sitio, sobre todo si el link funciona mal.
- **Publicar periódicamente sólo por cumplir con publicar algo**. A mí me parecen mucho más valiosos los que publican cuando tienen algo que decir.[6]

Lo malo, pero no tan malo

- **Restringir el contenido del blog a usos no-comerciales** (cláusula nc de Creative Commons). Si alguien usa algo que yo escribí en una obra comercial, tarde o temprano eso me traerá algún tipo de beneficio, y no pienso caer en el vicio de las editoriales, distribuidoras de música y películas, que quieren cobrar por todo lo que sea técnicamente posible cobrar. Como decía un profesor a quien respeto mucho: muchas veces es mejor entregar algo gratis que cobrar barato.
- **Mezclar temas personales y profesionales en el mismo blog**. Es mejor tener dos blogs.
- **No indicar claramente El autor (o La autora)**. Es lo más importante, porque la gracia de un blog es que hay una "mano" detrás, un criterio humano. Es esa persona la que le da a un blog su carácter.ç

Herramientas para hacer un blog

Para hacer un Blog, si no quieres instalar ningún programa puedes usar un servicio de creación de Blogs en línea, los más comunes son Blogger y LiveJournal, también hay proveedores especializados en Blogs en castellano como Vivito.

Si puedes instalar programas en un servidor, puedes usar otros software para crear blogs como Drupal o Movable Type.

1.5. Wikis

Qué es un Wiki?

Un Wiki[7] (del hawaiano wiki wiki, «rápido») **es un sitio web colaborativo que puede ser editado por varios usuarios**. Los usuarios de una wiki pueden así crear, editar, borrar o modificar el contenido de una página web, de una forma interactiva, fácil y rápida; dichas facilidades hacen de una wiki una herramienta efectiva para la escritura colaborativa.

Principales características de los Wikis. En general permiten:

- La publicación de forma immediata usando sólo el navegador web (ej. Explorer, Firefox, Mozilla, etc.)
- El control del acceso y de permisos de edición
- Que se registre quién y cuándo se ha hecho la modificación

- El acceso a versiones previas a la última modificación así como su restauración

Todo esto los dota de un gran potencial para el trabajo colaborativo en el aula.

Figura. Wikipedia[8]

Diferencias entre los Blogs y los Wikis:

Blogs	**Wikis**
- Normalmente un sólo autor/editor + comentarios - Estructura cronológica empezando por la última "entrada" - Links externos	- Muchos autores al mismo nivel - La estructura puede ser variada, sustituyéndose las versiones a medida que se modifica. - Links externos e internos

Cuadro de diferencias entre blogs y wikis[8]

Capítulo II: La Tecnología AJAX

2.1. Definición

AJAX,[9] acrónimo de ***A**synchronous **J**avaScript **A**nd **X**ML* (JavaScript y XML asíncronos), es una técnica de desarrollo web para crear aplicaciones interactivas. Éstas se ejecutan en el cliente, es decir, en el navegador de los usuarios, y mantiene comunicación asíncrona con el servidor en segundo plano. De esta forma es posible

realizar cambios sobre la misma página sin necesidad de recargarla. Esto significa aumentar la interactividad, velocidad y usabilidad en la misma.

AJAX creado por Adaptive Path, es una técnica de desarrollo que nace en el año 2005 y que promete igualar en cuanto a rendimiento y percepción del usuario las aplicaciones desktop y las basadas en web.

La Tecnología AJAX trata de eliminar los siguientes problemas clásicos en las aplicaciones web:

- ✓ *Interactividad pobre.* El usuario generalmente espera con una página en blanco hasta que se cargue toda la página.
- ✓ *Tardanza en la respuesta y alto consumo de ancho de banda*. Las aplicaciones clásicas web transfieren al servidor y este devuelve código HTML al navegador. Esto hace que se consuman en muchas ocasiones grandes anchos de banda y el performance de la aplicación sea bajo, cargue toda la página.
- ✓ Interfases demasiado simples. Para conseguir interfases más sofisticadas se ha utilizado generalmente Flash que implica desarrolladores muy especializados e implica la descarga en el cliente de un applet, normalmente pesados y en ocasiones no permitidos, cargue toda la página.
- ✓ Grado de usabilidad bajo.

La Tecnología AJAX trata de igualar la experiencia web y destop. Un ejemplo muy significativo es toda la tecnología AJAX desplegada en Google Maps**.**

AJAX es una combinación de tres tecnologías ya existentes:[9]

- ❖ **XHTML** (o **HTML**) y hojas de estilos en cascada (CSS) para el diseño que acompaña a la información.
- ❖ Document Object Model (DOM) accedido con un lenguaje de scripting por parte del usuario, especialmente implementaciones ECMAScript como JavaScript y JScript, para mostrar e interactuar dinámicamente con la información presentada.

- ❖ XML es el formato usado comúnmente para la transferencia de vuelta al servidor, aunque cualquier formato puede funcionar, incluyendo HTML preformateado, texto plano, JSON y hasta EBML.

Adicionalmente existe el objeto **XMLHttpRequest** para intercambiar datos asincrónicamente con el servidor web. En algunos frameworks y en algunas situaciones concretas, se usa un objeto iframe en lugar del XMLHttpRequest para realizar dichos intercambios.

Como el DHTML, LAMP o SPA, AJAX no constituye una tecnología en sí, sino que es un término que engloba a un grupo de éstas que trabajan conjuntamente.

Ajax[10] no es una tecnología. Es realmente muchas tecnologías, cada una floreciendo por su propio mérito, uniéndose en poderosas nuevas formas. AJAX incorpora:

- presentación basada en estándares usando XHTML y CSS;
- exhibición e interacción dinámicas usando el Document Object Model;
- Intercambio y manipulación de datos usando XML and XSLT;
- Recuperación de datos asincrónica usando XMLHttpRequest;
- y JavaScript poniendo todo junto.

2.2. El funcionamiento de AJAX.

El modelo clásico de aplicaciones Web funciona de esta forma: La mayoría de las acciones del usuario en la interfaz disparan un requerimiento HTTP al servidor web. El servidor efectúa un proceso (recopila información, procesa números, hablando con varios sistemas propietarios), y le devuelve una pagina HTLM al cliente. Este es un modelo adaptado del uso original de la Web como un medio hipertextual, pero como fans de The Elements of User Experience sabemos, lo que hace a la Web buena para el hipertexto, no la hace necesariamente buena para las aplicaciones de software.

Figura 1: El modelo tradicional para las aplicaciones Web (izq.) comparado con el modelo de AJAX *(der.).*[10]

Este acercamiento tiene mucho sentido a nivel técnico, pero no lo tiene para una gran experiencia de usuario. Mientras el servidor está haciendo lo suyo, que está haciendo el usuario? Exacto, esperando. Y, en cada paso de la tarea, el usuario espera por más.

Obviamente, si estuviéramos diseñando la Web desde cero para aplicaciones, no querríamos hacer esperar a los usuarios. Una vez que la interfaz está cargada, porque la interacción del usuario debería detenerse cada vez que la aplicación necesita algo del servidor? De hecho, porque debería el usuario ver la aplicación yendo al servidor?

Como es diferente AJAX

Una aplicación AJAX elimina la naturaleza arrancar-frenar- arrancar-frenar de la interacción en la Web introduciendo un intermediario -un motor AJAX- entre el usuario y el servidor. Parecería que sumar una capa a la aplicación la haría menos reactiva, pero la verdad es lo contrario.

En vez de cargar un página Web, al inicio de la sesión, el navegador carga al motor AJAX (escrito en JavaScript y usualmente procesado en un frame oculto). Este motor es el responsable por renderizar la interfaz que el usuario ve y por comunicarse con el servidor en nombre del usuario. El motor AJAX permite que la interacción del usuario con la aplicación suceda asincrónicamente (independientemente de la comunicación con el servidor). Así el usuario nunca estará mirando una ventana en blanco del navegador y un icono de reloj de arena esperando a que el servidor haga algo.

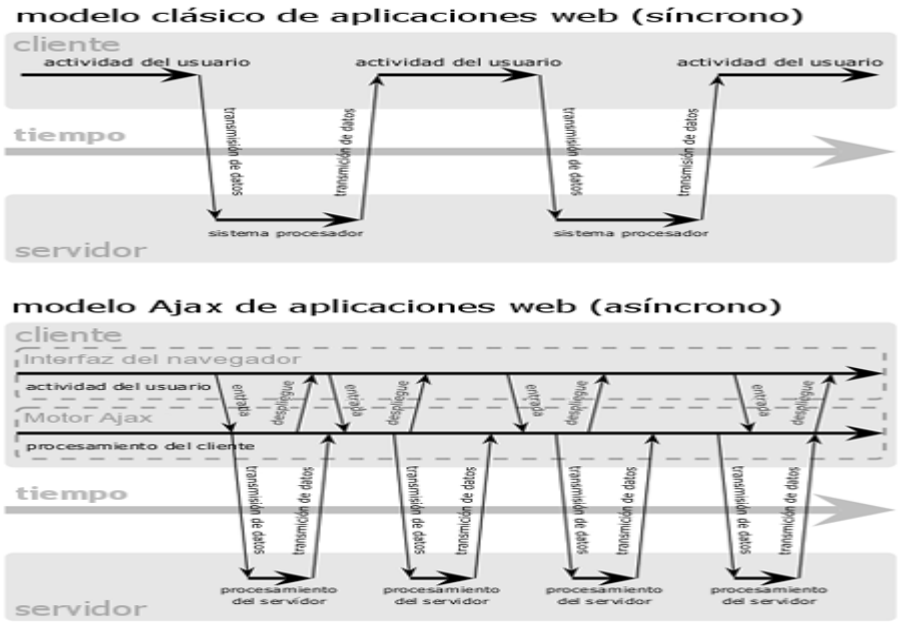


Figura 2: El patrón de interacción sincrónica de una aplicación Web tradicional (arriba) comparada con el patrón asincrónico de una aplicación AJAX *(abajo).*[10]

Cada acción de un usuario que normalmente generaría un requerimiento HTTP toma la forma de un llamado JavaScript al motor AJAX en vez de ese requerimiento. Cualquier respuesta a una acción del usuario que no requiera una viaje de vuelta al servidor (como una simple validación de datos, edición de datos en memoria, incluso algo de navegación) es manejado por su cuenta. Si el motor necesita algo del servidor para responder (sea enviando datos para procesar, cargar código adicional, o recuperando nuevos datos) hace esos pedidos asincrónicamente, usualmente usando XML, sin frenar la interacción del usuario con la aplicación.

Sin Ajax

Se crearía una página con un formulario, cuando el usuario envía los datos del formulario se produce una conexión a la base de datos y se muestra por pantalla la página que el servidor devuelve, todo esto hace que se recargue la página ya sea saltando a una diferente o a ella misma, el usuario debe esperar una nueva carga de página después de cada envío.

Es lento porque debe descargar la información HTML por duplicado.

Con Ajax

Utilizaríamos un código JavaScript que que crearía el mencionado objeto XMLHttpRequest al enviar el formulario, esta llamada se produce de forma asíncrona lo que significa que se envían los datos y no se recarga la página, una vez el servidor responde una función JavaScript es la que valora la respuesta del servidor, si esta respuesta es la deseada imprimiremos el texto que indique al usuario que sus datos fueron enviados correctamente.

El navegador no recarga la página, la experiencia desde el punto de vista del usuario es muy satisfactoria puesto que se asemeja a la respuesta del típico software de escritorio, ya no te planteas enlazar páginas sino enviar y recibir datos en una misma página que mediante funciones evalúa las diferentes respuestas.

Es bastante más rápido puesto que no tiene que descargar de nuevo el código HTML de la página de confirmación del formulario.

2.3. Razones para utilizar AJAX[11]

1. Basado en los estándares abiertos
2. Usabilidad
3. Válido en cualquier plataforma y navegador
4. Beneficia las aplicaciones web
5. No es difícil su utilización
6. Compatible con Flash
7. Adoptado por los "gordos" de la tecnología web
8. Web 2.0
9. Es independiente del tipo de tecnología de servidor que se utilice
10. Mejora la estética de la web

1. Basado en los estándares abiertos

Ajax está formado por las tecnologías Javascript, html, xml, css, y XML HTTP Request Object, siendo este último el único que "no es" estándar pero es soportado por los navegadores más utilizados de internet como son los basados en mozilla, internet explorer, safari y opera.

2. Usabilidad

Permite a las páginas hacer una pequeña petición de datos al servidor y recibirla sin necesidad de cargarla página entera. El incremento de las actualizaciones "on the fly" elimina el tener que refrescar el navegador, algo bastante apreciado a la hora de operar en una aplicación web.

3. Válido en cualquier plataforma y navegador

Internet explorer, los basados en mozilla y firefox son los que se llevan la do de internet y además son los navegadores en los que es más fácil programar aplicaciones Web AJAX, pero ahora es posible construir aplicaciones web basadas en AJAX para que funcionen en los navegadores más modernos. Es una de las razonas más importantes por las que AJAX se ha vuelto tan popular. Aunque si bien muchos desarrolladores sabían que era posible usarse años atrás con Internet Explorer, no era viable realizarse. Ahora ya es posible su avance gracias a Mozilla y Firefox.

4. Beneficia las aplicaciones web

AJAX es la cara del presente en las aplicaciones web - las aplicaciones web conllevan ciertos beneficios sobre las aplicaciones sobre escritorio (aplicaciones que dependan de un sistema operativo, librerías, lo que entendemos por programas compilados). Esto incluyo un menor coste de creación, facilidad de soporte y mantenimiento, menores tiempos a la hora de desarrollarlas, y sin necesidad de instalaciones; éstas son algunas de los beneficios que han llevado a las empresas y usuarios el adoptar aplicaciones web desde mediados de los 90. AJAX solo ayudará a las aplicaciones web a mejorar y conseguir un mejor resultado de cara al usuario final.

5. No es difícil su utilización

Porque AJAX está basada en los estándares que han sido utilizados durante muchos años, muchos desarrolladores web han tenido que utilizar las tecnologías que las aplicaciones AJAX requieren. Esto significa que no es un gran esfuerzo el aprendizaje de los desarrolladores el pasar de un simple código HTML y aplicaciones web a una potente aplicación AJAX. También significa que los desarrolladores puedes actualizar poco a poco las interfaces de usuario hacia unas interfaces con AJAX; no necesita una re-escritura de la aplicación entera, se puede hacer incrementalmente.

6. Compatible con Flash

Muchos desarrolladores tienen serias dudas sobre usar Flash o AJAX. Definitivamente hay ventajas y desventajas en ambas tecnologías según la situación que se de pero también hay muchas posibilidades y muy buenas para que amabas funcionen en conjunto.

7. Adoptado por los "gordos" de la tecnología web

La difusión de AJAX en los líderes de la industria de internet prueba que el mercado acepta y valida el uso de esta tecnología. Todo el mundo esta migrando hacia AJAX incluyendo Google, Yahoo, Amazon, Microsoft (por nombrar unas pocos). Google Maps fue lo que captó la atención de los desarrolladores web. Cuando empezaron a investigar como google era capaz de llevar esa increíble herramienta dentro de un navegador sin necesidad de ningún tipo de plug-in, encontraron que AJAX estaba detrás del tema.

8. Web 2.0

El movimiento Web 2.0 está cada vez más en auge y dando quebraderos de cabeza de muchos programadores, usuarios, y vendedores. Esto esta ayudando la adopción de AJAX. Las interfaces de AJAX son un componente clave de muchas de las aplicaciones Web 2.0, como puede ser BackPack (un organizador de disco online en entorno Web) y Google Maps. Afortunadamente gracias bombo que se le está dando, acelerará la adopción de AJAX y la los beneficios de su uso lo mantendrá en escena. Una de las claves principales de Web 2.0 es el usar la red como plataforma para el desarrollo de aplicaciones, en vez de simples páginas web. Siendo importante la iteración de los usuarios con la aplicación en sí.

9. Es independiente del tipo de tecnología de servidor que se utilice

Así como AJAX funciona en cualquier navegador, es perfectamente compatible con cualquier tipo de servidor estándar y lenguaje de programación Web. PHP, ASP. ASP.Net, Perl, JSP, Cold Fusión. El ser completamente compatible el desarrollo en éstas tecnologías ha ayudado a AJAX a que vaya cada vez más en auge.

10. Mejora la estética de la web

Con AJAX se puede interactuar la imaginación del desarrollador con la usabilidad de una aplicación web de forma que se pueda realizar una aplicación que si no estuviera dentro de un navegador, podría pasar por una aplicación normal de escritorio.

Capítulo III: Frameworks

3.1. Introducción.

Los frameworks orientados al objeto (llámense simplemente frameworks) son la piedra angular de la moderna ingeniería del software. El desarrollo del framework está ganando rápidamente la aceptación debido a su capacidad para promover la reutilización del código del diseño y el código fuente (source code). **Los frameworks**[12] son los Generadores de Aplicación que se relacionan directamente con un dominio específico, es decir, con una familia de problemas relacionados.

Como ejemplo, considere la construcción de un kit de herramientas de interface gráfica del usuario (GUI Tool Kit). Puede ser que elijamos diseñar y poner un solo kit de herramientas en ejecución. Por otra parte, si diseñamos el kit de herramientas como framework, este puro diseño nos permitirá generar una colección de los kits de herramientas para una variedad de aplicaciones del tipo GUI (Graphic User Interface). Los frameworks deben generar las aplicaciones para un dominio entero. Por lo tanto, debe haber puntos de flexibilidad que se puedan modificar los requisitos particulares para ajustarse a la aplicación. Por ejemplo, un punto de extensión puede ser el algoritmo usado para trazar elementos gráficos.

La capacidad de reutilización del código y del diseño de frameworks orientados al objeto permite una productividad mayor y un tiempo de Mercado breve en el desarrollo de aplicaciones, en comparación con el desarrollo tradicional de los sistemas de software. La configuración flexible de frameworks, permite la reutilización del núcleo kernel. El desarrollo del framework ha sido exitoso en muchos dominios. Algunos ejemplos incluyen el Microsoft Foundation Classes (**MFC**) framework,el Object Management Group's (**OMG**) y el COM+ y DCOM de Microsoft.

3.2. ¿Qué es un Framework?[12]

Un framework representa una arquitectura de software que modela las relaciones generales de las entidades del dominio. Provee una estructura y una metodología de trabajo la cual extiende o utiliza las aplicaciones del dominio.

Los Frameworks son diseñados con el intento de facilitar el desarrollo de software, permitiendo a los diseñadores y programadores pasar más tiempo identificando requerimientos de software que tratando con los tediosos detalles de bajo nivel de proveer un sistema funcional. Por ejemplo, un equipo que usa Apache Struts para desarrollar un sitio web de un banco puede enfocarse en cómo los retiros de ahorros van a funcionar en lugar de preocuparse de cómo se controla la navegación entre las páginas en una forma libre de errores. Sin embargo, hay quejas comunes acerca de que el uso de frameworks añade código innecesario y que la preponderancia de frameworks competitivos y complementarios significa que el tiempo que se pasaba programando y diseñando ahora se gasta en aprender a usar frameworks.

Fuera de las aplicaciones en la informática, un framework puede ser considerado como el conjunto de procesos y tecnologías usados para resolver un problema complejo. Es el esqueleto sobre el cual varios objetos son integrados para una solución dada.

3.3. ¿Qué es un Framework Web?[13]

Un framework web es una estructura definida, reusable en el que sus componentes facilitan la creación de aplicaciones web. En cierto sentido podemos afirmar que nos proveen una capa de abstracción sobre la arquitectura original ocultándola o adaptándola para no tener que utilizar el protocolo http de manera nativa y así acelerar los tiempos de desarrollo y mantenimiento.

3.4. Arquitectura básica de la Web[13]

La Web o www está basada en el protocolo http. Este protocolo, que apareció a principios de los 90, tiene dos tipos de mensajes: request y response, que se suceden sincrónicamente, es decir, no hay response sin un request previo y no puede haber más de un request consecutivo sin response ni viceversa. Estos mensajes se envían una vez establecida la conexión entre el servidor y el cliente. Hasta la versión 1.0

inclusive, por cada conexión sólo se podía enviar un request y un response antes de que esta se cerrara. A partir de la versión 1.1 se permiten las conexiones persistentes y es posible enviar una secuencia de estos mensajes (intercalados) mientras que ninguna de las partes decida cerrar la conexión.

Para los desarrolladores, este protocolo tiene una característica bastante particular que incide directamente en la arquitectura del sistema a construir y es que es un protocolo stateless (sin estado), o lo que es lo mismo, el servidor no recuerda si antes de un determinado mensaje se enviaron otros desde el mismo cliente ni recuerda los datos que se enviaron.

Los mensajes están estructurados internamente para contener distinto tipo de información.

Un mensaje request posee:

- Dirección de destino
- Tipo de Mensaje (GET, POST, HEAD, etc)
- Encabezado de Mensaje
- Cuerpo de mensaje (Si existe)

El tipo de mensaje indica cómo debe responder el servidor. En los encabezados se indica ciertos valores que debe tener el cuenta el servidor al tratar la información como por ejemplo, tipo de contenido enviado, tipo de contenido de respuesta admitido por el cliente, etc,etc...En el cuerpo del mensaje puede ir cualquier tipo de contenido siempre que se lo especifique correctamente en el encabezado. Sin embargo, generalmente o no se envía nada o se envía información en formato urlencoded que consiste en una cadena atributo-valor codificada de una manera especial para que el servidor no confunda la información con los caracteres de control

Por otro lado, un mensaje response posee:

- Código de resultado (200, 404, 500, etc,etc)
- Encabezados de mensaje
- Cuerpo de mensaje (Si existe)

El código indica como resultó la operación solicitada. El primer dígito indica el tipo de resultado (1=informativo, 2=exitoso, 4=error del cliente, 5=error del servidor). Los encabezados indican por lo general el tipo de contenido y temas referidos por ejemplo a si el navegador debe guardar o no el contenido en cache). Por otro lado, el cuerpo del mensaje, si existe, contiene el resultado de la operación solicitada. Generalmente está en formato html, xml, o de imagen pero puede contener cualquier tipo de datos siempre que se lo especifique correctamente en los encabezados

3.5. Evolución de los frameworks Web.[13]

Internet fue evolucionando y con ella lo hicieron los frameworks Webs, a medida que los desarrollos abandonaban el modelo cliente servidor para pasarse a los clientes livianos se puso más énfasis en la construcción de marcos de desarrollo más potentes

3.5.1. La Web estática

Al principio, en los primeros años de los noventa, el protocolo http sólo se utilizaba para transferir texto almacenado en documentos. El servidor respondía a las peticiones (request) enviando el contenido del archivo solicitado sin mayor procesamiento. No era posible parametrizar nada ni enviar formularios ni nada parecido, sólo la dirección del documento que se deseaba leer en el navegador, nada se podía generar dinámicamente, todo era contenido estático.

3.5.2. La interfaz CGI

La interfaz CGI (Common Gateway Interface) apareció en 1993 y definía una interfaz a través de la cual los programas y el servidor web se podían comunicar entre sí. Esto permitió que cualquiera con un navegador web pudiera ejecutar programas en la computadora que hace de servidor. A través de la interfaz cgi, el servidor y los programas que se encuentran en un área protegida (generalmente el directorio cgi-bin) se comunican de la siguiente manera:

1. Cuando llega una petición http-request al servidor y este detecta que se la debe transferir al programa, lo instancia y le pasa, a través de la entrada estándar y las variables de entorno, toda la información que necesite (CGI define unas variables de entorno mínimas a pasar, la más conocida llamada QUERYSTRING).

2. El programa lee esas variables y la entrada estándar, ejecuta su lógica y escribe el resultado en la salida estándar.

3. El servidor toma la salida estándar del programa y la envía al navegador como mensaje http-response.

Con este agregado se empezaron a construir las llamadas Aplicaciones WEB. Estas estaban escritas generalmente en lenguajes de scripting como Perl o, más tarde, PHP, aunque también existían cgis compilados programados en lenguajes como c o c++. Sin embargo, trabajar con la interfaz sola hacía difícil la construcción de aplicaciones grandes. Entre las desventajas principales estaban:

- Problemas de escalabilidad: Se creaba una instancia del programa por cada petición
- Se debía chequear recurrentemente que la entrada tuviera el formato correcto
- Era difícil de separar el contenido estático del contenido generado por el programa

A medida que las aplicaciones webs se fueron volviendo cada vez más importantes, el modelo de de interfaz entre el servidor y los ejecutables empezó a ser insuficiente y se buscaron soluciones alternativas.

