DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA

CARRERA DE INGENIERÍA EN SISTEMAS E INFORMÁTICA

PROYECTO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN SISTEMAS E INFORMÁTICA

TEMA: DESARROLLO DE APLICACIONES WEB INTERACTIVAS APLICANDO LA TÉCNICA AJAX

AUTOR: MERY SUSANA ZAMBONINO BAUTISTA

DIRECTOR: ING. XAVIER MONTALUISA

CODIRECTOR: ING. MARCELO ÁLVAREZ

LATACUNGA

2015

UNIVERSIDAD DE LAS FUERZAS ARMADAS - ESPE
CARRERA DE INGENIERÍA EN SISTEMAS E INFORMÁTICA

CERTIFICADO

ING. XAVIER MONTALUISA (DIRECTOR)
ING. MARCELO ÁLVAREZ (CODIRECTOR)

CERTIFICAN:

Que el trabajo titulado "DESARROLLO DE APLICACIONES WEB INTERACTIVAS APLICANDO LA TÉCNICA AJAX" realizado por la señorita: MERY SUSANA ZAMBONINO BAUTISTA, ha sido guiado y revisado periódicamente y cumple normas estatutarias establecidas por la ESPE, en el Reglamento de Estudiantes de la Universidad de las Fuerzas Armadas - ESPE.

Debido a que constituye un trabajo de excelente contenido científico que coadyuvará a la aplicación de conocimientos y al desarrollo profesional, SI recomiendan su publicación.

Latacunga, abril del 2015.

Ing. Xavier Montaluisa	Ing. Marcelo Álvarez
DIRECTOR	CODIRECTOR

UNIVERSIDAD DE LAS FUERZAS ARMADAS - ESPE
CARRERA DE INGENIERÍA EN SISTEMAS E INFORMÁTICA

DECLARACIÓN DE RESPONSABILIDAD

Yo, MERY SUSANA ZAMBONINO BAUTISTA

DECLARO QUE:

El proyecto de grado denominado "DESARROLLO DE APLICACIONES WEB INTERACTIVAS APLICANDO LA TÉCNICA AJAX" ha sido desarrollado con base a una investigación exhaustiva, respetando derechos intelectuales de terceros, conforme las citas que constan en las páginas correspondientes, cuyas fuentes se incorporan en la bibliografía.

Consecuentemente este trabajo es de mi autoría

En virtud de esta declaración, me responsabilizo del contenido, veracidad y alcance científico del proyecto de grado en mención.

Latacunga, abril del 2015

Mery Susana Zambonino Bautista
C.C.: 0502357486

UNIVERSIDAD DE LAS FUERZAS ARMADAS - ESPE
CARRERA DE INGENIERÍA EN SISTEMAS E INFORMÁTICA

AUTORIZACIÓN

Yo, MERY SUSANA ZAMBONINO BAUTISTA

Autorizo a la Universidad de las Fuerzas Armadas - ESPE la publicación, del trabajo "DESARROLLO DE APLICACIONES WEB INTERACTIVAS APLICANDO LA TÉCNICA AJAX" cuyo contenido, ideas y criterios son de mi exclusiva responsabilidad y autoría.

Latacunga, abril del 2015

_____

Mery Susana Zambonino Bautista
C.C.: 0502357486

DEDICATORIA

Este proyecto se lo dedicó en primer lugar a mi Dios, quién ha sido esa voz interna de aliento en cada obstáculo a través de su hijo Jesucristo.

A mis padres Reinaldo y Gladys que han estado ahí en cada paso durante mis estudios y en especial a mi madre quién ha sido ese ángel que guía y cuida cada una de mis acciones.

A mis hermanos Geovanny y Gabriela con quienes he compartido los más hermosos momentos en medio de tristezas y alegrías.

A mi cuñada y mis sobrinas quienes han sido ese apoyo incondicional en todo tiempo.

A cada uno de mis tíos y primos quienes han sido ese empuje para seguir adelante y culminar este proyecto.

A mis amigos quienes han estado ahí para apoyarme y colaborar en lo necesario.

Mery Zambonino

AGRADECIMIENTO

Mi agradecimiento especial a Dios por haberme darme este regalo de vida y por su amor y sabiduría en este proyecto.

Agradezco a mis padres y a mis hermanos que siempre han estado para apoyarme y darme un aliento de esperanza en cada paso.

A mis amigos y guías espirituales que me han apoyado en todo tiempo con sus sabios consejos y conocimientos.

A los Ingenieros Xavier Montaluisa y Marcelo Álvarez, Director y Codirector de Tesis respectivamente por todos sus conocimientos académicos y colaboración para poder culminar este proyecto.