3.5.3. Los Servidores de Aplicaciones Web.

Con la aparición de java y sus servlets, SUN creó una abstracción de los mensajes que recibe el servidor en objetos (GenericServlet, HttpServlet) de los que se debía heredar para redefinir su comportamiento. Estos objetos residen en un contenedor llamado Servlet Container que es el encargado de gestionar su ciclo de vida y de transformar la entrada http en objetos que se pasaban como parámetro a estos servlets. Así se solucionaban varios de los problemas que tenía la tecnología CGI como la escalabilidad y el chequeo inicial de los datos pero seguía existiendo el problema de la mezcla de contenido estático (html) con el código programado, sobre todo porque el código estático, que era la mayor parte de lo que se enviaba se escribía línea por línea mediante funciones del lenguaje lo que lo hacía completamente ilegible e inmantenible. Por este motivo SUN creó la especificación JSP que permitían la

escritura de una página html con algunas etiquetas especiales en las que se podía impregnar código java. Esta página, la primera vez que se ejecutaba se compilaba y se transformaba en un servlet con la misma funcionalidad que los originales, pero este detalle de poder escribirla como una página Web común y corriente fue lo que logró que se desarrollaran aplicaciones mucho más rápido y más mantenibles.

3.5.4. Frameworks Modelo I y Modelo 2.

Los términos Modelo I y Modelo II surgen de borradores de las primeras especificaciones de Java Server Pages (JSP) en donde se describían dos patrones básicos para construir aplicaciones basadas en esa tecnología.

El modelo 1 carece del concepto de controlador central que sí tiene el modelo 2.

Una arquitectura de Modelo 1 consiste básicamente en que desde la página JSP accedida desde el navegador se tuviera acceso a las clases del modelo, se les aplicara la lógica del negocio y se seleccionara la siguiente vista a ser enviada al navegador. Cada página procesaba su propio input.

En una arquitectura Modelo 2 existe un servlet que actúa de controlador central que recibe todos los requests y a partir de la dirección de origen, de los datos de entrada, el estado actual de la aplicación selecciona la siguiente vista a mostrar. En este servlet se pueden concentrar todos los temas relativos al conjunto total de las llamadas como la seguridad y la auditoría (logging). Las aplicaciones son más extensibles porque las vistas no se relacionan con el modelo ni entre sí. Por otro lado, a la las clases que implementan la lógica de la aplicación les llega la información digerida para su tratamiento (por ejemplo, en struts los datos forman parte de un formulario). Este servlet central recibe el nombre de Front Controller.

3.6. Modelos de Frameworks.[13]

3.6.1. Struts

Struts es por el momento el más difundido de los frameworks opensource en el ámbito java. Está basado en el modelo 2 y consta, por lo tanto, de un servlet que actúa de controlador central que recibe todas las peticiones de los clientes. Las facilidades de desarrollo que ofrece son:

- Lógica de navegación entre páginas
- Binding entre java y el html
- Validación de entradas
- Internacionalización
- Independencia del motor de visualización
- Maquetación

3.6.2. Tapestry

Tapestry es otro framework open-source modelo 2 mantenido por la comunidad Apache y una de sus principales características es que está basado en un modelo de componentes. Esto provee una estructura consistente, permitiendo al framework asumir responsabilidades sobre conceptos como la construcción de URL, el despacho, el almacenamiento del estado en el cliente o en el servidor, la validación del usuario, la localización/internacionalización, el manejo de reportes, etc. Un componente es un objeto que tiene sus responsabilidades definidas por el diseño y la estructura del framework en el cual se encuentra. Es decir, sigue una serie de convenciones (nomenclatura, implementación de ciertas interfaces, etc) que le exige el framework. Tapestry, al igual que todos los frameworks de modelo 2, tiene un servlet que centraliza toda la lógica de comunicación.

Existen ciertos conceptos clave que se definen para entender el funcionamiento:

Página: las aplicaciones consisten de una colección de páginas identificadas unívocamente a través de su nombre. Cada página contiene un template y otros componentes. **Template:** Un template puede ser para una página o un componente. Contiene html plano con algunos tags marcados con un atributo especial para indicar que en ese lugar deben situarse los componentes. **Componente:** Un objeto reusable que puede ser usado como parte de una página. Los componentes generan html cuando se renderiza la página y pueden participar cuando se selecciona un enlace o se envía un formulario. También se pueden utilizar para crear nuevos componentes. **Parámetro:** Los componentes tienen parámetros que sirven para enlazar las propiedades de los componentes con las propiedades de la página. Los componentes generalmente leen los parámetros pero a veces pueden modificarlos haciendo que se actualicen las propiedades de la página que estaban relacionadas con ese parámetro.

Desarrollar con Tapestry permite:

- Transparencia en la construcción de las vistas
- Binding entre java y html
- Manejo de eventos
- Construcción de componentes
- Validación de entradas
- Internacionalización

3.6.3. Java Server Faces.

Lo primero que se puede decir sobre Java Server Faces es que no es una implementación sino una especificación (JSR 127) aprobada por el Java Community Process (JCP) para construir interfaces de usuario para las aplicaciones que corren en un servidor. Esto quiere decir que es un estándar y pueden existir varias implementaciones mientras que cumplan con lo que exija la especificación.

JSF es otro framework modelo 2 que posee un controlador central (FrontControler) que se encarga de manejar todas las peticiones del cliente y gestionar su ciclo de vida. Está basado en un modelo de componentes para la interfaz de usuario. Un componente JSF es un elemento reusable y configurable que se puede utilizar en la interfaz de usuario. Los componentes se pueden anidar. Por ejemplo, una página contiene un componente mapa que a su vez contiene un componente botón. El diseño del framework permite navegar a través del árbol de componentes para acceder a sus propiedades. Además, los componentes pueden reaccionar a diferentes eventos y son capaces de almacenar su estado interno.

Un aspecto fundamental de JSF es el ciclo de vida de una petición web. El ciclo de vida está separado en 6 fases principales

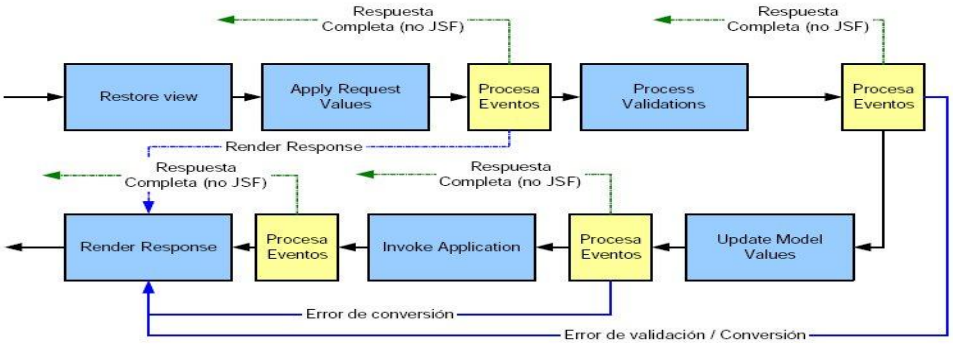


Figura. Ciclo de vida JSF[13]

- 1. Restore View: Crea el arbol de componentes de la página solicitada y carga el estado si esta ya había sido solicitada previamente.
- 2. Apply Request Value: Itera sobre el arbol de componentes recuperando el estado de cada uno asignándole los valores que viene desde el cliente.
- 3. Process Validations: Se realizan las validaciones de cada componente
- 4. Update Model Values: Se actualizan los valores de los backing beans del modelo cuyas propiedades estaban vinculadas a propiedades de los componentes de la vista.
- 5. Invoke application: Se ejecuta la lógica del negocio y se selecciona la próxima vista lógica.
- 6. Render Response: Se arma la vista con el estado actualizado de los componentes y se la envía al cliente.

Es posible generar, en cualquier momento, una respuesta que no tenga componentes framework y, por lo tanto, no necesite rendering, como por ejemplo una redirección a una página html estática.

Otro concepto importante es el de los beans administrados. JSF provee un contenedor de inversión de control para administrar los beans que realizan el binding entre la vista y el modelo. En estos beans se recuperan los valores ingresados una vez que estos fueron validados, es decir, en la fase Update Model Values. También se puede definir funcionalidad para controlar el flujo de alguna página o invocar los servicios de la capa de negocio.

Desarrollar con JSF permite:

- Lógica de navegación entre páginas
- Binding entre la vista y los beans de negocio
- Manejo de eventos
- Internacionalización
- Validación de entradas
- Independencia del dispositivo de presentación
- Construcción de componentes

3.6.4. ASP.NET WebForms

ASP.NET es un conjunto de tecnologías definidas por Microsoft para la capa de presentación WEB que forma parte del .NET Framework. En pocas palabras, una página ASP.NET es un archivo de texto con extensión aspx que el servidor sabe que debe procesar de una manera especial. El texto de las páginas puede ser html junto con código scripting que se compila dinámicamente y se ejecuta en el servidor. La página aspx se compila (sólo la primera vez) a código ejecutable .net cuando algún cliente la solicita al servidor. Para incluir código embebido en la página se utilizan los separadores <% y %>. En este sentido es similar al funcionamiento de las páginas JSP de java. Sin embargo la potencia de este framework no reside en estas características sino en las que se describen a continuación.

Las páginas ASP.NET pueden tener controles que se ejecutan del lado del servidor (server controls) que son objetos que representan elementos de la interfaz de usuario que se ejecutan en el servidor y generan código html como resultado de su ejecución. Los controles tienen propiedades, métodos y eventos a los que pueden responder y mediante los que se puede modificar su estado y comportamiento. Este comportamiento se puede declarar en los atributos de su declaración html o de manera programática.

Los controles permiten contener otros controles dentro de ellos y es posible, al igual que cualquier objeto, heredar y redefinir parte de su comportamiento.

Un control de servidor se identifica en una página html por su atributo runat="server". De esta manera un webform es una página html que contiene en algún lado una etiqueta del estilo:

```
<form runat=”server”>
    …
 </form>
```

Trabajar con ASP.NET permite

- Separación del html y el código .NET.
- Binding entre los elementos de la vista y el código .net
- Validación de entradas
- Manejo de eventos
- Creación de componentes propios
- Internacionalización (*)
- Maquetación (*)

Los ítems marcados con (*) sólo están disponibles en ASP.NET 2.0

3.6.5. Cocoon.

Cocoon es un framewok creado en 1999 por Stefano Mazzocchi. Está desarrollado en java y es completamente diferente a los que se detallaron anteriormente. Se basa en un modelo de componentes y en el concepto de tuberías (pipelines) de componentes. Este concepto se asemeja a una línea de producción donde cada componente se especializa en una operación en particular (existen componentes generadores, transformadores, serializadores, etc). El contenido ingresa en la tubería y es modificado por las sucesivas etapas hasta generar la respuesta que es enviada al cliente. De esta manera se busca separar el contenido, el estilo, la lógica y la administración de un sitio web basado en contenido XML.

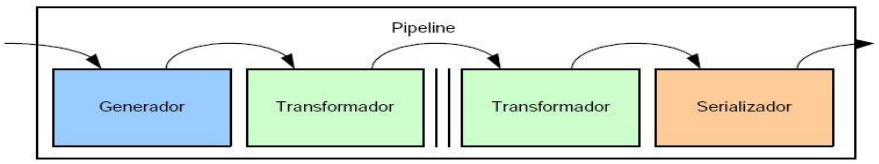


Figura. Pipeline[13]

El elemento global que utiliza para definir una aplicación se denomina sitemap. Este sitemap incluye todo lo que tendrá el sitio web y es configurado a través del archivo xml sitemap.xmap. En este archivo se especifican los componentes, las vistas, los recursos, los conjuntos de acciones y las tuberías que tendrá la aplicación.

Los componentes pueden ser:

- Generadores (generators): generan XML como eventos SAX y dan inicio al procesamiento del pipeline
- Transformadores (transformers): transforman los eventos SAX recibidos en otros eventos SAX
- Serializadores (serializers): transforman los eventos SAX en streams binarios o de texto que pueden ser entendibles por el cliente.
- Selectores (selectors): permiten implementar lógica condicional básica (if-then o switch)
- Mapeadores (matchers): mapea un patrón de entrada con un recurso
- Acciones (actions): son invocadas desde el pipeline y ejecutan algun tipo de operación antes de continuar con el resto del proceso. Las acciones tienen un método act() en donde ejecutan su lógica y devuelven un conjunto de resultados en forma de diccionario (map). Generalmente son las encargadas de invocar a la lógica del negocio.

Construir aplicaciones con Cocoon permite

- Control de flujo de navegación
- Separación de incumbencias
- Internacionalización
- Independencia del dispositivo de presentación
- Validación de entradas

3.6.6. Ruby on Rails

Ruby on Rails (ROR) es un framework para desarrollar aplicaciones webs basado en el lenguaje Ruby. En realidad no sólo aplica para la capa de presentación sino que es posible definir desde la lógica de navegación hasta el acceso a datos. ROR busca simplificar el desarrollo de aplicaciones promoviendo las convenciones sobre la configuración. De hecho, el framework no posee archivos de configuración (en realidad posee uno pero es para temas generales como dirección de la base de datos, etc).

Para crear una aplicación con ROR se utiliza un script que genera toda la estructura base con los directorios donde se deben situar los diferentes archivos de código. Por

ejemplo, hay directorios para el modelo, la vista y el controlador ya que ROR también está basado en el patrón MVC.

En las vistas, el código ruby es embebido en el código html de manera similar a jsp/php/asp de manera que no se logra una separación total entre el html y el código de implementación. Sin embargo, la lógica de la aplicación, las reglas del negocio, se ejecutan en código ruby puro accedidas a través de los controladores.

Desarrollar aplicaciones con ROR permite

- Mapeo transparente de URLs a métodos
- Mapeo transparente objeto-relacional
- Desarrollo rápido de aplicaciones CRUD

3.7. Comparativa de los frameworks.[13]

Struts:

Pros:

- Más de 6 años demostrando que funciona. Gran cantidad de desarrollos de gran envergadura concretados exitosamente.
- Es el framework más popular de la comunidad java por lo que existen infinidad de material disponible en la web. Buenas prácticas conocidas.
- Documentación muy buena
- Permite crear sitios internacionales de manera rápida y efectiva.
- Curva de aprendizaje mediana.
- Open Source (Licencia Apache)

Contras:

- No abstrae completamente al desarrollador del funcionamiento del protocolo http.
- Aunque se adapta a las incorporaciones de diferentes bibliotecas de tags no está diseñado para facilitar la creación de componentes propios
- No es natural el mapeo de los datos ingresados a los objetos del negocio.
- Las vistas quedan atadas al dispositivo en el cual se renderizan. No facilita el armado de vistas independientes del dispositivo.

- No es una especificación.

Tapestry

Pros:

- Open Source (Licencia Apache)
- Permite el desarrollo de componentes propios
- Los diseñadores web no necesitan aprenderse tags nuevos ni diferentes lenguajes ya que los templates se codifican en html estándar.
- La creación de componentes es relativamente sencilla.
- Separación completa entre la lógica y la presentación (html de java)

Contras:

- Comunidad de desarrolladores pequeña-mediana.
- Escasa documentación. Pocos libros editados. Poca información en la web.
- Se requiere configurar 3 archivos para cada página a crear.

ASP.NET

Pros:

- Curva de aprendizaje baja
- Permite el desarrollo de controles propios y utilizar controles de terceros
- Gran comunidad de desarrolladores
- Soporte oficial y amplia documentación.
- Permite binding directo entre los componentes y los orígenes de datos.
- Permite desarrollo con herramientas RAD.

Contras:

- Propietario de Microsoft. Sólo funciona con Information Server (*).
- Requiere un IDE como Visual Studio para un desarrollo productivo. Lo que deviene en un costo por desarrollador por el licenciamiento del IDE.
- El control de navegación no está centralizado.
- Código cerrado. Ante la aparición de bugs dentro del framework se depende de Microsoft para solucionarlo.

- Varias de las funcionalidades importantes (maquetación, internacionalización) sólo están disponibles a partir de la versión 2.0.
- Requiere javascript y cookies para funcionar correctamente.
- El estado interno de la vista (viewstate) viaja codificado dentro de un campo hidden. Esto trae problemas de performance y si se utiliza mal, problemas de seguridad.

(*) Apache cuenta con un módulo para la implementación libre de ASP.NET que corre bajo el proyecto Mono.

Cocoon

Pros:

- Permite separar claramente el contenido de la presentación y de la lógica.
- Open Source (Licencia Apache).
- Permite modificar el comportamiento de la aplicación sin conocer el lenguaje en el que está implementado.

Contras:

- Requiere amplios conocimientos de hojas de estilo XSL por parte de los diseñadores gráficos.
- Comunidad relativamente pequeña.
- Curva de aprendizaje elevada.
- La transformación de XMLs requiere bastante capacidad de proceso.

JSF

Pros:

- Permite separar claramente el contenido de la presentación y de la lógica.
- Es una especificación, lo que permite tener varias implementaciones (tanto de código cerrado como de código abierto).
- Permite modificar el comportamiento de la aplicación sin conocer el lenguaje en el que está implementado.
- No es necesario conocer el framework en detalle para poder comenzar a utilizarlo.

- Comunidad y herramientas de soporte en aumento.

Contras:

- La creación de componentes propios es compleja.
- Requiere javascript.

Ruby on Rails

Pros:

- Alta productividad para desarrollar aplicaciones de tipo CRUD.
- Solución TODO en 1. Desde la presentación hasta la persistencia.
- Es posible mantener ambientes separados de prueba y producción
- No necesita configuración (al menos no mucha).
- Gran aceptación en la comunidad de desarrolladores.

Contras:

- Aún no existe constancia de aplicaciones de gran envergadura desarrolladas con este framework más allá de varias aplicaciones Web masivas.
- Utiliza lenguaje interpretado y débilmente tipado, difícil de depurar.

3.8. Framework de la nueva generación[13]

3.8.1. DWR

DWR es una librería open source que permite incorporar tecnología AJAX a las aplicaciones WEB desarrolladas en Java. Permite que el código del navegador web use funciones java que se ejecutan en el servidor como si se ejecutaran desde el navegador. DWR está separado en 2 partes principales: Por un lado existe una parte que consiste en código javascript que se encarga de la comunicación con el servidor y de la actualización dinámica de la página. Por otro lado, en el servidor hay un servlet que recibe los requests provenientes de este código javascript, los procesa y devuelve la respuesta.

La utilización de bibliotecas como DWR tiene grandes beneficios sobre la implementación de frameworks AJAX propios. El principal es la portabilidad.

Generalmente el JavaScript utilizado tiene funciones que son dependientes de los navegadores y que, por lo tanto, generan cierta incompatibilidad entre plataformas. El uso de un motor AJAX permite abstraer estas complicaciones y utilizar las funciones provistas por el framework dejando que este se ocupe de las cuestiones particulares de cada navegador.

DWR genera dinámicamente código JavaScript basado en las clases de java. Así, el programador puede acceder a las clases java desde el cliente a través del JavaScript.

DWR no funciona con todos los navegadores pero si lo hace con los de última generación. Además tiene facilidades para integrarse con otros frameworks como spring, struts, hibernate, etc.

3.8.2. OpenLaszlo

OpenLaszlo es una plataforma de desarrollo de Aplicaciones de Internet Ricas (RIA) basado, principalmente, en Flash como tecnología de presentación, aunque aseguran que es independiente del producto de Macromedia. El framework fue inicialmente diseñado por Laszlo Systems de manera cerrada pero, a partir de octubre de 2004 se lo liberó con licencia libre.

NOTA: En la versión 4 de Laszlo (actualmente en beta) aparece una alternativa a la renderización de componentes en formato Flash y es que ahora existe la posibilidad de que los mismos sean renderizados en DHTML lo cual brinda una gran flexibilidad al no necesitar depender del plugin de Macromedia para poder ejecutar nuestra aplicación.

Las aplicaciones desarrolladas con openlaszlo se escriben en un lenguaje llamado LZX que combina XML, XPATH y JavaScript. Es un lenguaje basado en eventos y orientado a objetos. El LPS (Laszlo Presentation Server) se encarga de recibir las peticiones de los clientes y genera contenido en flash permitiendo desarrollar interfaces de gran interactividad y alto impacto visual en el cliente. Cabe destacar que sólo la primera vez se compila el archivo lzx a flash

Las primeras aplicaciones de uso masivo están apareciendo por estos días y, aunque el resultado visual es impactante, por el momento es necesario tener el plugin de flash instalado en el cliente lo que quita muchos de los beneficios de las aplicaciones web

como la independencia de la presentación del contenido, accesibilidad para que puedan leer las personas con capacidades diferentes, etc.

Capítulo IV: Patrones

4.1. Definición[14]

- Los patrones son soluciones de sentido común que deberían formar parte del conocimiento de un diseñador experto. Además facilitan la comunicación entre diseñadores, pues establecen un marco de referencia (terminología, justificación).
- Cada patrón describe un problema que ocurre una y otra vez en nuestro entorno, para describir después el núcleo de la solución a ese problema, de tal manera que esa solución pueda ser usada más de un millón de veces sin hacerlo ni siquiera dos veces de la misma forma.

  Patrón = < problema, contexto, solución> +<recurrencia, enseña, nombre>

- Un patrón es un modelo que podemos seguir para realizar algo. Los patrones surgen de la experiencia de seres humanos de tratar de lograr ciertos objetivos.
- Los patrones capturan la experiencia existente y probada para promover buenas prácticas.

Ventajas:

- ✓ Facilitan la comunicación interna
- ✓ Ahorran tiempo y experimentos inútiles
- ✓ Mejoran la calidad del diseño y la implementación
- ✓ Son como “normas de productividad”
- ✓ Facilitan el aprendizaje de los paquetes de Java
- ✓ Ayudan a construir la experiencia colectiva de Ingeniería de Software.
- ✓ Son una abstracción de "problema – solución".
- ✓ Se ocupan de problemas recurrentes.
- ✓ Identifican y especifican abstracciones de niveles más altos que componentes o clases individuales.
- ✓ Proporcionan vocabulario y entendimiento común

4.2. Características.

Solucionar un problema: Los patrones capturan soluciones, no sólo principios o estrategias abstractas.

Ser un concepto probado: capturan soluciones demostradas, no teorías o especulaciones.

La solución no es obvia: los mejores patrones generan una solución a un problema de forma indirecta

Describe participantes y relaciones entre ellos: describen módulos, estructuras del sistema y mecanismos complejos.

El patrón tiene un componente humano significativo: todo software proporciona a los seres humanos confort y calidad de vida (estética y utilidad)

4.3. Clases.

Patrones de arquitectura*:* se expresa una organización o esquema estructural fundamental para sistemas software. Proporciona un conjunto de subsistemas predefinidos y sus responsabilidades.

Patrones de diseño*:* proporciona esquemas para refinar subsistemas o componentes de un sistema.

Patrones de programación*:* describe la implementación de aspectos de componentes.

Patrones de análisis*:* prácticas que aseguran la consecución de un buen modelo de un problema y su solución.

Patrones organizacionales: describen la estructura y prácticas de las organizaciones humanas.

4.4. Patrones de diseño.

¿Qué son los patrones de diseño?

Son soluciones simples y elegantes a problemas específicos y comunes del diseño orientado a objetos. Son soluciones basadas en la experiencia y que se ha demostrado que funcionan.

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

Un patrón de diseño es una solución a un problema de diseño no trivial que es efectiva (ya se resolvió el problema satisfactoriamente en ocasiones anteriores) y reusable (se puede aplicar a diferentes problemas de diseño en distintas circunstancias).

En la programación orientada a objetos resulta complicado descomponer el sistema en objetos (encapsulación, granularidad, dependencias, flexibilidad, reusabilidad, etc.), los patrones de diseño nos permitirán identificar a los objetos apropiados de una manera mucho más sencilla. También nos permitirán determinar la granularidad de los objetos.

Además, los patrones de diseño, también nos ayudarán a especificar las interfaces, identificando los elementos claves en las interfaces y las relaciones existentes entre distintas interfaces. De igual modo nos facilitará la especificación de las implementaciones.

También, y de forma casi automática, nos ayudan a reutilizar código, facilitando la decisión entre "herencia o composición" (favorece la composición sobre la herencia y hace uso de la delegación).

Relacionan estructuras en tiempo de compilación y en tiempo de ejecución. Y nos permiten hacer un diseño preparado para el cambio.

Los diseñadores expertos no resuelven cada problema desde el principio:

- ✓ Se pueden encontrar patrones de clases y comunicaciones entre objetos en muchos sistemas orientados a objetos.
- ✓ Los patrones de diseño ayudan a los diseñadores a reutilizar con éxito diseños para obtener nuevos diseños.

- ✓ El objetivo de los patrones es guardar la experiencia en diseños de programas orientados a objetos.
- ✓ Los patrones de diseño hacen más fácil reutilizar con éxito los diseños y arquitecturas, ayudan a elegir entre diseños alternativos, hacen a un sistema reutilizable y evitan alternativas que comprometen la reutilización.

Elementos Característicos de un Patrón de Diseño

Un patrón de diseño tiene cuatro elementos característicos:

1. El nombre del patrón, describe el problema de diseño, su solución, y consecuencias en una o dos palabras. Tener un vocabulario de patrones nos permite hablar sobre ellos.
2. El problema describe cuando aplicar el patrón. Se explica el problema y su contexto. Puede describir estructuras de clases u objetos que son sintomáticas de un diseño inflexible. Se incluye una lista de condiciones.
3. La solución describe los elementos que forma el diseño, sus relaciones, responsabilidades y colaboraciones. No se describe un diseño particular. Un patrón es una plantilla.
4. Las consecuencias son los resultados de aplicar el patrón

Descripción de un Patrón de Diseño

- ❖ Nombre
- ❖ Objetivo (Problema)
- ❖ Contexto
- ❖ Aplicabilidad (Fuerzas)
- ❖ Solución
- ❖ Consecuencias
- ❖ Implementación
- ❖ Usos en el API de Java
- ❖ Código del ejemplo
- ❖ Patrones relacionados

Cualidades de un Patrón de Diseño

Encapsulamiento y abstracción. Cada patrón encapsula un problema bien definido y su solución en un dominio particular.

Extensión *y* ***variabilidad****.* Cada patrón debería ser abierto por extensión o parametrización por otros patrones, de tal forma que pueden aplicarse juntos para solucionar un gran problema.

Generatividad *y* ***composición****.* Cada patrón una vez aplicado genera un contexto resultante, el que concuerda con el contexto inicial de uno o más de uno de los patrones del catálogo.

Equilibrio*.* Cada patrón debe realizar algún tipo de balance entre sus efectos y restricciones

Un patrón describe un todo que es más grande que la suma de sus partes

4.4.1. Tipos de patrones de diseño.

4.4.1.1. De Creación

Los patrones de creación abstraen la forma en la que se crean los objetos, permitiendo tratar las clases a crear de forma genérica dejando para más tarde la decisión de qué clases crear o cómo crearlas. Estas decisiones normalmente serán resueltas dinámicamente decidiendo que clases instanciar o sobre que objetos un objeto delegará responsabilidades

Según donde se tome dicha decisión podemos clasificar a los patrones de creación en *patrones de creación de clase* (la decisión se toma en los constructores de las clases y usan la herencia para determinar la creación de las instancias) y *patrones de creación de objeto (se modifica la clase desde el objeto)*.

Patrones de Creación:

Abstract Factory: Proporciona una interfaz para crear familias de objetos relacionados o dependientes sin especificar su clase concreta

Builder: Permite a un objeto construir un objeto complejo especificando sólo su tipo y contenido.

Factory Method: Define una interfaz para crear un objeto dejando a las subclases decidir el tipo específico al que pertenecen.

Prototype: Permite a un objeto crear objetos personalizados sin conocer su clase exacta a los detalles de cómo crearlos

Singleton: Garantiza que solamente se crea una instancia de la clase y provee un punto de acceso global a él.

Estructurales

Tratan de conseguir que cambios en los requisitos de la aplicación no ocasionen cambios en las relaciones entre los objetos. Lo fundamental son las relaciones de uso entre los objetos, y, éstas están determinadas por las interfaces que soportan los objetos. Estudian cómo se relacionan los objetos en tiempo de ejecución. Sirven para diseñar las interconexiones entre los objetos.

Patrones Estructurales:

Adapter: convierte la interfaz que ofrece una clase en otra esperada por los clientes

Bridge: desacopla una abstracción de su implementación y les permite variar independientemente.

Composite: permite gestionar objetos complejos e individuales de forma uniforme

Decorator: extiende la funcionalidad de un objeto dinámicante de tal modo que es transparente a sus clientes.

Façade: simplifica los accesos a un conjunto de objetos relacionados proporcionando un objeto de comunicación.

Flyweight: usa la compartición para dar soporte a un gran número de objetos de grano fino de forma eficiente.

Proxy: proporciona un objeto con el que controlamos el acceso a otro objeto.

De Comportamiento

Los patrones de comportamiento estudian las relaciones entre llamadas entre los diferentes objetos, normalmente ligados con la dimensión temporal. Se utilizan para organizar, manejar y combinar comportamientos

Patrones de Comportamiento:

Chain of Responsability: evita el acoplamiento entre quien envía una petición y el receptor de la misma. Crea una cadena con los objetos receptores y pasa la petición a través de la cadena hasta que esta sea tratada por algún objeto.

Command. Encapsula una petición en un objeto, permitiendo así parametrizar a los clientes con distintas peticiones, encolar o llevar un registro de las peticiones y poder deshacer las operaciones.

Interpreter. Dado un lenguaje, define una representación de su gramática junto con un intérprete que usa dicha representación para interpretar las sentencias del lenguaje.

Iterator. Proporciona un modo de acceder secuencialmente a los elementos de un objeto agregado sin exponer su representación interna.

Mediator. Define un objeto que encapsula cómo interactúan un conjunto de objetos. Promueve un bajo acoplamiento al evitar que los objetos se refieran unos a otros explícitamente, y permite variar la interacción entre ellos de forma independiente.

Memento. Representa y externaliza el estado interno de un objeto sin violar la encapsulación, de forma que éste puede volver a dicho estado más tarde.

Observer. Define una dependencia de uno-a-muchos entre objetos, de forma que cuando un objeto cambia de estado se notifica y actualizan automáticamente todos los objetos.

State: modifica el comportamiento de un objeto cuando su estado interno cambia. Parecerá que cambia la clase del objeto.

Strategy: define una familia de algoritmos, encapsula cada uno y los hace intercambiables. Permite que un algoritmo varíe independientemente de los clientes que lo usan.

Template Method: define un esqueleto de algoritmo y delega partes concretas de un algoritmo a las subclases. Permite que las subclases redefinan ciertos pasos del algoritmo sin cambiar su estructura.

Visitor: representa una operación que será realizada sobre los elementos de una estructura de objetos, permitiendo definir nuevas operaciones sin cambiar las clases de los elementos sobre los que opera

4.5. Patrones que se utilizan en las Aplicaciones Web[15]

Los Patrones de Diseño,[15] en su forma más sencilla, son buenas soluciones que pueden ser aplicadas a problemas recurrentes que surgen durante el diseño de un sistema o aplicación.

La idea es interesante, y para aquellos que se están iniciando en desarrollo de aplicaciones para web representa un buen compendio para arrancar, un tanto clásico tal vez, pero es *trabajo en progreso*. Estos patrones están organizados en:

- Tipos de Sitios
- Experiencia del Usuario
- Ecommerce
- Navegación
- Búsqueda
- Tipos Básico de Página
- Administración de Colecciones
- Elementos de Página
- Interacciones Básicas

4.6 Patrón Modelo Vista Controlador (MVC)[16]

Este patrón se usa en aplicaciones interactivas que requieren una interfaz de usuario flexible.

Este patrón es adecuado para situaciones donde:

Es necesario presentar los mismos datos de diferentes formas (gráficos circulares, gráficos de barras, etc.) o utilizando diferentes interfaces de usuario (Motif, Windows 95, etc.). Si cambia un dato, los datos se deben de actualizar en todas las representaciones.

El interfaz de usuario debe ser fácil de modificar.

El principal objetivo de la arquitectura MVC es aislar tanto los datos de la aplicación como el estado (modelo) de la misma, del mecanismo utilizado para representar (vista) dicho estado, así como para modularizar esta vista y modelar la transición entre estados del modelo (controlador).

El modelo MVC es un patrón de arquitectura creado por gente de la comunidad Smalltalk que consiste en descomponer una aplicación interactiva en tres componentes principales

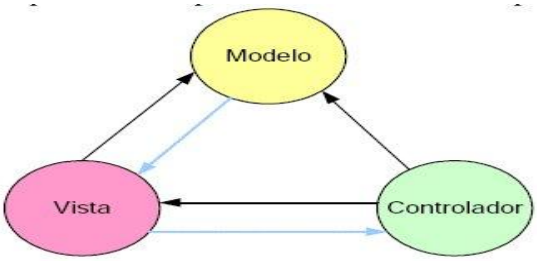


Figura: Modelo Vista Controlador[17]

- El controlador, que es el que único que puede recibir acciones del los usuarios externos. Está asociado a cada vista, recibe entradas que traduce en invocaciones de métodos del Modelo o de Vista. El usuario interactúa con el sistema solamente vía controladores.
- El modelo, que posee el estado interno de la aplicación (determinado por el estado de sus entidades) y las reglas del negocio. Encapsula los datos y la funcionalidad de la aplicación, es independiente de la representaron de los datos.
- Las vistas que reflejan el estado algún aspecto del estado del modelo. Despliega la información contenida en el modelo (pueden existir varias vistas). Cada vista tiene un controlador asociado.

El propósito del MVC es aislar los cambios. Es una arquitectura preparada para los cambios, que desacopla datos y lógica de negocio de la lógica de presentación, permitiendo la actualización y desarrollo independiente de cada uno de los citados

3. DISEÑO METODOLÓGICO

TIPO DE ESTUDIO

Para el presente proyecto se utilizará el **método explorativo,** el mismo que nos va a permitir relacionarnos con el problema central que existe en el entorno Web esto con el fin de dar mejoras para el aprendizaje de los estudiantes.

Además se usará el **método descriptivo,** puesto que nos permitirá realizar la interpretación y análisis de la información recolectada acerca de los frameworks, formularios Web, prototipo de pantallas y la tecnología de ajax los mismos que intervienen directamente para el desarrollo del framework.

Para la recolección de la información relacionada a las actividades, y posibles alternativas de solución referentes al desarrollo del framework nos ayudaremos del **método deductivo,** el mismo que nos permitirá abstraer las partes más relevantes de la información obtenida en el proceso de investigación.

También utilizaremos el **método analítico**, este nos permitirá realizar el análisis e interpretación de la información más relevante que se ha obtenido.

POBLACION, MUESTRA Y UNIDADES DE OBSERVACION

Dentro de nuestro proyecto, la población y muestra se tomada de un grupo de estudiantes y docentes de la carrera de ingeniería en sistemas.

Las unidades de observación están centradas en la generación de formularios en la Web y la creación de prototipo de pantallas, para la realización de las mismas se utilizara las técnicas de encuestas y entrevistas.

MÉTODOS E INSTRUMENTOS

Para el desarrollo de la presente investigación, utilizaremos los siguientes métodos:

- **MÉTODO CIENTÍFICO:**

Partiremos del análisis de la información recolectada que nos permitirá conocer la realidad de los avances tecnológicos y de esta manera poder cumplir los objetivos planteados.

- **MÉTODO EXPERIMENTACION CIENTIFICA:**

En cuanto a la experimentación científica, algunos de nuestros conocimientos los proporciona la experiencia y es un método que permite sentirse más seguro de los que se está haciendo, además admite la modificación de variables, lo cual da vía libre para la corrección de errores y el mejoramiento de la investigación.

La aplicación de este método será primordial, debido a que nos ayudara a buscar solución de calidad, efectividad, funcional y de satisfacción a las necesidades del usuario.

Instrumentos:

- **Técnica de Entrevista:**

Está técnica permitirá el contacto interpersonal, tiene por objeto la provisión de información por medio de testimonios orales para la cual se requiere la preparación de cuestionarios con lo que lograremos obtener datos reales para el desarrollo del proyecto.

Las entrevistas serán elaboradas de acuerdo a las personas entrevistas, como son:

- Cuestionario para los docentes
- Cuestionario para los estudiantes.

- **Técnica de Encuesta:**

Esta técnica permitirá obtener una información concreta expresada en forma escrita por los encuestados.

Las encuestas serán elaboradas, para los docentes y estudiantes de la carrera de Ingeniería en Sistemas de la Universidad Nacional de Loja.

PROCEDIMIENTOS

Identificación de Procedimientos:

Aquí se dará a conocer el procedimiento que se seguirá para la aplicación de las técnicas de recolección de información seleccionadas.

Cronograma de Trabajo o Calendario de Actividades

Es un instrumento de precisión y control de tiempo a emplearse en la normal realización de las diferentes actividades del proceso de investigación. Su elaboración

consiste en relacionar en forma coherente y sistemática las actividades que se van a desarrollar en la investigación con los tiempos previstos en el cumplimiento de cada una de ellas.

PLAN DE TABULACIÓN Y ANÁLISIS DE LA INFORMACIÓN

Para el presente proyecto en el análisis y la facilitación de la interpretación de los datos se utilizará cuadros estadísticos conjuntamente con los gráficos correspondientes.

ELABORACIÓN O REDACCIÓN DEL INFORME Y DE LAS ALTERNATIVAS DE SOLUCIÓN

El proyecto se documentará de la siguiente manera:

Investigación Preliminar:

- **Análisis de la Investigación Preliminar,** el que tiene los siguientes ítems:
  - Definición del proyecto
  - Objetivos del proyecto
  - Estudio de factibilidad.
  - Alcance del proyecto.
  - Documento de requerimientos.
  - Diccionario de términos.
- **Diseño de la aplicación:**
  - Diagramas Use Case.
  - Modelo del dominio (Diagrama UML).
  - Use case de bajo nivel.
  - Diagramas de robustez.
  - Diagramas de secuencia.
  - Diagramas de paquetes.
  - Modelo entidad-relación
  - Diseño de la base de datos
- **Construcción**
  - Prototipo de pantallas
  - Programación de la Aplicación
  - Implementación y Pruebas

- **Documentación de la aplicación**
    - Manual de instalación.
    - Manual de usuario.
    - Manual de programador.
    - Manual de administrador.

4. ORGANIZACIÓN Y GESTIÓN DE LA INVESTIGACIÓN

RECURSOS

Recursos Humanos

- Personal entrevistado y encuestado(docentes y estudiantes)
- Asesor y Director de Tesis
- Grupo de Investigación:

    Lady Rosillo

    Oscar Guamán

Recursos Económicos

Se realizará una identificación claramente de todos los egresos que ocasionará cada una de las fases del proceso investigativo. (Gasto en material bibliográfico, gastos en materiales y suministros de escritorio, transporte, alimentación, además se debe considerar una cuenta de imprevistos para cubrir un gasto no previsto)

Recursos Materiales

Aquí desglosaremos todos los recursos físicos; materiales y suministros que faciliten la realización de la investigación.

En el desarrollo de la investigación se requerirán de:

- Material y Suministros
- Anillados-Empastados
- Transporte
- Consultas (Internet)
- Impresión de información

Recursos Técnicos

Hardware

- 1 Computadora de Escritorio
- 1 Laptop
- 2 Impresora
- 2 Flash Memory

Software

- Microsoft Office
- Netbeans
- Enterprise Architect

Recursos Tecnológicos

Para el presente proyecto se utilizarán herramientas de tecnología las mismas que nos contribuirán para un desarrollo rápido, eficiente del sistema y a la vez sea amigable para el usuario

Las herramientas a utilizarse serán:

NetBeans, las características de esta herramienta son suficientes para cubrir los aspectos técnicos necesarios para el correcto desarrollo y posterior implementación de Sistema. Dentro de esta herramienta se utiliza JAVA, debido a que este lenguaje se lo está utilizando en la actualidad, además provee un modelo de programación consistente, ejecución multiplataforma, seguridad de tipos de datos.

Enterprise Architect - Este software nos permitirá la creación de los diferentes diagramas en UML (Lenguaje Modelado Unificado) necesarios para esta fase de diseño del sistema la cual nos permitirá trabajar de manera más sencilla y rápida.

Microsoft Office.- Este programa está conformado por World, Excel, PowerPoint, Visio, Project, los mismos que nos permitirán la elaboración de informes, el cálculo de valores con su representación específica si el caso lo amerita, la presentación del proyecto, diagramas de grantt, entre otras.

CRONOGRAMA DE ACTIVIDADES

Id	Nombre de tarea	Duración	Comienzo	Fin
1	Investigar y documentar todo lo referente a framework y prototipo de pantallas	60 días	lun 31/12/07	jue 20/03/08
2	Investigación bibliográficas	30 días	mié 23/11/07	mar 08/01/08
3	Documentación de la información más relevante	25 días	mié 09/01/08	lun 11/02/08
4	Identificar los requerimientos del usuario	15 días	mar 12/02/08	lun 03/03/08
5	Validar los requerimientos	8 días	mar 04/03/08	jue 13/03/08
6	Investigar y aprender como utilizar la tecnología Ajax	90 días	vie 14/03/08	jue 17/07/08
7	Investigacion bibliográficas	30 días	mar 22/01/08	lun 03/03/08
8	Analizar y utilizar la información para el desarrollo del framework	60 días	mar 04/03/08	lun 26/05/08
9	Modelar el framework y la herramienta para crear prototipo de pantallas utilizando el patrón MVC	45 días	mar 27/05/08	lun 28/07/08
10	Investigación y análisis de la información	30 días	mar 15/04/08	lun 26/05/08
11	Estructuración del proyecto	30 días	mar 27/05/08	lun 07/07/08
12	Diseñar el framework y la herramienta para crear prototipos de pantallas	200 días	mar 08/07/08	lun 13/04/09
13	Modelo del dominio	20 días	mar 29/07/08	lun 25/08/08
14	Definición de casos esenciales	30 días	mar 26/08/08	lun 06/10/08
15	Definición de casos expandidos	30 días	mar 07/10/08	lun 17/11/08
16	Prototipo de pantallas	60 días	mar 18/11/08	lun 09/02/09
17	Diagramas de modelado	65 días	mar 10/02/09	lun 11/05/09
18	Diseño de la Base de Datos	90 días	mar 12/05/09	lun 14/09/09
19	Construcción del framework y la herramienta para crear prototipo de pantallas	430 días	mar 15/09/09	lun 09/05/11
20	Crear una aplicación de ejemplo	90 días	mar 10/05/11	lun 12/09/11
21	Implementación del framework y la herramienta para crear prototipo de pantallas	90 días	mar 13/09/11	lun 16/01/12
22	Pruebas e implementación	30 días	mar 17/01/12	lun 27/02/12
23	Documentacion del proyecto	30 días	mar 28/02/12	lun 09/04/12

Proyecto: cronograma_Ultimo
Fecha: mié 25/04/12

Tarea		Tarea inactiva		Resumen manual	
División		Tarea inactiva		Sólo el comienzo	
Hito		Hito inactivo		Sólo fin	
Resumen		Resumen inactivo		Progreso	
Resumen del proyecto		Tarea manual		Fecha límite	
Tareas externas		Sólo duración			
Hito externo		Informe de resumen manual			

REFERENCIAS:

1. http://www.acm.org/crossroads/espanol/xrds7-4/frameworks.html#uml
2. http://www.ajaxhispano.com/diez-razones-para-usar-AJAX.html
3. http://es.wikipedia.org/wiki/AJAX
4. http://es.wikipedia.org/wiki/DHTML
5. http://es.wikipedia.org/wiki/Framework
6. http://es.wikipedia.org/wiki/Modelo_vista_controlador/
7. http://es.wikipedia.org/wiki/Wiki
8. http://www.isabelperez.com/taller1/wiki.htm
9. http://www.javahispano.org/contenidos/es/comparativa_de_frameworks_web/
10. http://www.librosweb.es/css/capitulo1/que_es_css.html
11. Patrón y patrones de Diseño. Pdf (Capitulo 4)
12. http://www.programacionweb.net/articulos/articulo/?num=318
13. http://sociedaddelainformacion.telefonica.es/jsp/articulos/detalle.jsp?elem=2146&salto=1&back=2&origen=2
14. http://www.tejedoresdelweb.com/307/article-69347.html#h2_1
15. http://www.thealphasite.org/patrones_de_diseno_introduccion
16. http://www.uberbin.net/archivos/internet/ajax-un-nuevo-acercamiento-a-aplicaciones-web.php

BIBLIOGRAFÍA

1. http://www.acm.org/crossroads/espanol/xrds7-4/frameworks.html#uml
2. http://www.ajaxhispano.com/diez-razones-para-usar-AJAX.html
3. http://www.ajaxhispano.com/que-es-ajax.html
4. http://www.ajaxhispano.com/tutorial-manual-ajax-ejemplos-metodo-GET-POST-principiantes.html
5. http://www.desarrolloweb.com/articulos/181.php
6. http://dhtmlnirvana.com/ajax/ajax_tutorial/#
7. http://dev.mysql.com/doc/refman/5.0/es/index.html
8. http://developer.mozilla.org/es/docs/AJAX:Primeros_Pasos
9. http://www.duamu.com/re/articulo/1344/id/315/articulos-formularios-web-recomendaciones-generale.html
10. http://www.elcodigo.net/cgibin/DBread.cgi?tabla=herramientas&campo=0&clave=49&info=1
11. http://www.elcodigo.net/taller/javascript/indices.html
12. http://www.elcodigo.net/tutoriales/html/html1.html#punto1
13. http://geneura.ugr.es/~maribel/xml/introduccion/index.shtml
14. http://www.infor.uva.es/~jvegas/cursos/buendia/pordocente/node10.html
15. http://www.infor.uva.es/~jvegas/cursos/buendia/pordocente/node17.html
16. http://www.librosweb.es/css/capitulo1.html
17. http://www.librosweb.es/css/capitulo1/que_es_css.html
18. http://www.programacionweb.net/articulos/articulo/?num=318
19. http://raultecnologia.wordpress.com/2006/12/04/historia-de-la-tecnologia.html
20. http://www.webestilo.com/mysql/
21. http://www.wikilearning.com/introduccion-wkccp-3877-1.htm
22. http://www.wikilearning.com/tutorial_de_ajax_asynchronous_javascript_+_xml-wkccp-6455-1.htm

ANEXOS

Plan Operativo

17. Presupuestos

Descripción	Cantidad	# Horas	V/Unitario	V/Total
Recursos Humanos:				
Analista, Diseñador, Programador (Lady Rosillo, Oscar Guamán)	2	Duración del Proyecto		
Director de Tesis	1			
Recursos Materiales:				
Material y Suministros:	4 resmas		3.75	
- Hojas	1		3.00	15.00
- Grapadora	1		3.00	3.00
- Perforadora	2		0.25	3.00
- Borrador	4		0.25	0.50
- Lápices, Esferos	2		50.00	1.00
- Calculadora	1 caja		0.50	100.00
- Clips	5		0.35	0.50
- Cds en blanco	8		3.10	1.75
- Cartuchos de Tinta Negra	4		3.50	24.80
- Cartuchos de Tinta de Color	15		1.00	14.00
Anillados	8		7.00	15.00
Empastados	1600 viajes		0.25	56.00
Transporte				
Consultas (Internet)				
Imprevistos				

		625	0.80	400.00 500.00 200.00
Recursos Técnicos: Hardware: - Computadora de Escritorio - Laptop - Impresora Canon Ip1000 - Impresora Canon Ip2500 - Flash Memory Voyager 1GB Software: - Windows XP Media Center Edition 2005 - Windows XP version 2002 Service Pack 2 - Microsoft Office - Netbeans - Enterprise Architect Comunicación: - Telefonía Celular - Telefonía Convencional	1 1 1 1 2 1 1 1 1 1 1 2 2	14 100	1.050 1.500 45.00 50.00 25.00 Propia 138.00 130.00 Gratuita 137.00 10.00 0.60	1.050,00 1.500,00 45.00 50.00 50.00 138.00 130.00 137.00 280.00 120.00
TOTAL:				**4.834,55**

MATRIZ DE CONSISTENCIA GENERAL

"Las herramientas que se pueden utilizar en la actualidad para el desarrollo de aplicaciones Web no son en su mayoría de fácil uso y no tienen una interacción dinámica con el usuario**"**				
TEMA	**PROBLEMA GENERAL**	**OBJETO DE INVESTIGACION**	**OBJETIVO GENERAL**	**OBJETIVOS ESPECIFICOS**
Construcción de un Framework para desarrollo de aplicaciones Web 2.0 con acceso a base de datos.	Las herramientas que se usan para el desarrollo de aplicaciones Web son complejas y debido a este factor consume mucho tiempo en el proceso de desarrollo de las mismas,	Diseñar un framework para la creación dinámica de Aplicaciones Web.	Construir un framework para el desarrollo de aplicaciones Web2.0 con acceso a base de datos MySql	Implementar el patrón Modelo Vista Controlador en el diseño del Framework. Diseñar el Framework utilizando la tecnología AJAX que se encuentra dentro de la Web2.0 Diseñar los componentes básicos que ofrecerá el framework. Permitir que las aplicaciones Web basadas en el Framework tengan el acceso a la base de Datos MySql. Las aplicaciones se almacenaran en un archivo XML. Implementación del Framework Crear una aplicación de Ejemplo basado en el framework desarrollado.

MATRIZ DE CONSISTENCIA ESPECÍFICA

OBJETIVO ESPECIFICO	PROBLEMA ESPECIFICO	ALTERNATIVA DE SOLUCION	FUNDAMENTACION CIENTIFICA-CATEGORIAL
Implementar el patrón Modelo Vista Controlador en el diseño del Framework.	Existen patrones de diseño que no son adecuados para este tipo de aplicaciones.	Elegir para el diseño del framework un patrón que se use en aplicaciones interactivas.	Información obtenida en la formación académica y consultas bibliográficas: libros e Internet. Analizar la información Obtenida y aplicarla en el desarrollo del proyecto
Diseñar el Framework utilizando la tecnología AJAX que es parte de la Web2.0	Los framework y herramientas existentes consumen mucho tiempo en el desarrollo de aplicaciones No existe un completo conocimiento acerca de la utilización de la Tecnología Ajax. Que es parte de la Web2.0.	Desarrollar el framework utilizando la tecnología Ajax para optimizar el tiempo en el proceso de desarrollo de aplicaciones.	Información obtenida en la formación en el proceso de análisis del proyecto. Información obtenida en la formación académica y consultas bibliográficas: libros e Internet.

Diseñar los componentes básicos que ofrecerá el framework.	Los framework y herramientas existentes son muy complejas	Desarrollar el framework que permita desarrollar las aplicaciones las mismas que podrán contener los componentes básicos.	Información obtenida en el proceso de análisis del proyecto.
Permitir que las aplicaciones Web basadas en el Framework tengan el acceso a la base de Datos MySql.	Los framework existentes no todos permiten realizar las operaciones con base de datos.	Desarrollar el framework que permita el acceso a la base de datos MySql.	Información sobre gestión de base de datos MySql.
Las aplicaciones se almacenaran en un archivo XML.	Los framework existentes no todos permiten guardar las aplicaciones en un archivo.	Permitirá guardar las aplicaciones dentro de un archivo XML	Información sobre archivos XML
Implementación del Framework	No todas las aplicaciones que se realizan en los proyectos de tesis son implementadas.	Implementar el Framework	Documentación realizada en el diseño del proyecto
Crear una aplicación de Ejemplo basado en el framework desarrollado.	Utilización del framework		Documentación del Framework.

MATRIZ DE OPERATIVIDAD DE OBJETIVOS

OBJETIVO ESPECIFICO: Diseñar el Framework utilizando la tecnología AJAX que es parte de la Web2.0

ACTIVIDADES O TAREA	METODOLOGIA	FECHA		RESPONSABLE	PRESUPUESTO	RESULTADOS ESPERADOS
		INICIO	FINAL			
Modelo del dominio	Aplicar los análisis realizados Metodología OO mediante el uso de UML	26-10-2007	06-11-2007	Oscar Guamán Lady Rosillo	$ 20	Documento del modelo del domino
Definición de casos de uso esenciales	Utilización de los requerimientos	07-11-2007	13-11-2007	Oscar Guamán Lady Rosillo		Diseño de los casos de uso esenciales
Definición de casos de uso expandidos	Casos de uso esenciales	14-11-2007	27-11-2007	Oscar Guamán Lady Rosillo		Diseño de los casos de uso expandidos
Prototipo de pantallas	Casos de uso expandidos	28-11-2007	04-12-2007	Oscar Guamán		Prototipo de pantallas

				Lady Rosillo		
Diagramas del modelado	Fase de diseño	05-12-2007	01-01-2008	Oscar Guamán Lady Rosillo		Diagramas del modelo del dominio
Diseño de la base de datos	Diagramas de entidad-relación	02-01-2008	03-01-2008	Oscar Guamán Lady Rosillo		Diagrama de la base de datos

OBJETIVO ESPECIFICO: Diseñar los componentes básicos que ofrecerá el framework.

ACTIVIDADES O TAREA	**METODOLOGIA**	**FECHA**		**RESPONSABLE**	**PRESUPUESTO**	**RESULTADOS ESPERADOS**
		INICIO	**FINAL**			
Construcción del framework	Utilización de la fase de diseño	04-01-2008	31-07-2008	Oscar Guamán Lady Rosillo	$ 20	El framework que permite crear aplicaciones que contengan los componentes básicos

OBJETIVO ESPECIFICO: Permitir que las aplicaciones Web basadas en el Framework tengan el acceso a la base de Datos MySql						
ACTIVIDADES O TAREA	**METODOLOGIA**	**FECHA**		**RESPONSABLE**	**PRESUPUESTO**	**RESULTADOS ESPERADOS**
		INICIO	**FINAL**			
Investigación bibliográfica de cómo conectarse a la base de datos MySql	Se consultara en bibliotecas, Internet, manuales, libros de la bases de datos	01-08-2008	12-08-2008	Oscar Guamán Lady Rosillo	$ 100	Información para poderse conectar a la base de datos desde la aplicación construida.
Permitir que el framework pueda tener el acceso a la base de datos MySql	Análisis de la información obtenida	13-08-2008	11-09-2008	Oscar Guamán Lady Rosillo	$ 20	Framework con acceso a la base de datos MySql.

OBJETIVO ESPECIFICO: Las aplicaciones se almacenaran en un archivo XML.						
ACTIVIDADES O TAREA	**METODOLOGIA**	**FECHA**		**RESPONSABLE**	**PRESUPUESTO**	**RESULTADOS ESPERADOS**
		INICIO	**FINAL**			
Permitir que el Framework guarde las aplicaciones en un archivo XML	Análisis de la información obtenida a cerca de archivos XML	12-09-2008	23-10-2008	Oscar Guamán Lady Rosillo	$ 100	Framework que guarda las aplicaciones en archivos XML

OBJETIVO ESPECIFICO: Implementación del Framework						
ACTIVIDADES O TAREA	**METODOLOGIA**	**FECHA**		**RESPONSABLE**	**PRESUPUESTO**	**RESULTADOS ESPERADOS**
		INICIO	**FINAL**			
Pruebas e implementación.	Utilización del Framework.	24-10-2008	30-10-2008	Oscar Guamán Lady Rosillo	$ 100	Framework implementado

Documentación del proyecto	Utilización del framework ya construidos	31-10-2008	06-11-2008	Oscar Guamán Lady Rosillo		Manuales del proyecto

OBJETIVO ESPECÍFICO: Crear una aplicación de Ejemplo basado en el framework desarrollado.						
ACTIVIDADES O TAREA	**METODOLOGIA**	**FECHA**		**RESPONSABLE**	**PRESUPUESTO**	**RESULTADOS ESPERADO**
		INICIO	**FINAL**			
Crear la aplicación de ejemplo.	Utilización del framework.	07-11-2008	20-11-2008	Oscar Guamán Lady Rosillo	$ 100	La funcionalidad del Framework desarrollado.

ANEXO 2

UNIVERSIDAD NACIONAL DE LOJA

CARRERA DE INGENIERIA EN SISTEMAS

CUESTIONARIO DIRIGIDO A LOS ESTUDIANTES QUE UTILIZAN EL FRAMEWORK TOOLWEB

Lea detenidamente y conteste cada pregunta en forma clara y con la mayor objetividad posible. Por favor marque con una “x” la pregunta según su criterio y experiencia.

1. ¿La interfaz de las pantallas es amigable para el usuario?
   SI _____ NO _____

2. ¿Cómo evalúa la calidad de procesos realizado en el sistema Toolweb?
   a) Deficiente ( )
   b) Aceptable ( )
   c) Satisfactorio ( )
   d) Excelente ( )

   PORQUE:
   _____

3. ¿Considera que la funcionalidad del Framework Toolweb es efectiva?
   SI _____ NO _____
   PORQUE:
   _____

4. ¿Considera que el Framework Toolweb realiza los procesos de manera eficiente?
   SI _____ NO _____
   PORQUE:
   _____

5. ¿Cómo evalúa la usabilidad del sistema Toolweb?
    a) Deficiente ( )
    b) Aceptable ( )
    c) Satisfactorio ( )
    d) Excelente ( )

6. ¿Es amigable el Framework Toolweb para los desarrolladores?
   SI _____ NO _____
   PORQUE:
   _____

7. ¿Es sencillo de entender y manejar el Framework Toolweb para los usuarios a los cuales está destinado su uso?
   SI _____ NO _____

   PORQUE:
   _____

   Sugerencias:
   _____
   _____
   _____

GRACIAS

ANEXO 3

RESULTADOS DE ENCUESTAS.

1. ¿La interfaz de las pantallas es amigable para el usuario?

Si	90%
No	10%
Total:	**100%**

Análisis.

La mayoría de los alumnos que asistieron a la capacitación del Framework TOOLWEB, según su criterio la interfaz grafica del mismo, facilita su uso como herramienta de desarrollo.

2. ¿Cómo evalúa la calidad de procesos realizado en el sistema TOOLWEB?

Deficiente	0%
Aceptable	73.3%
Satisfactorio	26.7%
Excelente	0%
Total:	**100%**

Análisis.

De acuerdo a los encuestados la herramienta es aceptable debido a que se puede realizar una aplicación web funcional. Y la otra parte manifiesta que los procesos que se realizan con esta herramienta son de manera satisfactoria para el desarrollo de las aplicaciones.

3. ¿Considera que la funcionalidad del Framework TOOLWEB es efectiva?

Si	73,7%
No	26,3%
Total:	**100%**

Análisis:

Las personas que utilizaron la herramienta de desarrollo consideran que esta desarrollada para cumplir las funciones establecida, como son la creación de las aplicaciones web con acceso a las bases de datos.

4. **¿Considera que el Framework TOOLWEB realiza los procesos de manera eficiente?**

Si	70%
No	30%
Total:	**100%**

Análisis.

El 70% de los encuestados consideran que los procesos que se realizan con la herramienta se realizan de una manera eficiente y rápida, en cuanto al 30 % sugiere que se aumente algunos controles para un mejor manejo.

5. **¿Cómo evalúa la usabilidad del sistema TOOLWEB?**

Deficiente	0%
Aceptable	70%
Satisfactorio	30%
Excelente	0%
Total:	**100%**

Análisis.

En cuanto a la usabilidad de la herramienta la mayoría tiene una aceptación considerable para la creación de las aplicaciones web que cuentan con la tecnología AJAX.

6. **¿Es amigable el Framework TOOLWEB para los desarrolladores?**

Si	73,7%
No	26,3%
Total:	**100%**

Análisis.

Según el resultado de esta pregunta la mayoría de alumnos que recibieron la capacitación nos dieron a conocer q es una herramienta que les va a permitir desarrollar sus aplicaciones web.

7. **¿Es sencillo de entender y manejar el Framework TOOLWEB para los usuarios a los cuales está destinado su uso?**

Si	99%
No	1%
Total:	**100%**

Análisis.

Muchos de los estudiantes que se les dio la capacitación pudieron entender y manejar el Framework de una manera fácil y sencilla, ya que la herramienta tiene los procesos muy entendibles.

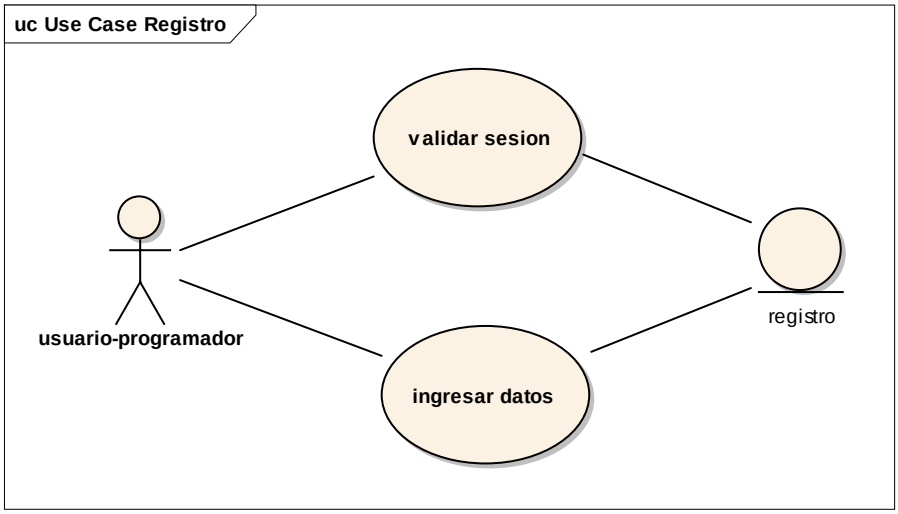
ANEXO 4

APLICACIÓN WEB “REGISTRO”

La aplicación desarrollada con la herramienta TOOLWEB la denominamos Registro la cual permite acceder a una sesión, al usuario con su nombre y contraseña, estos datos son validados en la base de datos Registro en la tabla ingreso, y en caso de no estar registrado el Usuario debe registrarse, es decir ingresar sus datos como nombres, apellido, dirección, usuario, contraseña a la base de datos.

MODELAMIENTO DE CASOS DE USO DE LA APLICACIÓN WEB “REGISTRO”

1. Diagramas de Casos de Uso



2. Descripción de Casos de Uso

Validar Sesión

Fig. 1: Icono TOOLWEB

Fig. 2: Ventana TOOLWEB menú Archivo opción Nueva Aplicación

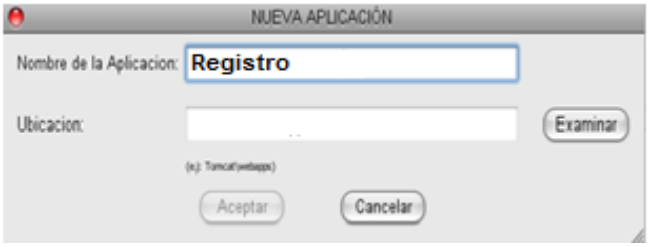


Fig. 3: Ventana Inicial Nueva Aplicación

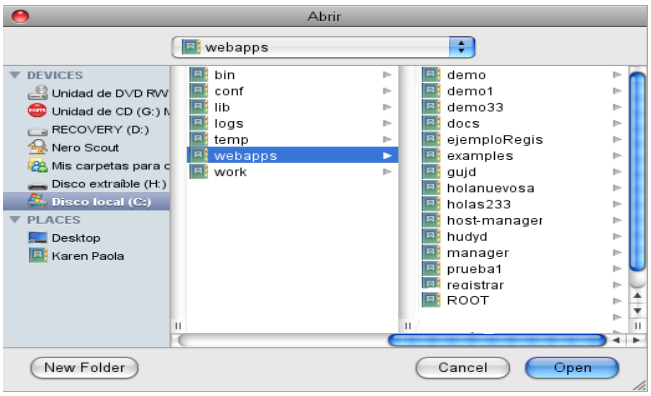


Fig. 4: Ventana Abrir Nueva Aplicación

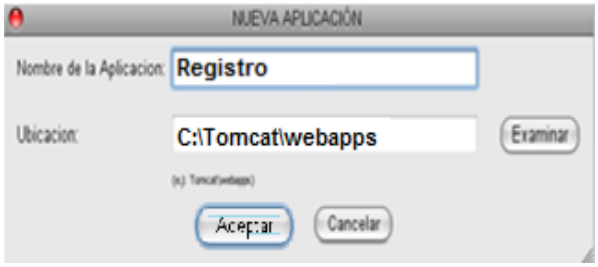


Fig. 5: Ventana Nueva Aplicación

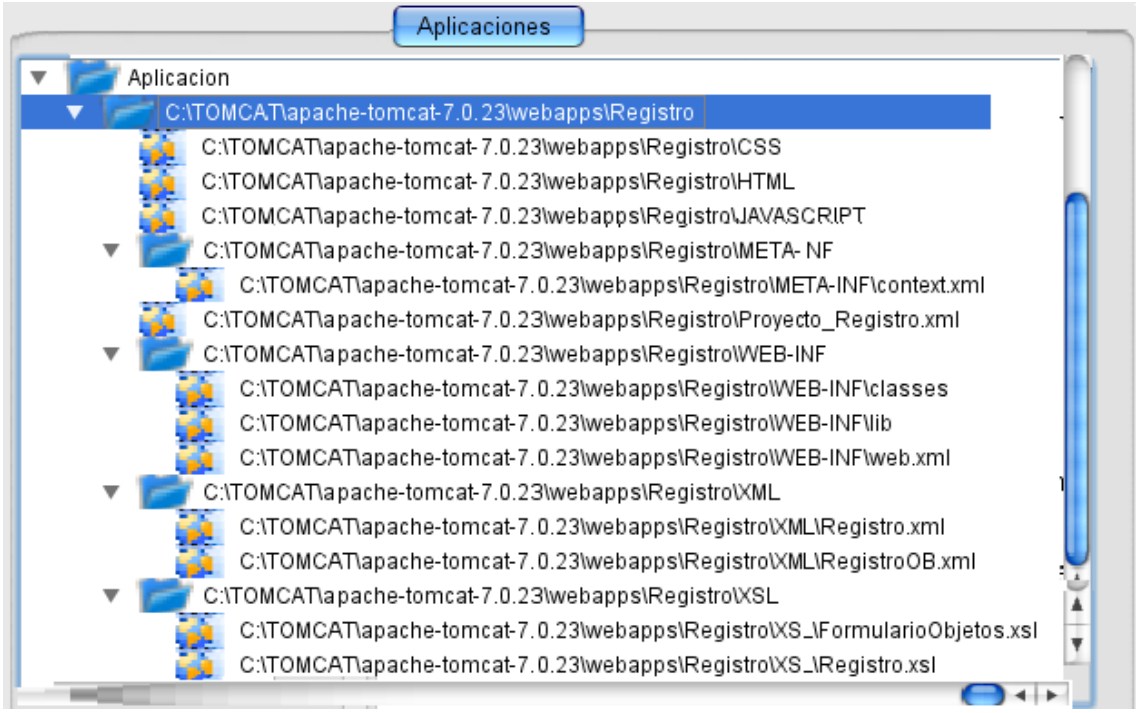


Fig. 6: Panel Aplicación

Fig. 7: Panel Código xml archivo inicial

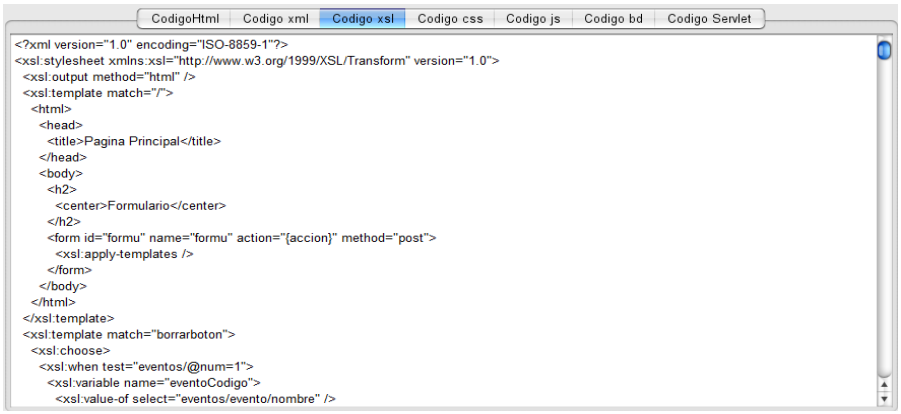


Fig. 8: Panel Código xsl archivo inicial

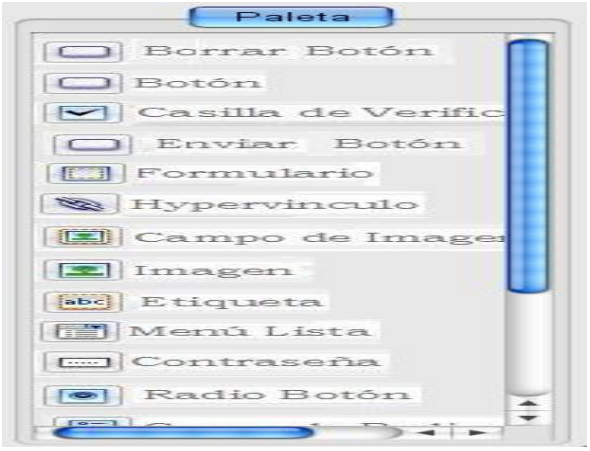


Fig. 8: Panel Paleta

Fig. 9: Ventana Etiqueta (REGISTRO)

Fig. 10: Ventana Etiqueta (USUARIO:)

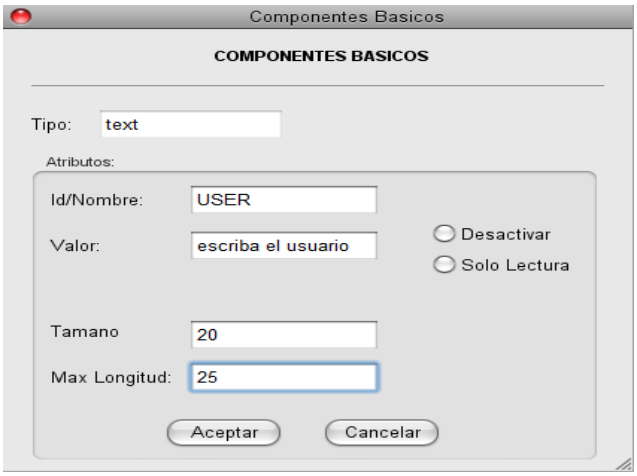


Fig. 11: Ventana Componentes Básicos (Campo de Texto) usuario

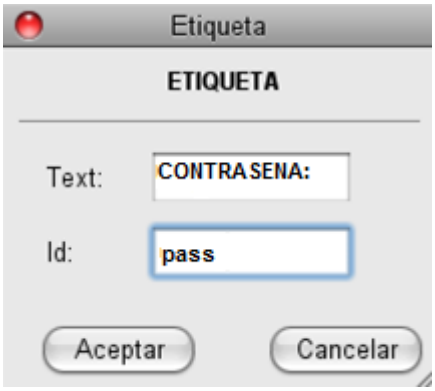


Fig. 12: Ventana Etiqueta (CONTRASEÑA:)

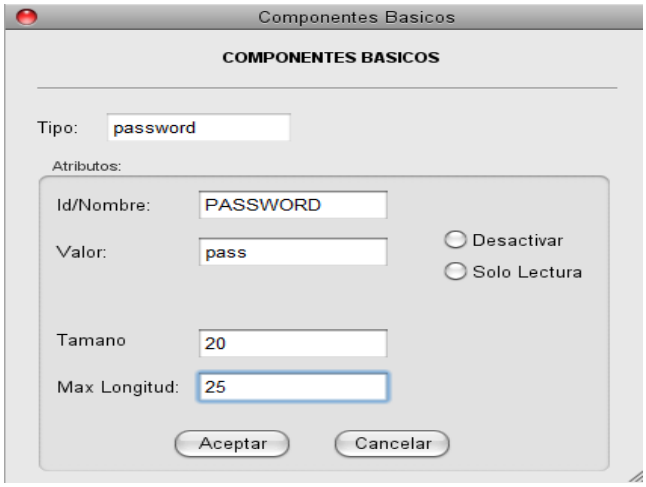


Fig. 13: Ventana Componentes Básicos (Campo de Texto) password

Fig. 14: Ventana Componentes Básicos (Enviar Botón) Enviar

Fig. 15: Ventana Componentes Básicos (Borrar Botón) Limpiar

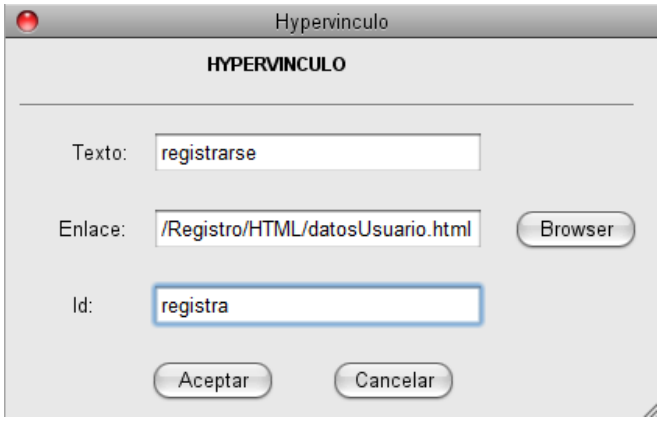


Fig. 16: Ventana Componentes Hypervinculo (registrarse)

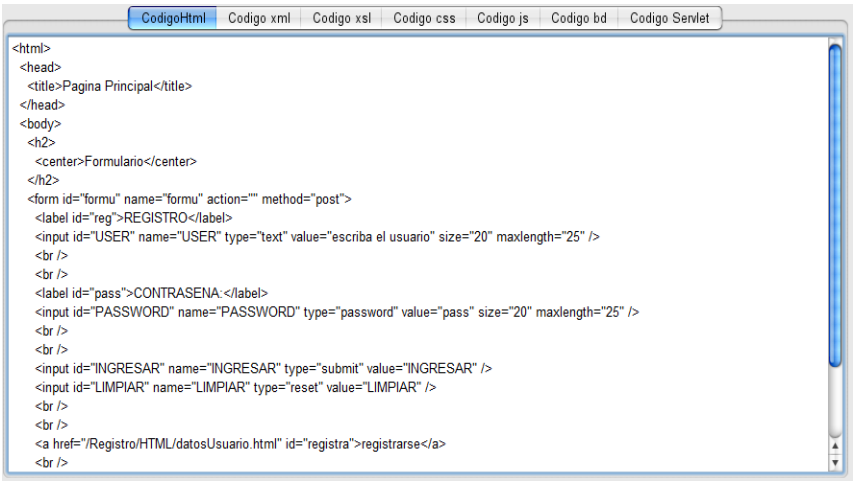


Fig. 16: Panel CódigoHtml (Index.html)

Fig. 17: Panel Vista Previa (Index.html)

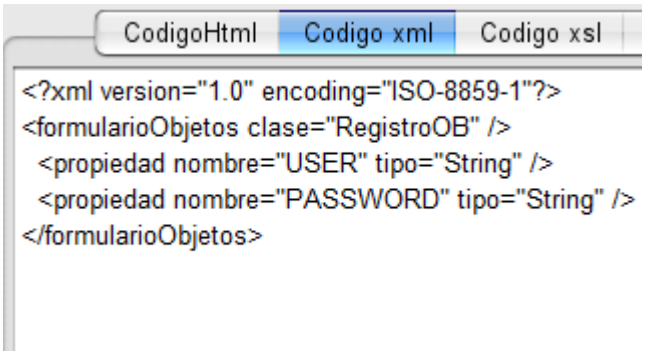


Fig. 18: Panel Código xml (RegistroOB.xml)

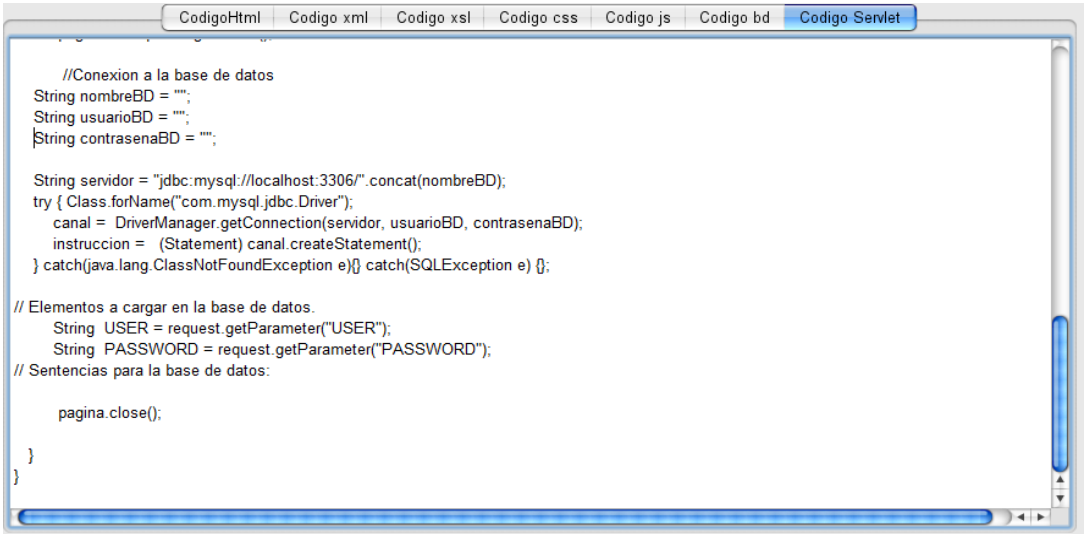


Fig. 19: Panel Código xml (RegistroOB.java)

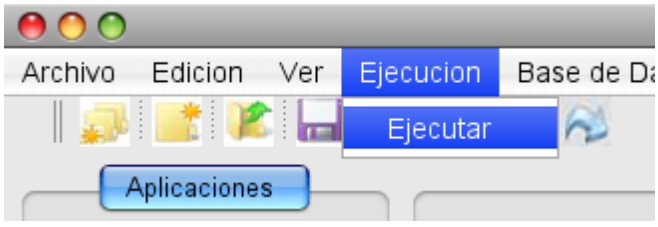


Fig. 20: Ventana TOOLWEB menú Ejecución opción Ejecutar

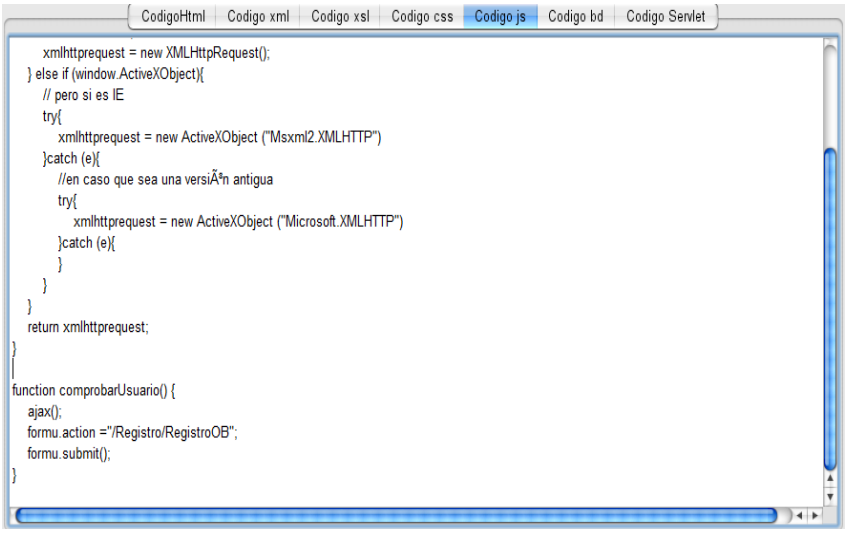


Fig. 21: Panel Código js (Registro.js)

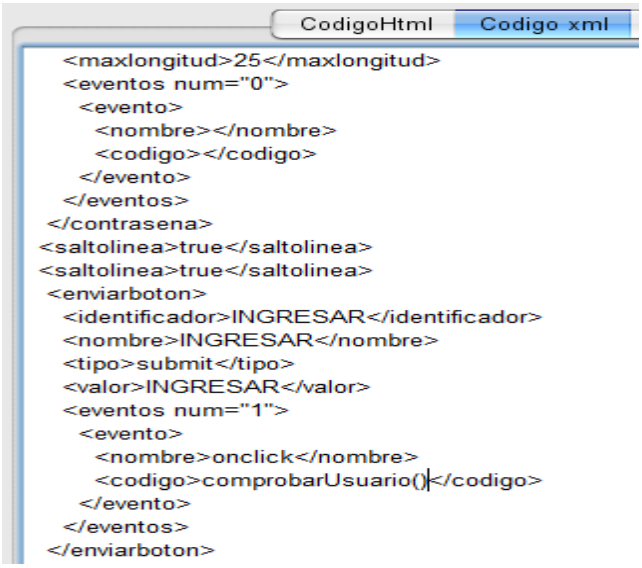


Fig. 22: Panel Código xml (Registro.xml)

Fig. 23: Panel Código css (Registro.css)

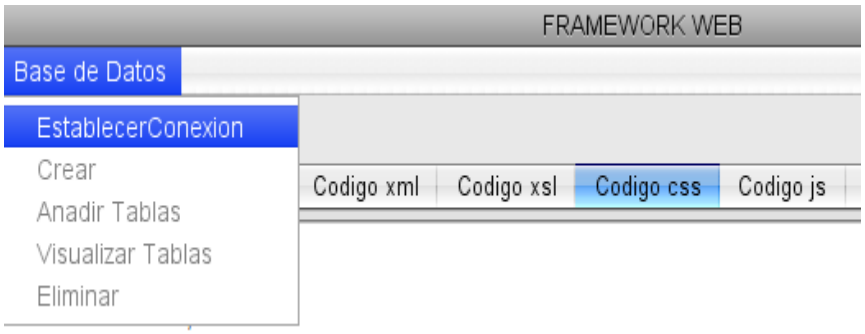


Fig. 24: Ventana TOOLWEB menú Base de Datos opción Establecer Conexión

Fig. 25: Ventana Establecer Conexión

Fig. 26: Ventana Mensaje de Aviso (Conexión Establecida)

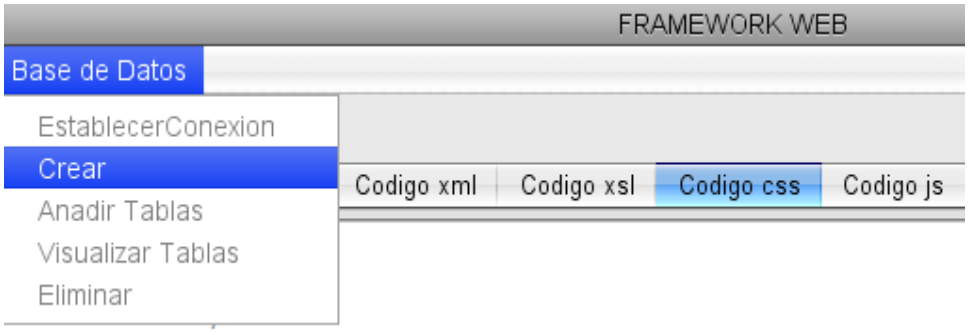


Fig. 27: Ventana TOOLWEB menú Base de Datos opción Crear

Fig. 28: Ventana Crear Base de Datos (registro)

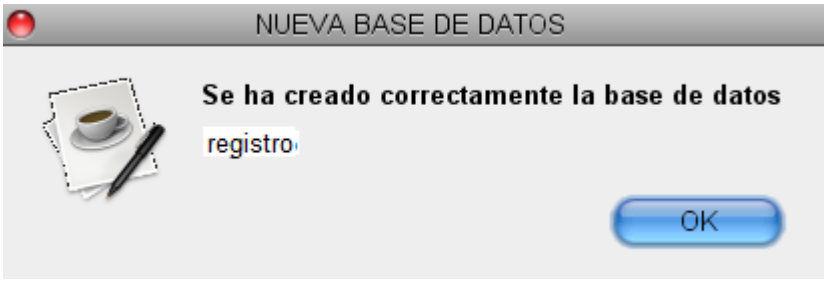


Fig. 29: Ventana Mensaje de Aviso (Nueva Base de Datos)

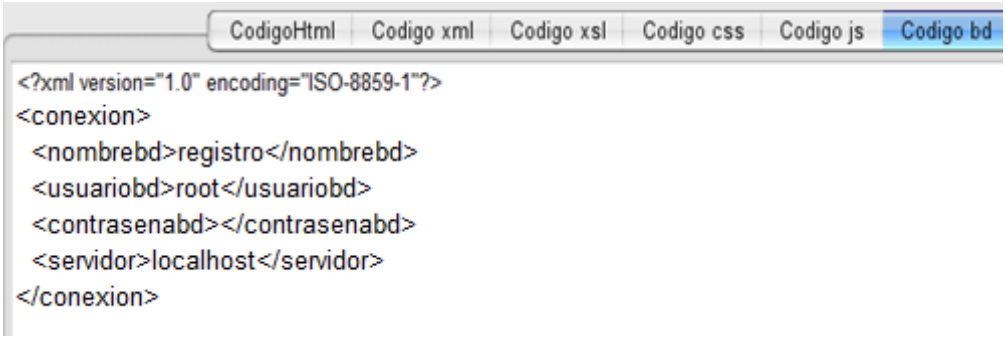


Fig. 30: Panel Código bd (conexionbd_registro.xml)

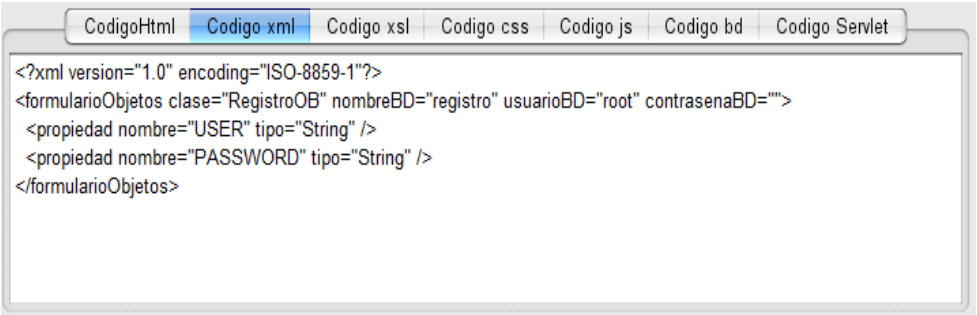


Fig. 31: Panel Código xml (RegistroOB.xml)

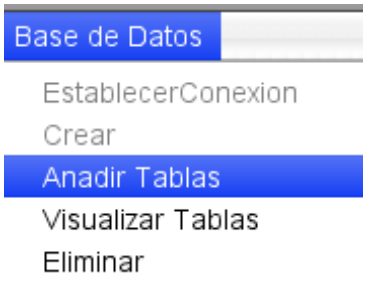


Fig. 32: Ventana TOOLWEB menú Base de Datos opción Añadir Tablas

Fig. 33: Ventana Añadir Tablas

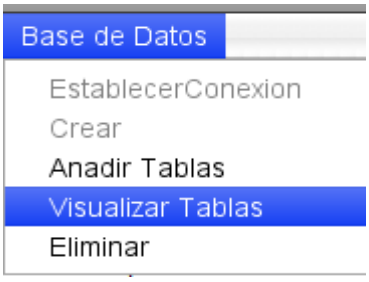


Fig. 34: Ventana TOOLWEB menú Base de Datos opción Visualizar Tablas

Fig. 34: Ventana Visualizar Base de Datos y Tablas

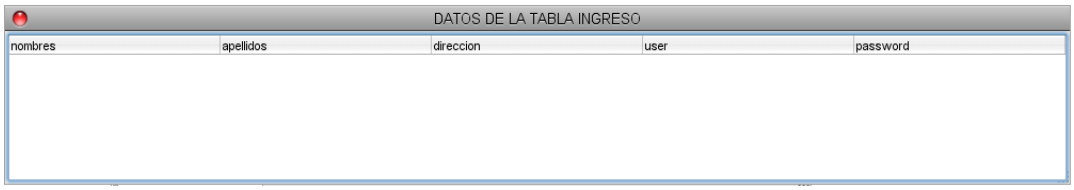


Fig. 35: Ventana Datos de la Tabla ingreso.

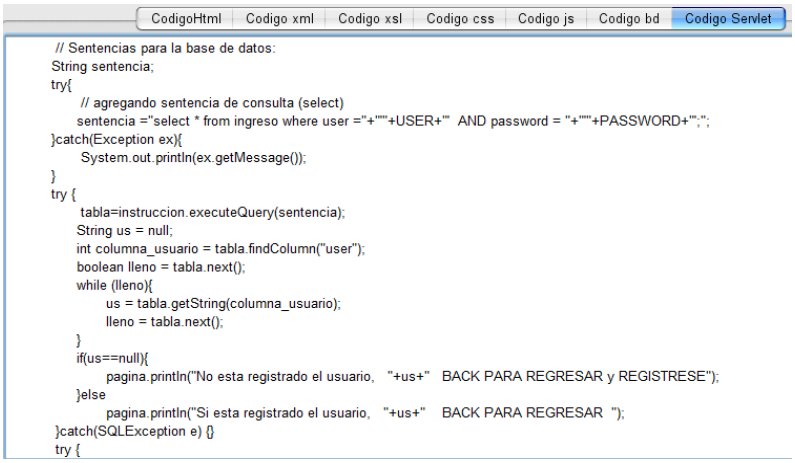


Fig. 36: Panel Código Servlet (RegistroOB.java)

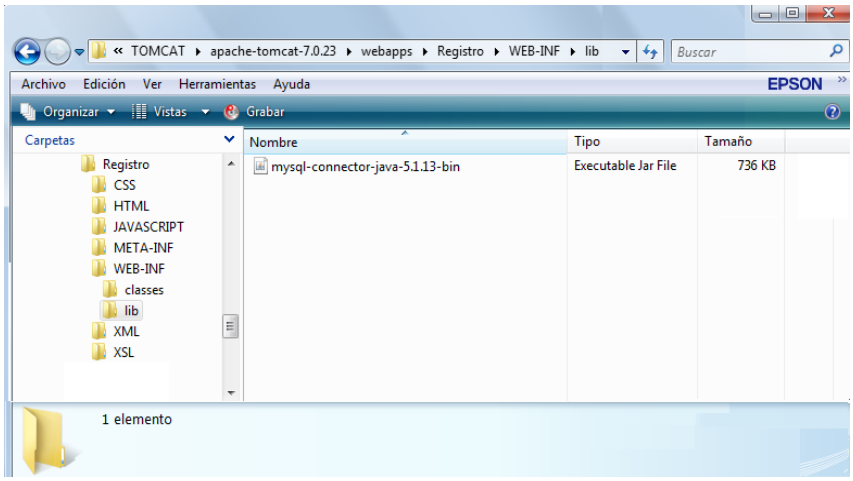


Fig. 37: Directorio /TOMCAT7.0/webapps/Registro /WEB-INF/lib

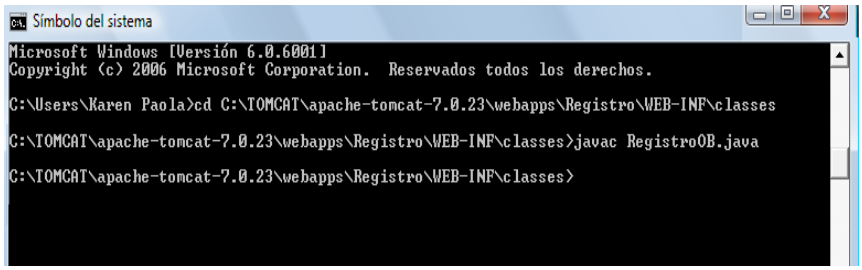


Fig. 38: Ventana Símbolo del Sistema (compilar archivo RegistroOB.java)

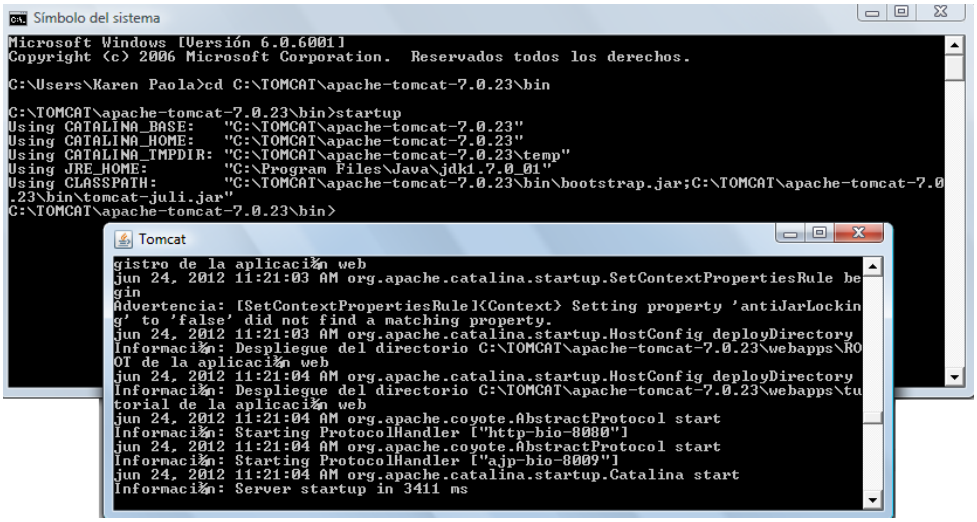


Fig. 39: Ventana Símbolo del Sistema (iniciar Tomcat)

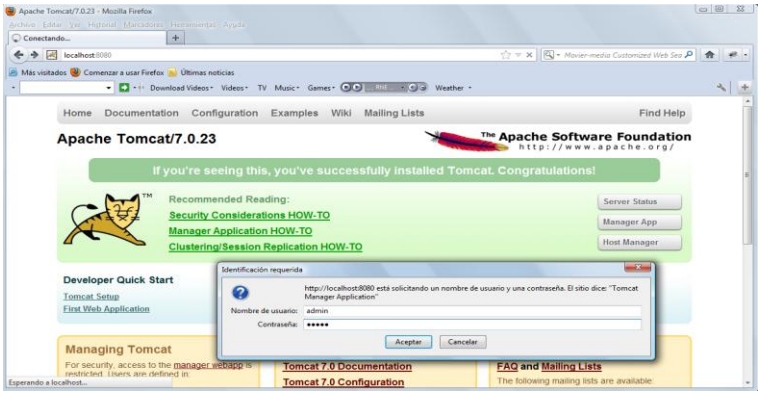


Fig. 40: Ventana de Inicio del Apache Tomcat

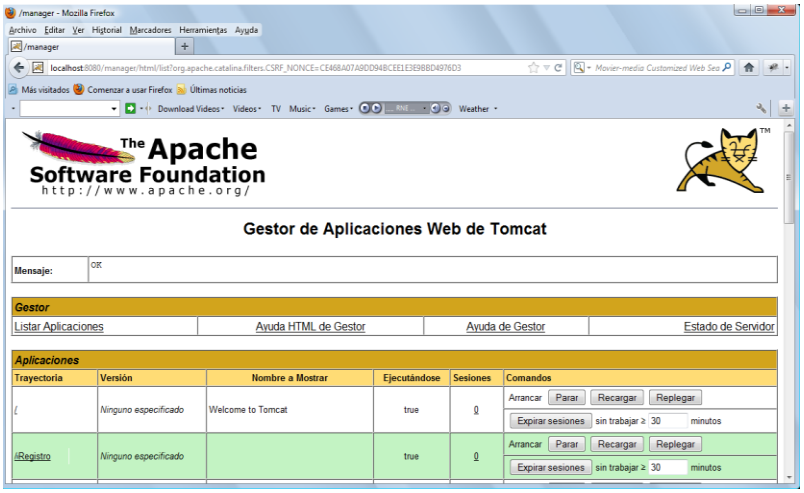


Fig. 41: Ventana Gestor de Aplicaciones Web de Tomcat

Fig. 42: Ventana Página de Inicio Registro (index.html)

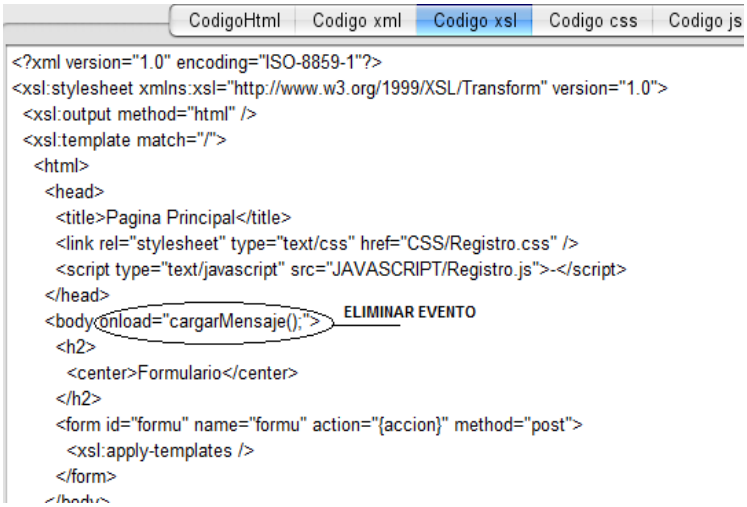


Fig. 43: Panel Código xsl (Registro.xsl)

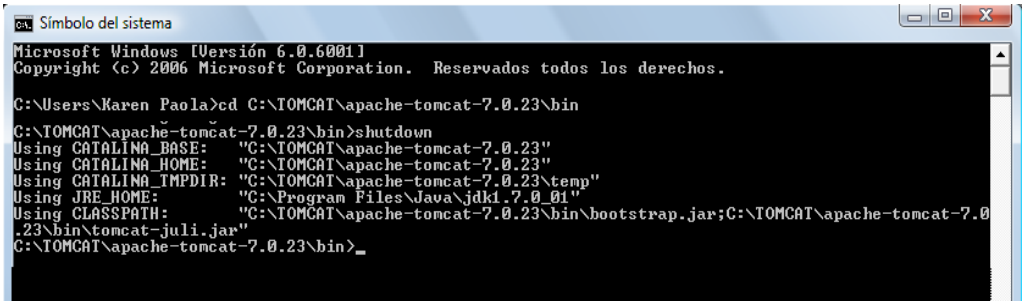


Fig. 44: Ventana Símbolo del Sistema (parar Tomcat)

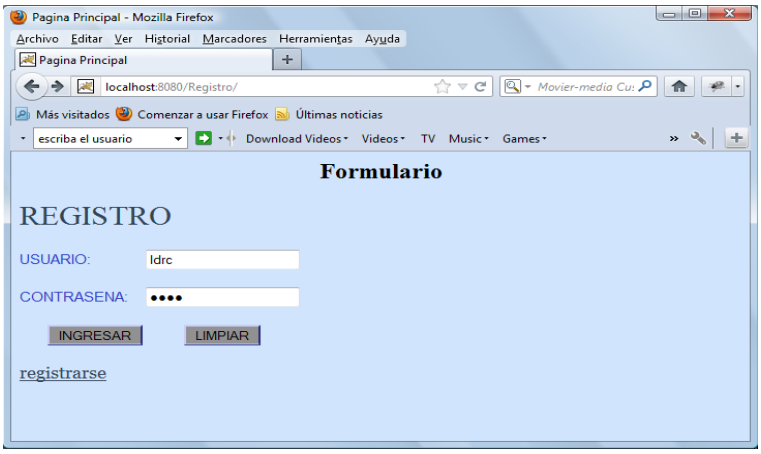


Fig. 45: Ventana Página del formulario (index.html)

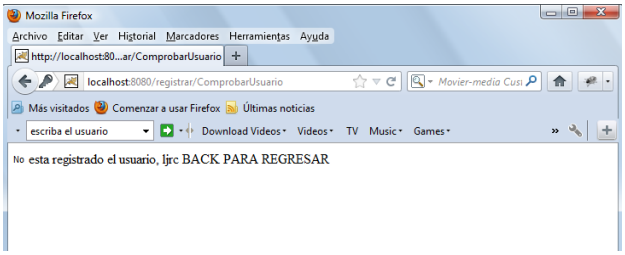


Fig. 46: Ventana de Resultado de la Validación de Sesión

Nombre:	Validar Sesión
Actores:	Usuario
Propósito:	Permitir al usuario iniciar sesión con su nombre de Usuario y contraseña
Visión General:	El usuario accede a una sesión.
Tipo:	Primario, Esencial
Curso Típico de Eventos	
1.	El usuario-programador ejecuta el icono de acceso directo, para iniciar la aplicación.
2.	El Framework TOOLWEB presenta la ventana inicial.
3.	El usuario-programador selecciona la opción crear Nueva Aplicación dentro del menú Archivo de la ventana TOOLWEB.
4.	El Framework TOOLWEB presenta la ventana inicial Nueva Aplicación.
5.	El usuario-programador ingresa el nombre de la aplicación que se creará denominada Registro, en la ventana Nueva Aplicación.
6.	Presionamos el botón Examinar, en la ventana Abrir seleccionamos el directorio del webapps del Tomcat 7.
7.	Una vez ingresados los datos presionamos el botón Aceptar, para confirmar la creación de la aplicación web.
8.	El Framework TOOLWEB valida los datos ingresados del nombre y ubicación.
9.	El Framework TOOLWEB crea la aplicación Registro la cual se visualiza en el panel Aplicación con los directorios que contiene y los archivos.
10.	El Framework TOOLWEB muestra los archivos iniciales de xml y xsl en el panel Código xml y panel Código xsl.
11.	El usuario-programador procede a crear los componentes para hacer el formulario.
12.	El usuario-programador selecciona del panel Paleta el componente Etiqueta,

para crear la etiqueta REGISTRO.

13. El Framework TOOLWEB presenta la ventana Etiqueta.
14. El usuario-programador ingresa los datos del componente: Text y Id, luego presiona la opción Aceptar.
15. El Framework TOOLWEB valida los datos y crea el componente Etiqueta, adicionando la información al archivo xml.
16. Seleccionamos del panel Paleta el componente Salto de Línea.
17. El Framework TOOLWEB crea el componente Salto de Línea, adicionando la información al archivo xml.
18. El usuario-programador selecciona del panel Paleta el componente Etiqueta, para crear la etiqueta USUARIO.
19. El Framework TOOLWEB presenta la ventana Etiqueta.
20. El usuario-programador ingresa los datos del componente: Text y Id, luego presiona la opción Aceptar.
21. El Framework TOOLWEB valida los datos y crea el componente Etiqueta, adicionando la información al archivo xml.
22. El usuario-programador selecciona del panel Paleta el componente Campo de Texto, para crear el campo usuario.
23. El Framework TOOLWEB presenta la ventana Componentes Básicos (Campo de Texto).
24. El usuario-programador ingresa los datos del componente: Id/Nombre, Valor, Tamaño, Max. Longitud, luego presiona la opción Aceptar.
25. El Framework TOOLWEB valida los datos y crea el componente Campo de Texto, adicionando la información al archivo xml.
26. El usuario-programador selecciona del panel Paleta el componente Etiqueta, para crear la etiqueta CONTRASEÑA.
27. El Framework TOOLWEB presenta la ventana Etiqueta.
28. El usuario-programador ingresa los datos del componente: Text y Id, luego presiona la opción Aceptar.
29. El Framework TOOLWEB valida los datos y crea el componente Etiqueta, adicionando la información al archivo xml.
30. El usuario-programador selecciona del panel Paleta el componente Campo de Texto, para crear el campo contraseña.
31. El Framework TOOLWEB presenta la ventana Componentes Básicos (Campo

de Texto).

32. El usuario-programador ingresa los datos del componente: Id/Nombre, Valor, Tamaño, Max. Longitud, luego presiona la opción Aceptar.
33. El Framework TOOLWEB valida los datos y crea el componente Campo de Texto, adicionando la información al archivo xml.
34. El usuario-programador selecciona del panel Paleta el componente Enviar Botón, para crear el botón Enviar.
35. El Framework TOOLWEB presenta la ventana Componentes Básicos (Enviar Botón).
36. El usuario-programador ingresa los datos del componente: Id/Nombre, Valor, luego presiona la opción Aceptar.
37. El Framework TOOLWEB valida los datos y crea el componente Enviar Botón, adicionando la información al archivo xml.
38. El usuario-programador selecciona del panel Paleta el componente Borrar Botón, para crear el botón Limpiar.
39. El Framework TOOLWEB presenta la ventana Componentes Básicos (Borrar Botón).
40. El usuario-programador ingresa los datos del componente: Id/Nombre, Valor, luego presiona la opción Aceptar.
41. El Framework TOOLWEB valida los datos y crea el componente Borrar Botón, adicionando la información al archivo xml.
42. El usuario-programador selecciona del panel Paleta el componente Hypervinculo, para crear el botón registrarse.
43. El Framework TOOLWEB presenta la ventana Componentes Hypervinculo.
44. El usuario-programador ingresa los datos del componente: Texto, Enlace, Id, luego presiona la opción Aceptar.
45. El Framework TOOLWEB valida los datos y crea el componente Hypervinculo, adicionando la información al archivo xml.
46. El usuario-programador seleccionar la opción Vista Previa.
47. El Framework TOOLWEB genera el archivo index.html con los componentes en código fuente y componente gráficos, los mismos que se visualizan en el panel CodigoHtml y panel Vista Previa.
48. El Framework TOOLWEB adiciona los parámetros de los componentes de entrada/salida Campo de Texto y Contraseña en el archivo RegistroOB.xml y

el archivo RegistroOB.java con los parámetros del formulario.

49. El Framework TOOLWEB visualiza en el panel Código Servlet el archivo RegistroOB.java.
50. El usuario-programador selecciona la opción Ejecutar, del menú Ejecución de la ventana TOOLWEB.
51. El Framework TOOLWEB crea los archivos javascript y Css
52. El usuario-programador procede a editar en el panel Código Js el archivo JavaScript: Registro.js, adicionando una función denominada comprobar Usuario.
53. El usuario-programador procede a editar en el panel Código xml el archivo Registro.xml, adicionando el evento onclick al botón Enviar, el mismo que hace referencia a la función comprobarUsuario() del archivo Registro.js.
54. El usuario-programador seleccionar la opción Vista Previa, para actualizar los cambios que se realizaron del evento.
55. El usuario-programador edita el archivo Registro.css utilizando los identificadores de los componentes del formulario, con la finalidad de dar estilo a la página index.html.
56. El usuario-programador seleccionar la opción Establecer Conexión en el menú Base de Datos de la ventana TOOLWEB.
57. El Framework TOOLWEB presenta la ventana Establecer Conexión.
58. El usuario-programador ingresa los datos para la Conexión: usuario y contraseña del motor de Base de Datos MySql en la ventana Establecer Conexión.
59. El usuario-programador selecciona la opción Aceptar para confirmar la conexión a la Base de Datos MySql.
60. El Framework TOOLWEB presenta un mensaje de Aviso, de la conexión establecida.
61. El usuario-programador elige la opción del menú Base de Datos Crear, de la ventana TOOLWEB
62. El Framework TOOLWEB presenta la ventana Crear Base de Datos.
63. El usuario-programador introduce el nombre de la Base de Datos “registro”, luego presiona el botón Aceptar.
64. El Framework TOOLWEB crea la Base de Datos y presenta un mensaje de Aviso, Nueva Base de Datos.

65. El Framework TOOLWEB crea el archivo conexionbd_registro.xml con la información de la base de datos, se visualiza en el Panel Código bd.
66. El Framework TOOLWEB actualiza el archivo RegistroOB.xml, adicionando los atributos de la Base de Datos: nombreBD, usuarioBD, contrasenaBD.
67. El usuario-programador en el menú Base de Datos, elige la opción Añadir Tablas.
68. El Framework TOOLWEB presenta la ventana Añadir Tablas.
69. El usuario-programador crea una tabla en la Base de Datos registro introduce el nombre: ingreso, luego presiona el botón Añadir Campos.
70. El Framework TOOLWEB presenta el panel Campos.
71. El usuario-programador adiciona los campos: nombres, apellidos, dirección, user y password; ingresando el nombre, seleccionar el tipo, ingresar la longitud, luego se presiona el botón añadir. Una vez que se ingreso a todos los campos se presiona el botón Aceptar.
72. El Framework TOOLWEB ha creado la tabla ingreso con sus respectivos campos en la Base de Datos registro.
73. El usuario-programador desea visualizar la tabla con sus campos, se selecciona la opción Visualizar Tablas del menú Base de Datos, de la ventana TOOLWEB.
74. El Framework TOOLWEB presenta la ventana Visualizar Base de Datos y Tablas.
75. El usuario-programador selecciona la Base de Datos registro y la tabla ingreso, luego se presiona el botón Visualizar Datos para ver la tabla.
76. El Framework TOOLWEB presenta la ventana Datos de Tabla ingreso.
77. El usuario-programador edita en el Panel Código Servlet la clase RegistroOB.java adicionando la sentencia de consulta para validar al Usuario en la Base de Datos.
78. El usuario-programador copia la librería del conector de MySql en el directorio de /TOMCAT7.0/webapps/Registro /WEB-INF/lib.
79. El usuario-programador abre el símbolo de Sistema para compilar la clase RegistroOB.java.
80. El usuario-programador abre el navegador para visualizar la aplicación en el servidor Tomcat 7, previamente levantado el servidor, para esto abrimos el Símbolo del Sistema y agregamos el comando startup, en el directorio

/Tomcat/bin.

81. El usuario-programador ingresa en la página de inicio de Apache Tomcat en la opción Manager App, introduce los datos del Nombre de Usuario (admin) y Contraseña (admin).
82. El navegador presenta la Ventana Gestor de Aplicaciones Web de Tomcat
83. El usuario-programador selecciona en Aplicaciones Trayectoria /Registro.
84. El navegador presenta la ventana del archivo index.html, con el mensaje de inicio.
85. El Framework TOOLWEB en todas las aplicación web tiene su página de inicio, para visualizar la funcionalidad de la página index.html, se debe eliminar el evento de la etiqueta body que llama a la función cargarMsj() del archivo Registro.xsl.
86. El usuario-programador para actualizar los cambios debe elegir la opción vista Previa en la herramienta TOOLWEB, y parar el servidor Tomcat 7, en la ventana Símbolo del Sistema con el comando shutdown.
87. El usuario-programador procede a levantar nuevamente el servidor Tomcat 7 (startup), y se elige la aplicación web Registro.
88. El navegador carga la página del formulario index.html.
89. El usuario-programador procede a realizar la validación de sesión, ingresando el usuario y contraseña, luego se presiona el botón Ingresar.
90. El Framework hace la conexión de la página index.html con el archivo Registro.js el mismo que llama a la clase RegistroOB.java la cual hace la sentencia de consulta en la Base de Dato registro, dando de resultado la validación del usuario.
91. El Framework presenta la ventana de Resultado de la Validación de Sesión, verificando si el usuario está o no registrado.

Ingresar Datos

Fig. 47: Ventana TOOLWEB menú archivo opción Nuevo Archivo

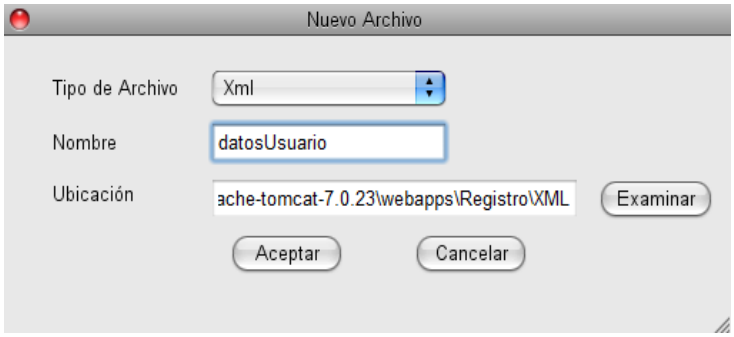


Fig. 48: Ventana Nuevo Archivo

Fig. 49: Ventana Componente Etiqueta

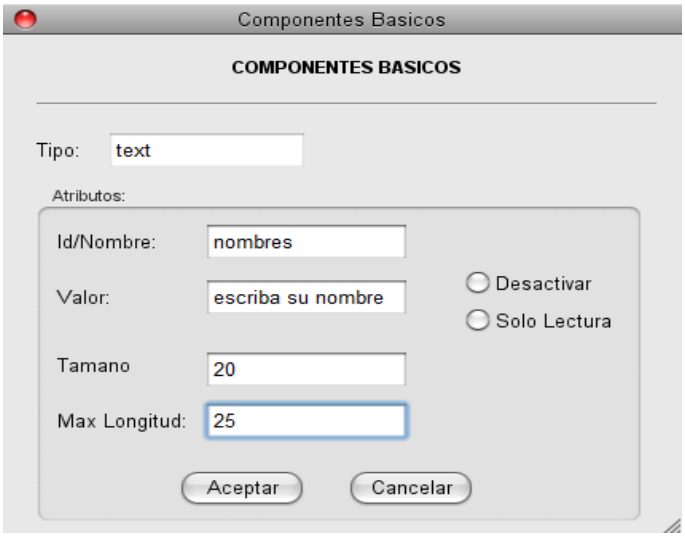


Fig. 50: Ventana Componentes Básicos (Campo de Texto)

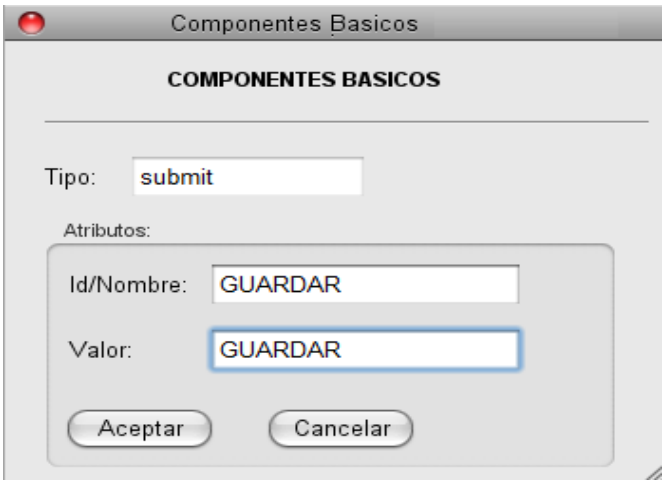


Fig. 51: Ventana Componentes Básicos (Enviar Botón)

Fig. 52: Ventana Componentes Básicos (Borrar Botón)

Fig. 48: Ventana del Formulario (datosUsuario.html)

Nombre:	Ingresar Datos
Actores:	Usuario
Propósito:	Permitir al usuario registrarse a la sesión.
Visión General:	El usuario accede a una sesión.
Tipo:	Primario, Esencial
Curso Típico de Eventos	
1. El usuario-programador selecciona la opción Nuevo Archivo del menú Archivo de la ventana TOOLWEB. 2. El Framework TOOLWEB presenta la ventana Nuevo Archivo. 3. El usuario-programador selecciona el tipo de archivo: xml, ingresa el nombre del archivo: datosUsuario. 4. El Framework TOOLWEB ubica el archivo en el directorio respectivo a su tipo: /Registro/XML. 5. El usuario-programador presiona el botón Aceptar en la ventana Nuevo Archivo. 6. El Framework TOOLWEB valida los datos tipo, nombre y ubicación. 7. El Framework TOOLWEB crea el archivo datosUsuario.xml que se visualiza en el panel Código xml. 8. El Framework TOOLWEB crea el archivo datosUsuarioOB.xml donde se	

almacenara la propiedad de cada componente de entrada/salida.

9. El usuario-programador procede a crear los componentes para hacer el formulario, selecciona del panel Paleta el componente Etiqueta, para crear la etiqueta DATOS PERSONALES.
10. El Framework TOOLWEB presenta la ventana Etiqueta.
11. El usuario-programador ingresa los datos del componente: Text y Id, luego presiona la opción Aceptar.
12. El Framework TOOLWEB valida los datos y crea el componente Etiqueta, adicionando la información al archivo xml.
13. Seleccionamos del panel Paleta el componente Salto de Línea.
14. El Framework TOOLWEB crea el componente Salto de Línea, adicionando la información al archivo xml.
15. El usuario-programador selecciona del panel Paleta el componente Etiqueta, para crear la etiqueta NOMBRES.
16. El Framework TOOLWEB presenta la ventana Etiqueta.
17. El usuario-programador ingresa los datos del componente: Text y Id, luego presiona la opción Aceptar.
18. El Framework TOOLWEB valida los datos y crea el componente Etiqueta, adicionando la información al archivo xml.
19. El usuario-programador selecciona del panel Paleta el componente Campo de Texto, para crear el campo nombres.
20. El Framework TOOLWEB presenta la ventana Componentes Básicos (Campo de Texto).
21. El usuario-programador ingresa los datos del componente: Id/Nombre, Valor, Tamaño, Max. Longitud, luego presiona la opción Aceptar.
22. El Framework TOOLWEB valida los datos y crea el componente Campo de Texto, adicionando la información al archivo xml.
23. El usuario-programador selecciona del panel Paleta el componente Etiqueta, para crear la etiqueta APELLIDOS.
24. El Framework TOOLWEB presenta la ventana Etiqueta.
25. El usuario-programador ingresa los datos del componente: Text y Id, luego presiona la opción Aceptar.
26. El Framework TOOLWEB valida los datos y crea el componente Etiqueta, adicionando la información al archivo xml.

27. El usuario-programador selecciona del panel Paleta el componente Campo de Texto, para crear el campo apellidos.
28. El Framework TOOLWEB presenta la ventana Componentes Básicos (Campo de Texto).
29. El usuario-programador ingresa los datos del componente: Id/Nombre, Valor, Tamaño, Max. Longitud, luego presiona la opción Aceptar.
30. El Framework TOOLWEB valida los datos y crea el componente Campo de Texto, adicionando la información al archivo xml.
31. El usuario-programador selecciona del panel Paleta el componente Etiqueta, para crear la etiqueta DIRECCION.
32. El Framework TOOLWEB presenta la ventana Etiqueta.
33. El usuario-programador ingresa los datos del componente: Text y Id, luego presiona la opción Aceptar.
34. El Framework TOOLWEB valida los datos y crea el componente Etiqueta, adicionando la información al archivo xml.
35. El usuario-programador selecciona del panel Paleta el componente Campo de Texto, para crear el campo dirección.
36. El Framework TOOLWEB presenta la ventana Componentes Básicos (Campo de Texto).
37. El usuario-programador ingresa los datos del componente: Id/Nombre, Valor, Tamaño, Max. Longitud, luego presiona la opción Aceptar.
38. El Framework TOOLWEB valida los datos y crea el componente Campo de Texto, adicionando la información al archivo xml.
92. El usuario-programador selecciona del panel Paleta el componente Etiqueta, para crear la etiqueta USUARIO.
93. El Framework TOOLWEB presenta la ventana Etiqueta.
94. El usuario-programador ingresa los datos del componente: Text y Id, luego presiona la opción Aceptar.
95. El Framework TOOLWEB valida los datos y crea el componente Etiqueta, adicionando la información al archivo xml.
96. El usuario-programador selecciona del panel Paleta el componente Campo de Texto, para crear el campo usuario.
97. El Framework TOOLWEB presenta la ventana Componentes Básicos (Campo de Texto).

98. El usuario-programador ingresa los datos del componente: Id/Nombre, Valor, Tamaño, Max. Longitud, luego presiona la opción Aceptar.
99. El Framework TOOLWEB valida los datos y crea el componente Campo de Texto, adicionando la información al archivo xml.
39. El usuario-programador selecciona del panel Paleta el componente Etiqueta, para crear la etiqueta CONTRASENA.
40. El Framework TOOLWEB presenta la ventana Etiqueta.
41. El usuario-programador ingresa los datos del componente: Text y Id, luego presiona la opción Aceptar.
42. El Framework TOOLWEB valida los datos y crea el componente Etiqueta, adicionando la información al archivo xml.
43. El usuario-programador selecciona del panel Paleta el componente Campo de Texto, para crear el campo contraseña.
44. El Framework TOOLWEB presenta la ventana Componentes Básicos (Campo de Texto).
45. El usuario-programador ingresa los datos del componente: Id/Nombre, Valor, Tamaño, Max. Longitud, luego presiona la opción Aceptar.
46. El Framework TOOLWEB valida los datos y crea el componente Campo de Texto, adicionando la información al archivo xml.
47. El usuario-programador selecciona del panel Paleta el componente Enviar Botón, para crear el botón Guardar.
48. El Framework TOOLWEB presenta la ventana Componentes Básicos (Enviar Botón).
49. El usuario-programador ingresa los datos del componente: Id/Nombre, Valor, luego presiona la opción Aceptar.
50. El Framework TOOLWEB valida los datos y crea el componente Enviar Botón, adicionando la información al archivo xml.
51. El usuario-programador selecciona del panel Paleta el componente Borrar Botón, para crear el botón Limpiar.
52. El Framework TOOLWEB presenta la ventana Componentes Básicos (Borrar Botón).
53. El usuario-programador ingresa los datos del componente: Id/Nombre, Valor, luego presiona la opción Aceptar.
54. El Framework TOOLWEB valida los datos y crea el componente Borrar Botón,

adicionando la información al archivo xml.

55. El usuario-programador seleccionar la opción Vista Previa.
56. El Framework TOOLWEB genera el archivo datosUsuario.html con los componentes en código fuente y componente gráficos, los mismos que se visualizan en el panel CodigoHtml y panel Vista Previa.
57. El Framework TOOLWEB adiciona los parámetros de los componentes de entrada/salida Campo de Texto y Contraseña en el archivo datosUsuarioOB.xml y el archivo datosUsuarioOB.java con los parámetros del formulario.
58. El Framework TOOLWEB visualiza en el panel Código Servlet el archivo datosUsuarioOB.java.
59. El usuario-programador procede a editar en el panel Código Js el archivo JavaScript: Registro.js, adicionando una función denominada registrar Usuario.
60. El usuario-programador procede a editar en el panel Código xml el archivo Registro.xml, adicionando el evento onclick al botón Guardar, el mismo que hace referencia a la función registrarUsuario() del archivo Registro.js.
61. El usuario-programador seleccionar la opción Vista Previa, para actualizar los cambios que se realizaron del evento.
62. El usuario-programador selecciona la opción Nuevo Archivo del menú Archivo de la ventana TOOLWEB.
63. El Framework TOOLWEB presenta la ventana Nuevo Archivo.
64. El usuario-programador selecciona el tipo de archivo: xsl, ingresa el nombre del archivo: Registro1.
65. El Framework TOOLWEB ubica el archivo en el directorio respectivo a su tipo: /Registro/XSL.
66. El usuario-programador presiona el botón Aceptar en la ventana Nuevo Archivo.
67. El Framework TOOLWEB valida los datos tipo, nombre y ubicación.
68. El Framework TOOLWEB crea el archivo Registro1.css que se visualiza en el panel Código css.
69. El usuario-programador edita el archivo Registro.css utilizando los identificadores de los componentes del formulario, con la finalidad de dar estilo a la página datosUsuario.html.

70. El usuario-programador edita en el Panel Código Servlet la clase datosUsuarioOB.java adicionando la sentencia de ingreso para registrar al Usuario en la Base de Datos.
71. El usuario-programador abre el símbolo de Sistema para compilar la clase datosUsuarioOB.java.
72. El usuario-programador abre el navegador para visualizar la aplicación en el servidor Tomcat 7, previamente levantado el servidor, para esto abrimos el Símbolo del Sistema y agregamos el comando startup, en el directorio /Tomcat/bin.
73. El usuario-programador ingresa en la página de inicio de Apache Tomcat en la opción Manager App, introduce los datos del Nombre de Usuario (admin) y Contraseña (admin).
74. El navegador presenta la Ventana Gestor de Aplicaciones Web de Tomcat
75. El usuario-programador selecciona en Aplicaciones Trayectoria /Registro.
76. El usuario-programador procede a realizar la registración de los usuarios, ingresando los datos personales del usuario, luego se presiona el botón GUARDAR.
77. El Framework hace la conexión de la página datosUsuario.html con la función registrarUsuario() del archivo Registro.js el mismo que llama a la clase datosUsuarioOB.java la cual hace la sentencia de inserción en la tabla ingreso de la Base de Datos registro, dando de resultado el ingreso de datos del usuario.

CÓDIGO FUENTE DE LA APLICACIÓN WEB "REGISTRO"

Registro.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="XSL\Registro.xsl"?>
<componentes>
 <etiqueta>
  <identificador>reg</identificador>
  <texto>REGISTRO</texto>
  <eventos num="0">
   <evento>
    <nombre></nombre>
    <codigo></codigo>
   </evento>
  </eventos>
 </etiqueta>
```

```
<saltolinea>true</saltolinea>
<saltolinea>true</saltolinea>
<etiqueta>
  <identificador>us</identificador>
  <texto>USUARIO:</texto>
  <eventos num="0">
    <evento>
      <nombre></nombre>
      <codigo></codigo>
    </evento>
  </eventos>
</etiqueta>
<campotexto>
  <identificador>USER</identificador>
  <nombre>USER</nombre>
  <tipo>text</tipo>
  <valor>escriba el usuario</valor>
  <tamano>20</tamano>
  <maxlongitud>25</maxlongitud>
  <eventos num="0">
    <evento>
      <nombre></nombre>
      <codigo></codigo>
    </evento>
  </eventos>
</campotexto>
<saltolinea>true</saltolinea>
<saltolinea>true</saltolinea>
<etiqueta>
  <identificador>pass</identificador>
  <texto>CONTRASENA:</texto>
  <eventos num="0">
    <evento>
      <nombre></nombre>
      <codigo></codigo>
    </evento>
  </eventos>
</etiqueta>
<contrasena>
  <identificador>PASSWORD</identificador>
  <nombre>PASSWORD</nombre>
  <tipo>password</tipo>
  <valor>pass</valor>
  <tamano>20</tamano>
  <maxlongitud>25</maxlongitud>
  <eventos num="0">
    <evento>
      <nombre></nombre>
      <codigo></codigo>
    </evento>
  </eventos>
</contrasena>
```

```
        <saltolinea>true</saltolinea>
        <saltolinea>true</saltolinea>
        <enviarboton>
          <identificador>INGRESAR</identificador>
          <nombre>INGRESAR</nombre>
          <tipo>submit</tipo>
          <valor>INGRESAR</valor>
          <eventos num="1">
            <evento>
              <nombre>onclick</nombre>
              <codigo>comprobarUsuario()</codigo>
            </evento>
          </eventos>
        </enviarboton>
        <borrarboton>
          <identificador>LIMPIAR</identificador>
          <nombre>LIMPIAR</nombre>
          <tipo>reset</tipo>
          <valor>LIMPIAR</valor>
          <eventos num="0">
            <evento>
              <nombre>onclick</nombre>
              <codigo>registrarUsuario()</codigo>
            </evento>
          </eventos>
        </borrarboton>
        <saltolinea>true</saltolinea>
        <saltolinea>true</saltolinea>
        <hypervinculo>
          <identificador>registra</identificador>
          <texto>registrarse</texto>
          <enlace>/Registro/HTML/datosUsuario.html</enlace>
          <eventos num="0">
            <evento>
              <nombre></nombre>
              <codigo></codigo>
            </evento>
          </eventos>
        </hypervinculo>
</componentes>
```

Registro.xsl

```
        <?xml version="1.0" encoding="ISO-8859-1"?>
        <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
        version="1.0">
          <xsl:output method="html" />
          <xsl:template match="/">
            <html>
              <head>
                <title>Pagina Principal</title>
                <link rel="stylesheet" type="text/css" href="CSS/Registro.css" />
```

```
      <script type="text/javascript" src="JAVASCRIPT/Registro.js"></script>
    </head>
    <body>
      <h2>
        <center>Formulario</center>
      </h2>
      <form id="formu" name="formu" action="{accion}" method="post">
        <xsl:apply-templates />
      </form>
    </body>
  </html>
</xsl:template>
<xsl:template match="borrarboton">
  <xsl:choose>
    <xsl:when test="eventos/@num=1">
      <xsl:variable name="eventoCodigo">
        <xsl:value-of select="eventos/evento/nombre" />
      </xsl:variable>
      <input id="{identificador}" name="{nombre}" type="{tipo}" value="{valor}">
        <xsl:attribute name="{$eventoCodigo}">
          <xsl:value-of select="eventos/evento/codigo" />
        </xsl:attribute>
      </input>
    </xsl:when>
    <xsl:otherwise>
      <input id="{identificador}" name="{nombre}" type="{tipo}" value="{valor}" />
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>
<xsl:template match="boton">
  <xsl:choose>
    <xsl:when test="eventos/@num=1">
      <xsl:variable name="eventoCodigo">
        <xsl:value-of select="eventos/evento/nombre" />
      </xsl:variable>
      <input id="{identificador}" name="{nombre}" type="{tipo}" value="{valor}">
        <xsl:attribute name="{$eventoCodigo}">
          <xsl:value-of select="eventos/evento/codigo" />
        </xsl:attribute>
      </input>
    </xsl:when>
    <xsl:otherwise>
      <input id="{identificador}" name="{nombre}" type="{tipo}" value="{valor}" />
    </xsl:otherwise>
  </xsl:choose>
</xsl:template>
<xsl:template match="casillaverificacion">
  <xsl:choose>
    <xsl:when test="eventos/@num=1">
      <xsl:variable name="eventoCodigo">
        <xsl:value-of select="eventos/evento/nombre" />
      </xsl:variable>
```

```
        <xsl:choose>
          <xsl:when test="activar = 'true' ">
            <input id="{identificador}" name="{nombre}" type="{tipo}" value="{valor}"
checked="{activar}">
              <xsl:attribute name="{$eventoCodigo}">
                <xsl:value-of select="eventos/evento/codigo" />
              </xsl:attribute>
            </input>
          </xsl:when>
          <xsl:when test="desactivar = 'true' ">
            <input id="{identificador}" name="{nombre}" type="{tipo}" value="{valor}"
disabled="{desactivar}">
              <xsl:attribute name="{$eventoCodigo}">
                <xsl:value-of select="eventos/evento/codigo" />
              </xsl:attribute>
            </input>
          </xsl:when>
          <xsl:otherwise>
            <input id="{identificador}" name="{nombre}" type="{tipo}"
value="{valor}">
              <xsl:attribute name="{$eventoCodigo}">
                <xsl:value-of select="eventos/evento/codigo" />
              </xsl:attribute>
            </input>
          </xsl:otherwise>
        </xsl:choose>
      </xsl:when>
      <xsl:otherwise>
        <xsl:choose>
          <xsl:when test="activar = 'true' ">
            <input id="{identificador}" name="{nombre}" type="{tipo}" value="{valor}"
checked="{activar}" />
          </xsl:when>
          <xsl:when test="desactivar = 'true' ">
            <input id="{identificador}" name="{nombre}" type="{tipo}" value="{valor}"
disabled="{desactivar}" />
          </xsl:when>
          <xsl:otherwise>
            <input id="{identificador}" name="{nombre}" type="{tipo}" value="{valor}"
/>
          </xsl:otherwise>
        </xsl:choose>
      </xsl:otherwise>
    </xsl:choose>
  </xsl:template>
  <xsl:template match="enviarboton">
    <xsl:choose>
      <xsl:when test="eventos/@num=1">
        <xsl:variable name="eventoCodigo">
          <xsl:value-of select="eventos/evento/nombre" />
        </xsl:variable>
        <input id="{identificador}" name="{nombre}" type="{tipo}" value="{valor}">
```

```
          <xsl:attribute name="{$eventoCodigo}">
            <xsl:value-of select="eventos/evento/codigo" />
          </xsl:attribute>
        </input>
      </xsl:when>
      <xsl:otherwise>
        <input id="{identificador}" name="{nombre}" type="{tipo}" value="{valor}" />
      </xsl:otherwise>
    </xsl:choose>
  </xsl:template>
  <xsl:template match="formulario">
    <xsl:choose>
      <xsl:when test="eventos/@num=1">
        <xsl:variable name="eventoCodigo">
          <xsl:value-of select="eventos/evento/nombre" />
        </xsl:variable>
        <form id="{identificador}" name="{nombre}" action="{accion}"
method="{metodo}">
          <xsl:attribute name="{$eventoCodigo}">
            <xsl:value-of select="eventos/evento/codigo" />
          </xsl:attribute>
        </form>
      </xsl:when>
      <xsl:otherwise>
        <form id="{identificador}" name="{nombre}" action="{accion}"
method="{metodo}" />
      </xsl:otherwise>
    </xsl:choose>
  </xsl:template>
  <xsl:template match="hypervinculo">
    <xsl:choose>
      <xsl:when test="eventos/@num=1">
        <xsl:variable name="eventoCodigo">
          <xsl:value-of select="eventos/evento/nombre" />
        </xsl:variable>
        <a href="{enlace}" id="{identificador}">
          <xsl:attribute name="{$eventoCodigo}">
            <xsl:value-of select="eventos/evento/codigo" />
          </xsl:attribute>
          <xsl:value-of select="texto" />
        </a>
      </xsl:when>
      <xsl:otherwise>
        <a href="{enlace}" id="{identificador}">
          <xsl:value-of select="texto" />
        </a>
      </xsl:otherwise>
    </xsl:choose>
  </xsl:template>
  <xsl:template match="campoimagen">
    <xsl:choose>
      <xsl:when test="eventos/@num=1">
```

```
        <xsl:variable name="eventoCodigo">
          <xsl:value-of select="eventos/evento/nombre" />
        </xsl:variable>
        <input id="{identificador}" name="{nombre}" type="{tipo}" value="{valor}"
src="{direccion}">
          <xsl:attribute name="{$eventoCodigo}">
            <xsl:value-of select="eventos/evento/codigo" />
          </xsl:attribute>
        </input>
      </xsl:when>
      <xsl:otherwise>
        <input id="{identificador}" name="{nombre}" type="{tipo}" value="{valor}"
src="{direccion}" />
      </xsl:otherwise>
    </xsl:choose>
  </xsl:template>
  <xsl:template match="imagen">
    <xsl:choose>
      <xsl:when test="eventos/@num=1">
        <xsl:variable name="eventoCodigo">
          <xsl:value-of select="eventos/evento/nombre" />
        </xsl:variable>
        <img src="{direccion}" id="{identificador}">
          <xsl:attribute name="{$eventoCodigo}">
            <xsl:value-of select="eventos/evento/codigo" />
          </xsl:attribute>
        </img>
      </xsl:when>
      <xsl:otherwise>
        <img src="{direccion}" id="{identificador}" />
      </xsl:otherwise>
    </xsl:choose>
  </xsl:template>
  <xsl:template match="etiqueta">
    <xsl:choose>
      <xsl:when test="eventos/@num=1">
        <xsl:variable name="eventoCodigo">
          <xsl:value-of select="eventos/evento/nombre" />
        </xsl:variable>
        <label id="{identificador}">
          <xsl:attribute name="{$eventoCodigo}">
            <xsl:value-of select="eventos/evento/codigo" />
          </xsl:attribute>
          <xsl:value-of select="texto" />
        </label>
      </xsl:when>
      <xsl:otherwise>
        <label id="{identificador}">
          <xsl:value-of select="texto" />
        </label>
      </xsl:otherwise>
    </xsl:choose>
```

```
</xsl:template>
<xsl:template match="menulista">
  <xsl:choose>
    <xsl:when test="eventos/@num=1">
      <xsl:variable name="eventoCodigo">
        <xsl:value-of select="eventos/evento/nombre" />
      </xsl:variable>
      <xsl:choose>
        <xsl:when test="multiple = 'true' ">
          <select id="{identificador}" name="{nombre}" multiple="{multiple}">
            <xsl:attribute name="{$eventoCodigo}">
              <xsl:value-of select="eventos/evento/codigo" />
            </xsl:attribute>
            <xsl:for-each select="opciones/opcion">
              <option>
                <xsl:value-of select="." />
              </option>
            </xsl:for-each>
          </select>
        </xsl:when>
        <xsl:when test="desactivar = 'true' ">
          <select id="{identificador}" name="{nombre}" disabled="{desactivar}">
            <xsl:attribute name="{$eventoCodigo}">
              <xsl:value-of select="eventos/evento/codigo" />
            </xsl:attribute>
            <xsl:for-each select="opciones/opcion">
              <option>
                <xsl:value-of select="." />
              </option>
            </xsl:for-each>
          </select>
        </xsl:when>
        <xsl:otherwise>
          <select id="{identificador}" name="{nombre}">
            <xsl:attribute name="{$eventoCodigo}">
              <xsl:value-of select="eventos/evento/codigo" />
            </xsl:attribute>
            <xsl:for-each select="opciones/opcion">
              <option>
                <xsl:value-of select="." />
              </option>
            </xsl:for-each>
          </select>
        </xsl:otherwise>
      </xsl:choose>
    </xsl:when>
    <xsl:otherwise>
      <xsl:choose>
        <xsl:when test="multiple = 'true' ">
          <select id="{identificador}" name="{nombre}" multiple="{multiple}">
            <xsl:for-each select="opciones/opcion">
              <option>
```

```
                <xsl:value-of select="." />
              </option>
            </xsl:for-each>
          </select>
        </xsl:when>
        <xsl:when test="desactivar = 'true' ">
          <select id="{identificador}" name="{nombre}" disabled="{desactivar}">
            <xsl:for-each select="opciones/opcion">
              <option>
                <xsl:value-of select="." />
              </option>
            </xsl:for-each>
          </select>
        </xsl:when>
        <xsl:otherwise>
          <select id="{identificador}" name="{nombre}">
            <xsl:for-each select="opciones/opcion">
              <option>
                <xsl:value-of select="." />
              </option>
            </xsl:for-each>
          </select>
        </xsl:otherwise>
      </xsl:choose>
    </xsl:otherwise>
  </xsl:choose>
 </xsl:template>
 <xsl:template match="contrasena">
  <xsl:choose>
    <xsl:when test="eventos/@num=1">
      <xsl:variable name="eventoCodigo">
        <xsl:value-of select="eventos/evento/nombre" />
      </xsl:variable>
      <xsl:choose>
        <xsl:when test="soloLectura = 'true' ">
          <input id="{identificador}" name="{nombre}" type="{tipo}" value="{valor}"
size="{tamano}" maxlength="{maxlongitud}" readonly="{soloLectura}">
            <xsl:attribute name="{$eventoCodigo}">
              <xsl:value-of select="eventos/evento/codigo" />
            </xsl:attribute>
          </input>
        </xsl:when>
        <xsl:when test="desactivar = 'true' ">
          <input id="{identificador}" name="{nombre}" type="{tipo}" value="{valor}"
size="{tamano}" maxlength="{maxlongitud}" disabled="{desactivar}">
            <xsl:attribute name="{$eventoCodigo}">
              <xsl:value-of select="eventos/evento/codigo" />
            </xsl:attribute>
          </input>
        </xsl:when>
        <xsl:otherwise>
```

```
          <input id="{identificador}" name="{nombre}" type="{tipo}" value="{valor}"
size="{tamano}" maxlength="{maxlongitud}">
            <xsl:attribute name="{$eventoCodigo}">
              <xsl:value-of select="eventos/evento/codigo" />
            </xsl:attribute>
          </input>
        </xsl:otherwise>
      </xsl:choose>
    </xsl:when>
    <xsl:otherwise>
      <xsl:choose>
        <xsl:when test="soloLectura = 'true' ">
          <input id="{identificador}" name="{nombre}" type="{tipo}" value="{valor}"
size="{tamano}" maxlength="{maxlongitud}" readonly="{soloLectura}" />
        </xsl:when>
        <xsl:when test="desactivar = 'true' ">
          <input id="{identificador}" name="{nombre}" type="{tipo}" value="{valor}"
size="{tamano}" maxlength="{maxlongitud}" disabled="{desactivar}" />
        </xsl:when>
        <xsl:otherwise>
          <input id="{identificador}" name="{nombre}" type="{tipo}" value="{valor}"
size="{tamano}" maxlength="{maxlongitud}" />
        </xsl:otherwise>
      </xsl:choose>
    </xsl:otherwise>
  </xsl:choose>
 </xsl:template>
 <xsl:template match="radioboton">
  <xsl:choose>
    <xsl:when test="eventos/@num=1">
      <xsl:variable name="eventoCodigo">
        <xsl:value-of select="eventos/evento/nombre" />
      </xsl:variable>
      <xsl:choose>
        <xsl:when test="activar = 'true' ">
          <input id="{identificador}" name="{nombre}" type="{tipo}" value="{valor}"
checked="{activar}">
            <xsl:attribute name="{$eventoCodigo}">
              <xsl:value-of select="eventos/evento/codigo" />
            </xsl:attribute>
            <xsl:value-of select="texto" />
          </input>
          <BR />
        </xsl:when>
        <xsl:when test="desactivar = 'true' ">
          <input id="{identificador}" name="{nombre}" type="{tipo}" value="{valor}"
disabled="{desactivar}">
            <xsl:attribute name="{$eventoCodigo}">
              <xsl:value-of select="eventos/evento/codigo" />
            </xsl:attribute>
            <xsl:value-of select="texto" />
          </input>
```

```
          <BR />
         </xsl:when>
         <xsl:otherwise>
          <input id="{identificador}" name="{nombre}" type="{tipo}"
value="{valor}">
           <xsl:attribute name="{$eventoCodigo}">
            <xsl:value-of select="eventos/evento/codigo" />
           </xsl:attribute>
           <xsl:value-of select="texto" />
          </input>
          <BR />
         </xsl:otherwise>
        </xsl:choose>
      </xsl:when>
      <xsl:otherwise>
       <xsl:choose>
        <xsl:when test="activar = 'true' ">
         <input id="{identificador}" name="{nombre}" type="{tipo}" value="{valor}"
checked="{activar}">
          <xsl:value-of select="texto" />
         </input>
         <BR />
        </xsl:when>
        <xsl:when test="desactivar = 'true' ">
         <input id="{identificador}" name="{nombre}" type="{tipo}" value="{valor}"
disabled="{desactivar}">
          <xsl:value-of select="texto" />
         </input>
         <BR />
        </xsl:when>
        <xsl:otherwise>
         <input id="{identificador}" name="{nombre}" type="{tipo}"
value="{valor}">
          <xsl:value-of select="texto" />
         </input>
         <BR />
        </xsl:otherwise>
       </xsl:choose>
      </xsl:otherwise>
    </xsl:choose>
  </xsl:template>
  <xsl:template match="gruporadios">
    <xsl:for-each select="radios/radioboton">
      <label>
       <xsl:choose>
        <xsl:when test="eventos/@num=1">
         <xsl:variable name="eventoCodigo">
          <xsl:value-of select="eventos/evento/nombre" />
         </xsl:variable>
         <input id="{identificador}" name="{nombregrupo}" type="{tipo}"
value="{valor}">
          <xsl:attribute name="{$eventoCodigo}">
```

```
              <xsl:value-of select="eventos/evento/codigo" />
            </xsl:attribute>
          </input>
          <xsl:value-of select="nombre" />
        </xsl:when>
        <xsl:otherwise>
          <input id="{identificador}" name="{nombregrupo}" type="{tipo}"
value="{valor}" />
          <xsl:value-of select="nombre" />
        </xsl:otherwise>
      </xsl:choose>
    </label>
    <br />
  </xsl:for-each>
 </xsl:template>
 <xsl:template match="saltolinea">
  <br />
 </xsl:template>
 <xsl:template match="tabla">
  <xsl:choose>
    <xsl:when test="eventos/@num=1">
      <xsl:variable name="eventoCodigo">
        <xsl:value-of select="eventos/evento/nombre" />
      </xsl:variable>
      <table id="{identificador}" width="{ancho}" border="{borde}"
cellspacing="{espacio}" cellpadding="{relleno}">
        <xsl:attribute name="{$eventoCodigo}">
          <xsl:value-of select="eventos/evento/codigo" />
        </xsl:attribute>
        <tr />
        <xsl:for-each select="datostabla|datosfila">
          <xsl:apply-templates select="datostabla|datosfila" />
        </xsl:for-each>
      </table>
    </xsl:when>
    <xsl:otherwise>
      <table id="{identificador}" width="{ancho}" border="{borde}"
cellspacing="{espacio}" cellpadding="{relleno}">
        <tr />
        <xsl:for-each select="datostabla|datosfila">
          <xsl:apply-templates select="datostabla|datosfila" />
        </xsl:for-each>
      </table>
    </xsl:otherwise>
  </xsl:choose>
 </xsl:template>
 <xsl:template match="datostabla|datosfila">
  <tr>
    <xsl:apply-templates />
  </tr>
 </xsl:template>
 <xsl:template match="datoscolumna">
```

```
    <td>
      <xsl:value-of select="." />
    </td>
  </xsl:template>
  <xsl:template match="areatexto">
    <xsl:choose>
      <xsl:when test="eventos/@num=1">
        <xsl:variable name="eventoCodigo">
          <xsl:value-of select="eventos/evento/nombre" />
        </xsl:variable>
        <xsl:choose>
          <xsl:when test="soloLectura = 'true' ">
            <textarea id="{identificador}" name="{nombre}" cols="{columnas}"
rows="{filas}" readonly="{soloLectura}">
              <xsl:attribute name="{$eventoCodigo}">
                <xsl:value-of select="eventos/evento/codigo" />
              </xsl:attribute>
              texto
            </textarea>
          </xsl:when>
          <xsl:when test="desactivar = 'true' ">
            <textarea id="{identificador}" name="{nombre}" cols="{columnas}"
rows="{filas}" disabled="{desactivar}">
              <xsl:attribute name="{$eventoCodigo}">
                <xsl:value-of select="eventos/evento/codigo" />
              </xsl:attribute>
              texto
            </textarea>
          </xsl:when>
          <xsl:otherwise>
            <textarea id="{identificador}" name="{nombre}" cols="{columnas}"
rows="{filas}">
              <xsl:attribute name="{$eventoCodigo}">
                <xsl:value-of select="eventos/evento/codigo" />
              </xsl:attribute>
              texto
            </textarea>
          </xsl:otherwise>
        </xsl:choose>
      </xsl:when>
      <xsl:otherwise>
        <xsl:choose>
          <xsl:when test="soloLectura = 'true' ">
            <textarea id="{identificador}" name="{nombre}" cols="{columnas}"
rows="{filas}" readonly="{soloLectura}">texto</textarea>
          </xsl:when>
          <xsl:when test="desactivar = 'true' ">
            <textarea id="{identificador}" name="{nombre}" cols="{columnas}"
rows="{filas}" disabled="{desactivar}">texto</textarea>
          </xsl:when>
          <xsl:otherwise>
```

```
          <textarea id="{identificador}" name="{nombre}" cols="{columnas}"
rows="{filas}">texto</textarea>
        </xsl:otherwise>
      </xsl:choose>
    </xsl:otherwise>
  </xsl:choose>
 </xsl:template>
 <xsl:template match="campotexto">
  <xsl:choose>
    <xsl:when test="eventos/@num=1">
      <xsl:variable name="eventoCodigo">
        <xsl:value-of select="eventos/evento/nombre" />
      </xsl:variable>
      <xsl:choose>
        <xsl:when test="soloLectura = 'true' ">
          <input id="{identificador}" name="{nombre}" type="{tipo}" value="{valor}"
size="{tamano}" maxlength="{maxlongitud}" readonly="{soloLectura}">
            <xsl:attribute name="{$eventoCodigo}">
              <xsl:value-of select="eventos/evento/codigo" />
            </xsl:attribute>
          </input>
        </xsl:when>
        <xsl:when test="desactivar = 'true' ">
          <input id="{identificador}" name="{nombre}" type="{tipo}" value="{valor}"
size="{tamano}" maxlength="{maxlongitud}" disabled="{desactivar}">
            <xsl:attribute name="{$eventoCodigo}">
              <xsl:value-of select="eventos/evento/codigo" />
            </xsl:attribute>
          </input>
        </xsl:when>
        <xsl:otherwise>
          <input id="{identificador}" name="{nombre}" type="{tipo}" value="{valor}"
size="{tamano}" maxlength="{maxlongitud}">
            <xsl:attribute name="{$eventoCodigo}">
              <xsl:value-of select="eventos/evento/codigo" />
            </xsl:attribute>
          </input>
        </xsl:otherwise>
      </xsl:choose>
    </xsl:when>
    <xsl:otherwise>
      <xsl:choose>
        <xsl:when test="soloLectura = 'true' ">
          <input id="{identificador}" name="{nombre}" type="{tipo}" value="{valor}"
size="{tamano}" maxlength="{maxlongitud}" readonly="{soloLectura}" />
        </xsl:when>
        <xsl:when test="desactivar = 'true' ">
          <input id="{identificador}" name="{nombre}" type="{tipo}" value="{valor}"
size="{tamano}" maxlength="{maxlongitud}" disabled="{desactivar}" />
        </xsl:when>
        <xsl:otherwise>
```

```
            <input id="{identificador}" name="{nombre}" type="{tipo}" value="{valor}"
        size="{tamano}" maxlength="{maxlongitud}" />
            </xsl:otherwise>
          </xsl:choose>
        </xsl:otherwise>
      </xsl:choose>
    </xsl:template>
</xsl:stylesheet>
```

FormularioObjetos.xsl

```
        <?xml version="1.0" encoding="ISO-8859-1"?>
        <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
        version="1.0">
          <xsl:output method="text"></xsl:output>
          <xsl:variable name="className"
        select="/formularioObjetos/@clase"></xsl:variable>
          <!--  ********************************************************************************
          **   Generar la clase servelt con los  objetos  del formulario
          **   que se utilizan para realizar las operaciones en la BD.
          ********************************************************************************
        -->
          <xsl:template match="/formularioObjetos">
            /*
             *  Clase de los objetos del Formulario.
             */
        import java.io.IOException;
        import java.io.PrintWriter;
        import java.sql.*;
        import java.sql.Connection;
        import java.sql.DriverManager;
        import java.sql.ResultSet;
        import java.sql.SQLException;
        import javax.servlet.ServletConfig;
        import javax.servlet.ServletException;
        import javax.servlet.http.HttpServlet;
        import javax.servlet.http.HttpServletRequest;
        import javax.servlet.http.HttpServletResponse;
           <xsl:value-of select="concat('public class ', $className, ' extends
        HttpServlet {')"></xsl:value-of>
           PrintWriter pagina;
           Connection canal = null;
           ResultSet tabla= null;
           Statement instruccion;

           public void init(ServletConfig conf) throws ServletException {
               super.init(conf);
           }

           public void doGet(HttpServletRequest request, HttpServletResponse
        response)     throws ServletException, IOException{
                processRequest(request, response);
```

```
    }

    public void doPost(HttpServletRequest request, HttpServletResponse
response)     throws ServletException, IOException{
        processRequest(request, response);
    }

    public void processRequest(HttpServletRequest request,
HttpServletResponse response)     throws ServletException, IOException{
        response.setContentType("text/html");
        pagina = response.getWriter();

         //Conexion a la base de datos
    <xsl:value-of select="concat('String nombreBD = &quot;', @nombreBD,
'&quot;;&#xA;')"></xsl:value-of>
    <xsl:value-of select="concat('String usuarioBD = &quot;', @usuarioBD,
'&quot;;&#xA;')"></xsl:value-of>
    <xsl:value-of select="concat('String contrasenaBD = &quot;',
@contrasenaBD, '&quot;;&#xA;')"></xsl:value-of>
    String servidor = "jdbc:mysql://localhost:3306/".concat(nombreBD);
    try { Class.forName("com.mysql.jdbc.Driver");
        canal =  DriverManager.getConnection(servidor, usuarioBD,
contrasenaBD);
        instruccion =   (Statement) canal.createStatement();
    } catch(java.lang.ClassNotFoundException e){} catch(SQLException e) {};

// Elementos a cargar en la base de datos.
    <xsl:apply-templates select="propiedad"
mode="generateRequest"></xsl:apply-templates>
    <xsl:text>// Sentencias para la base de datos:

        pagina.close();

   }
}</xsl:text>
  </xsl:template>
  <!--  ************************************************************************
  **     Generar los objetos del formulario
  **     con su correspondiente variable.
  ************************************************************************
-->
  <xsl:template match="propiedad" mode="generateRequest">
    <xsl:value-of select="concat('    ', @tipo, ' ')"></xsl:value-of>
    <xsl:value-of select="concat(' ', @nombre, ' = ')"></xsl:value-of>
    <xsl:text>request.getParameter("</xsl:text>
    <xsl:value-of select="concat(@nombre, '&quot;);&#xA;')"></xsl:value-of>
  </xsl:template>
</xsl:stylesheet>
```

RegistroOB.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

```
<formularioObjetos clase="RegistroOB" nombreBD="registro" usuarioBD="root"
contrasenaBD="">
      <propiedad nombre="USER" tipo="String" />
      <propiedad nombre="PASSWORD" tipo="String" />
</formularioObjetos>
```

datosUsuarioOB.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<formularioObjetos clase="datosUsuarioOB" nombreBD="registro"
usuarioBD="root" contrasenaBD="">
  <propiedad nombre="NOMBRES" tipo="String" />
  <propiedad nombre="APELLIDOS" tipo="String" />
  <propiedad nombre="DIRECCION" tipo="String" />
  <propiedad nombre="USER" tipo="String" />
  <propiedad nombre="PASSWORD" tipo="String" />
</formularioObjetos>
```

RegistroOB.java

```
/*
 *  Clase de los objetos del Formulario.
 */
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.*;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class RegistroOB extends HttpServlet {
   PrintWriter pagina;
   Connection canal = null;
   ResultSet tabla= null;
   Statement instruccion;

   public void init(ServletConfig conf) throws ServletException {
       super.init(conf);
   }

   public void doGet(HttpServletRequest request, HttpServletResponse
response)     throws ServletException, IOException{
       processRequest(request, response);
   }
```

```
   public void doPost(HttpServletRequest request, HttpServletResponse
response)     throws ServletException, IOException{
        processRequest(request, response);
   }

   public void processRequest(HttpServletRequest request,
HttpServletResponse response)     throws ServletException, IOException{
        response.setContentType("text/html");
        pagina = response.getWriter();

        //Conexion a la base de datos
       String nombreBD = "registro";
       String usuarioBD = "root";
       String contrasenaBD = "";
       String servidor = "jdbc:mysql://localhost:3306/".concat(nombreBD);
       try { Class.forName("com.mysql.jdbc.Driver");
           canal =  DriverManager.getConnection(servidor, usuarioBD,
contrasenaBD);
           instruccion =   (Statement) canal.createStatement();
       } catch(java.lang.ClassNotFoundException e){} catch(SQLException e) {};
       // Elementos a cargar en la base de datos.
       String nombres = request.getParameter("NOMBRES");
       String apellidos = request.getParameter("APELLIDOS");
       String direccion = request.getParameter("DIRECCION");
       String user = request.getParameter("USER");
       String password = request.getParameter("PASSWORD");
       // Sentencias para la base de datos:
       String sentencia = "";
       try{
         // agregando sentencia de consulta a la BD (select)
          sentencia ="select * from ingreso where user ="+"'"+USER+"'  AND
password = "+"'"+PASSWORD+"';";
          System.out.println("la cadena es:  "+sentencia);
       }catch(Exception ex){
          System.out.println(ex.getMessage());
       }
       try {
           tabla=instruccion.executeQuery(sentencia);
           String us = null;
            int columna_usuario = tabla.findColumn("user");
            boolean lleno = tabla.next();
            while (lleno){
                  us = tabla.getString(columna_usuario);
                  lleno = tabla.next();
            }
            if(us==null){
                  pagina.println("No esta registrado el usuario,   "+USER+"   BACK
PARA REGRESAR y REGISTRESE");
            }else
                  pagina.println("Si esta registrado el usuario,   "+us+"    BACK
PARA REGRESAR  ");
            } catch(SQLException e) {}
```

```
                try {
                    canal.close();
                    instruccion.close();
                } catch(SQLException e) {}
                pagina.close();
            }
}
```

datosUsuarioOB.java

```
            /*
             *  Clase de los objetos del Formulario.
             */
        import java.io.IOException;
        import java.io.PrintWriter;
        import java.sql.*;
        import java.sql.Connection;
        import java.sql.DriverManager;
        import java.sql.ResultSet;
        import java.sql.SQLException;
        import javax.servlet.ServletConfig;
        import javax.servlet.ServletException;
        import javax.servlet.http.HttpServlet;
        import javax.servlet.http.HttpServletRequest;
        import javax.servlet.http.HttpServletResponse;

        public class datosUsuarioOB extends HttpServlet {
            PrintWriter pagina;
            Connection canal = null;
            ResultSet tabla= null;
            Statement instruccion;

            public void init(ServletConfig conf) throws ServletException {
                super.init(conf);
            }

            public void doGet(HttpServletRequest request, HttpServletResponse
        response)     throws ServletException, IOException{
                processRequest(request, response);
            }

            public void doPost(HttpServletRequest request, HttpServletResponse
        response)     throws ServletException, IOException{
                processRequest(request, response);
            }

            public void processRequest(HttpServletRequest request,
        HttpServletResponse response)     throws ServletException, IOException{
                response.setContentType("text/html");
                pagina = response.getWriter();

                //Conexion a la base de datos
```

```
            String nombreBD = "registro";
            String usuarioBD = "root";
            String contrasenaBD = "";
            String servidor = "jdbc:mysql://localhost:3306/".concat(nombreBD);
            try { Class.forName("com.mysql.jdbc.Driver");
                canal =  DriverManager.getConnection(servidor, usuarioBD,
        contrasenaBD);
                instruccion =   (Statement) canal.createStatement();
            } catch(java.lang.ClassNotFoundException e){} catch(SQLException e) {};

            // Elementos a cargar en la base de datos.
            String  NOMBRES = request.getParameter("NOMBRES");
            String  APELLIDOS = request.getParameter("APELLIDOS");
            String  DIRECCION = request.getParameter("DIRECCION");
           String  USER = request.getParameter("USER");
           String  PASSWORD = request.getParameter("PASSWORD");
            // Sentencias para la base de datos:
            String sentencia = "";
            try{
               // agregando sentencia de insercion a la BD (insert)
                sentencia ="insert into ingreso values("+"'"+NOMBRES+"'"+","+"'"+
        APELLIDOS+"'"+","+"'"+ DIRECCION+"'"+","+"'"+USER +"'"+","+"'"+
        PASSWORD+"'"+") ";
           }catch(Exception ex){
                System.out.println(ex.getMessage());
           }
           try {
               int n=instruccion.executeUpdate(sentencia);
                    pagina.println("YA SE INSERTO EL USUARIO,    "+user+"
        "+"BACK PARA REGRESAR");

            } catch(SQLException e) {}

            try {
                canal.close();
                instruccion.close();
            } catch(SQLException e) {}
             pagina.close();
         }
}
```

Index.html

```
<html>
  <head>
    <title>Pagina Principal</title>
    <link rel="stylesheet" type="text/css" href="CSS/Registro.css" />
    <script type="text/javascript" src="JAVASCRIPT/Registro.js"></script>
  </head>
  <body>
    <h2>
      <center>Formulario</center>
    </h2>
```

```
        <form id="formu" name="formu" action="" method="post">
          <label id="reg">REGISTRO</label>
          <br />
          <br />
          <label id="us">USUARIO:</label>
          <input id="USER" name="USER" type="text" value="escriba el usuario" size="20"
      maxlength="25" />
          <br />
          <br />
          <label id="pass">CONTRASENA:</label>
          <input id="PASSWORD" name="PASSWORD" type="password" value="pass"
      size="20" maxlength="25" />
          <br />
          <br />
          <input id="INGRESAR" name="INGRESAR" type="submit" value="INGRESAR"
      onclick="comprobarUsuario()" />
          <input id="LIMPIAR" name="LIMPIAR" type="reset" value="LIMPIAR" />
          <br />
          <br />
          <a href="/Registro/HTML/datosUsuario.html" id="registra">registrarse</a>
        </form>
      </body>
</html>
```

datosUsuario.html

```
      <html>
        <head>
          <title>Pagina Principal</title>
          <link rel="stylesheet" type="text/css" href="../CSS/Registro1.css" />
          <script type="text/javascript" src="JAVASCRIPT/Registro.js"></script>
        </head>
        <body>
          <h2>
            <center>Formulario</center>
          </h2>
          <form id="formu" name="formu" action="" method="post">
            <label id="datos">DATOS PERSONALES</label>
            <br />
            <br />
            <label id="nom">NOMBRES:</label>
            <input id="NOMBRES" name="NOMBRES" type="text" value="escribir nombre"
      size="20" maxlength="25" />
            <br />
            <br />
            <label id="apell">APELLIDOS:</label>
            <input id="APELLIDOS" name="APELLIDOS" type="text" value="escriba apellido"
      size="20" maxlength="25" />
            <br />
            <br />
            <label id="us">USUARIO:</label>
            <input id="USER" name="USER" type="text" value="escriba usuario" size="20"
      maxlength="25" />
            <br />
            <br />
            <label id="pass">PASSWORD</label>
```

```
        <input id="PASSWORD" name="PASSWORD" type="password" value="pass"
      size="20" maxlength="25" />
        <br />
        <br />
        <input id="GUARDAR" name="GUARDAR" type="submit" value="GUARDAR"
      onclick="registrarUsuario()"/>
        <input id="LIMPIAR" name="LIMPIAR" type="reset" value="LIMPIAR" />
       </form>
      </body>
</html>
```

Proyecto_Registro.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<Proyecto>
  <Lista_Archivos>
    <XML>
      <Nombre_Archivo_XML>Registro.xml</Nombre_Archivo_XML>
      <Nombre_Ubicacion_XML>C:\TOMCAT\apache-tomcat-
7.0.23\webapps\Registro\XML\Registro.xml</Nombre_Ubicacion_XML>
    </XML>
    <XSL>
      <Nombre_Archivo_XSL>Registro.xsl</Nombre_Archivo_XSL>
      <Nombre_Ubicacion_XSL>C:\TOMCAT\apache-tomcat-
7.0.23\webapps\Registro\XSL\Registro.xsl</Nombre_Ubicacion_XSL>
    </XSL>
    <XML>
      <Nombre_Archivo_XML>RegistroOB.xml</Nombre_Archivo_XML>
      <Nombre_Ubicacion_XML>C:\TOMCAT\apache-tomcat-
7.0.23\webapps\Registro\XML\RegistroOB.xml</Nombre_Ubicacion_XML>
    </XML>
    <XSL>
      <Nombre_Archivo_XSL>FormularioObjetos.xsl</Nombre_Archivo_XSL>
      <Nombre_Ubicacion_XSL>C:\TOMCAT\apache-tomcat-
7.0.23\webapps\Registro\XSL\FormularioObjetos.xsl</Nombre_Ubicacion_XSL
>
    </XSL>
    <Servlet>
      <Nombre_Archivo_Servlet>RegistroOB.java</Nombre_Archivo_Servlet>
      <Nombre_Ubicacion_Servlet>C:\TOMCAT\apache-tomcat-
7.0.23\webapps\Registro\WEB-
INF\classes\RegistroOB.java</Nombre_Ubicacion_Servlet>
    </Servlet>
    <HTML>
      <Nombre_Archivo_HTML>index.html</Nombre_Archivo_HTML>
      <Nombre_Ubicacion_HTML>C:\TOMCAT\apache-tomcat-
7.0.23\webapps\Registro\index.html</Nombre_Ubicacion_HTML>
    </HTML>
    <JavaScript>
      <Nombre_Archivo_JavaScript>Registro.js</Nombre_Archivo_JavaScript>
      <Nombre_Ubicacion_JavaScript>C:\TOMCAT\apache-tomcat-
7.0.23\webapps\Registro\JAVASCRIPT\Registro.js</Nombre_Ubicacion_JavaS
cript>
```

```
  </JavaScript>
  <CSS>
    <Nombre_Archivo_CSS>Registro.css</Nombre_Archivo_CSS>
    <Nombre_Ubicacion_CSS>C:\TOMCAT\apache-tomcat-
7.0.23\webapps\Registro\CSS\Registro.css</Nombre_Ubicacion_CSS>
  </CSS>
  <Servlet>
    <Nombre_Archivo_Servlet>Conexion.java</Nombre_Archivo_Servlet>
    <Nombre_Ubicacion_Servlet>C:\TOMCAT\apache-tomcat-
7.0.23\webapps\Registro\WEB-
INF\classes\Conexion.java</Nombre_Ubicacion_Servlet>
  </Servlet>
       <XML>
    <Nombre_Archivo_XML>conexionbd_registro.xml</Nombre_Archivo_XML>
    <Nombre_Ubicacion_XML>C:\TOMCAT\apache-tomcat-
7.0.23\webapps\Registro</Nombre_Ubicacion_XML>
  </XML>
  <XML>
    <Nombre_Archivo_XML>datosUsuario.xml</Nombre_Archivo_XML>
    <Nombre_Ubicacion_XML>C:\TOMCAT\apache-tomcat-
7.0.23\webapps\Registro\XML\datosUsuario.xml</Nombre_Ubicacion_XML>
  </XML>
  <XML>
    <Nombre_Archivo_XML>datosUsuarioOB.xml</Nombre_Archivo_XML>
    <Nombre_Ubicacion_XML>C:\TOMCAT\apache-tomcat-
7.0.23\webapps\Registro\XML\datosUsuarioOB.xml</Nombre_Ubicacion_XML
>
  </XML>
  <Servlet>

<Nombre_Archivo_Servlet>datosUsuarioOB.java</Nombre_Archivo_Servlet>
    <Nombre_Ubicacion_Servlet>C:\TOMCAT\apache-tomcat-
7.0.23\webapps\Registro\XML\datosUsuarioOB.java</Nombre_Ubicacion_Servl
et>
  </Servlet>
  <HTML>
    <Nombre_Archivo_HTML>datosUsuario.html</Nombre_Archivo_HTML>
    <Nombre_Ubicacion_HTML>C:\TOMCAT\apache-tomcat-
7.0.23\webapps\Registro\HTML\datosUsuario.html</Nombre_Ubicacion_HTML
>
  </HTML>
    <CSS>
    <Nombre_Archivo_CSS>Registro1.css</Nombre_Archivo_CSS>
    <Nombre_Ubicacion_CSS>C:\TOMCAT\apache-tomcat-
7.0.23\webapps\Registro\CSS\Registro1.css</Nombre_Ubicacion_CSS>
  </CSS>
  <Servlet>

<Nombre_Archivo_Servlet>datosUsuarioOB.java</Nombre_Archivo_Servlet>
    <Nombre_Ubicacion_Servlet>C:\TOMCAT\apache-tomcat-
7.0.23\webapps\Registro\XML\datosUsuarioOB.java</Nombre_Ubicacion_Servl
et>
```

```
    </Servlet>
  </Lista_Archivos>
  <Nombre_Proyecto>Registro</Nombre_Proyecto>
  <Ubicacion_Proyecto>C:\TOMCAT\apache-tomcat-
7.0.23\webapps\Registro</Ubicacion_Proyecto>
</Proyecto>
```

conexionbd_registro.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<conexion>
     <nombrebd>registro</nombrebd>
    <usuariobd>root</usuariobd>
    <contrasenabd></contrasenabd>
    <servidor>localhost</servidor>
</conexion>
```

Conexion.java

```
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.*;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class Conexion extends HttpServlet {

   @Override
   public void init(ServletConfig conf) throws ServletException {
      super.init(conf);
   }

   @Override
   public void doGet(HttpServletRequest request, HttpServletResponse
response)throws ServletException, IOException{
      processRequest(request, response);
   }

   @Override
   public void doPost(HttpServletRequest request, HttpServletResponse
response)throws ServletException, IOException{
      processRequest(request, response);
   }
```

```
    public void processRequest(HttpServletRequest request,
HttpServletResponse response)throws ServletException, IOException{
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<body>");
        out.println("<h1>BIENVENIDO A TOOLWEB</h1>");
        out.println("</body>");
        out.println("</html>");
    }
}
```

Registro.css

```
/* CSS Document */
body{
background-color:#d0e4fe;
}

/*index.html*/
#reg{
font-family:"Times New Roman";
font-size:30px;
color: #345;
}

#us{
font-family:"Arial";
font-size:15px;
color: #2F3BC1;
}

#pass{
font-family:"Arial";
font-size:15px;
color: #2F3BC1;
}

#USER{
margin-left: 3.7em;
}

#PASSWORD{
margin-left: 1em;
}

#INGRESAR{
padding-top: 1;
margin-left: 2em;
margin-right: 10px;
border-bottom-style:ridge;
border-color: #4956E2;
background-color:#929192;
```

```
	}

	#LIMPIAR{
	padding-top: 1;
	margin-left: 2em;
	margin-right: 10px;
	border-bottom-style:ridge;
	border-color: #4956E2;
	background-color:#929192;
	}

	#registra{
	font-family:Georgia,serif;
	font-size:18px;
	color: #345;
}
```

Registro1.css

```
	/* CSS Document */
	body{
	background-color:#d0e4fe;
	}

	/*datosUsuario.html*/
	#datos{
	font-family:"Times New Roman";
	font-size:30px;
	color: #345;
	}

	#nom{
	font-family:"Arial";
	font-size:15px;
	color: #2F3BC1;
	}

	#apell{
	font-family:"Arial";
	font-size:15px;
	color: #2F3BC1;
	}

	#dir{
	font-family:"Arial";
	font-size:15px;
	color: #2F3BC1;
	}

	#us{
	font-family:"Arial";
	font-size:15px;
	color: #2F3BC1;
```

```
}

#pass{
font-family:"Arial";
font-size:15px;
color: #2F3BC1;
}
#NOMBRES{
margin-left: 2em;
}

#APELLIDOS{
margin-left: 1.5em;
}

#DIRECCION{
margin-left: 0.7em;
}

#USER{
margin-left: 1.9em;
}

#PASSWORD{
margin-left: 0.2em;
}

#GUARDAR{
padding-top: 1;
margin-left: 2em;
margin-right: 10px;
border-bottom-style:ridge;
border-color: #4956E2;
background-color:#929192;
}

#LIMPIAR{
padding-top: 1;
margin-left: 2em;
margin-right: 10px;
border-bottom-style:ridge;
border-color: #4956E2;
background-color:#929192;
}
```

Registro.js

```
// JavaScript Document

// Function del objeto HttpRequest
function ajax(){
   var xmlhttprequest = false;
```

```
        if (window.XMLHttpRequest){
            // Si es Mozilla, Safari etc
            xmlhttprequest = new XMLHttpRequest();
        } else if (window.ActiveXObject){
            // pero si es IE
            try{
                xmlhttprequest = new ActiveXObject ("Msxml2.XMLHTTP")
            }catch (e){
                //en caso que sea una versión antigua
                try{
                    xmlhttprequest = new ActiveXObject ("Microsoft.XMLHTTP")
                }catch (e){
                }
            }
        }
        return xmlhttprequest;
    }

    function cargarMensaje() {
        ajax();
        formu.action ="/Registro/Conexion";
        formu.submit();
    }

    function comprobarUsuario() {
        ajax();
        formu.action ="/Registro/RegistroOB";
        formu.submit();
    }

    function registrarUsuario() {
        ajax();
        formu.action ="/Registro/datosUsuarioOB";
        formu.submit();
}
```

context.xml

```
    <?xml version="1.0" encoding="ISO-8859-1"?>
    <!--
    Licensed to the Apache Software Foundation (ASF) under one or more
    contributor license agreements.  See the NOTICE file distributed with
    this work for additional information regarding copyright ownership.
    The ASF licenses this file to You under the Apache License, Version 2.0
    (the "License"); you may not use this file except in compliance with
    the License.  You may obtain a copy of the License at
    http://www.apache.org/licenses/LICENSE-2.0
    Unless required by applicable law or agreed to in writing, software distributed
    under the License is distributed on an "AS IS" BASIS, WITHOUT
    WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
    See the License for the specific language governing permissions and limitations
    under the License.
    -->
```

```
<Context docBase="${catalina.home}/webapps/Registro" privileged="true"
antiResourceLocking="false" antiJarLocking="false" />
```

web.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" version="3.0">
  <!--  Tus definiciones van aqui  -->
  <servlet xmlns="">
    <servlet-name>Conexion</servlet-name>
    <servlet-class>Conexion</servlet-class>
  </servlet>
  <servlet-mapping xmlns="">
    <servlet-name>Conexion</servlet-name>
    <url-pattern>/Conexion</url-pattern>
  </servlet-mapping>
  <servlet xmlns="">
    <servlet-name>RegistroOB.java</servlet-name>
    <servlet-class>RegistroOB.java</servlet-class>
  </servlet>
  <servlet-mapping xmlns="">
    <servlet-name>RegistroOB.java</servlet-name>
    <url-pattern>/RegistroOB.java</url-pattern>
  </servlet-mapping>
  <servlet xmlns="">
    <servlet-name>datosUsuarioOB.java</servlet-name>
    <servlet-class>datosUsuarioOB.java</servlet-class>
  </servlet>
  <servlet-mapping xmlns="">
    <servlet-name>datosUsuarioOB.java</servlet-name>
    <url-pattern>/datosUsuarioOB.java</url-pattern>
  </servlet-mapping>
</web-app>
```