Mery Zambonino

ÍNDICE DE CONTENIDOS

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

OBJETIVO GENERAL

Desarrollar aplicaciones web utilizando la técnica AJAX demostrando su eficacia, eficiencia y rendimiento mediante un sitio web, indicando que en Internet también pueden encontrarse aplicaciones interactivas con un desempeño y rendimiento similar al de las aplicaciones de escritorio.

OBJETIVOS ESPECÍFICOS

- Investigar la evolución sobre el desarrollo de aplicaciones web dinámicas.
- Indagar sobre la técnica AJAX en el desarrollo de aplicaciones web dinámicas.
- Conocer las tecnologías que comprenden la técnica AJAX y sus beneficios.
- Desarrollar un sitio web dinámico aplicando lo anteriormente investigado.

RESUMEN

En el desarrollo de aplicaciones web dinámicas cuyo funcionamiento en tiempos óptimos de respuesta se ha encontrado como efectivo la aplicación de la técnica AJAX que es un acrónimo de Javascript asíncrono y XML, el mismo que utiliza varias tecnologías que ayudan a que el desarrollo de dichas aplicaciones sea con efectividad; AJAX se ha utilizado en la presente tesis como una técnica para desarrollar aplicaciones dinámicas que ayuden a que el usuario no permanezca largos ratos esperando por el retorno de una página, pues ésta técnica hace que rutinas y script viajen al servidor en busca de datos para actualizar ciertas partes y así mostrar u ocultar ciertas porciones; está compuesta de cuatro tecnologías como son: XHTML y CSS para el diseño, DOM que es una jerarquía de objetos los mismos que describen los elemento de una página web y también las características en el proceso de navegación, JAVAScript que es un lenguaje de programación interpretado y XML que corresponde a un estándar que permite a diferentes aplicaciones interactuar con facilidad a través de la red; durante el proceso de desarrollo además se ha utilizado la metodología UWE basado en UML, y su enfoque principal sobre nivel de Análisis y Diseño. La arquitectura refleja un modelo de tres capas: de presentación, negocios y acceso a datos; y como un punto muy importante a destacar se encuentra el uso de Entity Framework que es el que traslada cada tabla de la base de datos y los convierte en objetos.

PALABRAS CLAVE:

- AJAX – DESARROLLO WEB
- APLICACIONES WEB
- DINÁMICA DE PÁGINAS WEB
- METODOLOGÍA UWE
- JAVASCRIPT

ABSTRACT

In the development of dynamic web applications that operate at optimal response times have been found to be effective the implementation of the AJAX technique which is an acronym for Asynchronous Javascript and XML , it uses several technologies that help the development of such applications it effectively. AJAX has been used in this thesis as a technique for developing rich applications that help the user not remain long hours waiting for the return of a page , as this technique makes traveling script and routines to the server for data to update parts and show or hide certain portions ; It is composed by four technologies such as: XHTML and CSS are used for design, DOM is a hierarchy of objects that describe the same element of a web page and also features in the navigation process , Javascript is a interpreted programming language and XML corresponding to a standard that allows different applications to interact with ease through the network. In the process of development has also used a methodology known as UWE based on software engineering UML, It has a primary focus on the level of analysis and design. The architecture reflects a model of three layers: presentation , business and data access ; and as a very important point to note is the use of Entity Framework which is what moves each table in the database and converts them into objects.

KEY WORDS:

- AJAX WEB DEVELOPMENT
- WEB APPLICATIONS
- WEB PAGES DYNAMICS
- UWE METHODOLOGY
- JAVASCRIPT

CAPÍTULO I

1. EVOLUCIÓN EN EL DESARROLLO DE APLICACIONES WEB DINÁMICAS

1.1. Introducción

Con la introducción de Internet y de la Web en concreto, se han abierto infinidad de posibilidades en cuanto al acceso a la información desde cualquier sitio. Esto representa un desafío a los desarrolladores de aplicaciones, ya que los avances en tecnología demandan cada vez aplicaciones más rápidas, ligeras y robustas que permitan utilizar la Web.

Es importante mencionar que una página Web puede contener elementos que permiten una comunicación activa entre el usuario y la información. Esto permite que el usuario acceda a los datos de modo interactivo, gracias a que la página responderá a cada una de sus acciones, como por ejemplo rellenar y enviar formularios, participar en juegos diversos y acceder a gestores de base de datos de todo tipo.

En un principio la web era sencillamente una colección de páginas estáticas, documentos; para su consulta o descarga. El paso inmediatamente posterior en su evolución fue la inclusión de métodos para elaborar páginas dinámicas que permitieran que lo mostrado tuviese carácter dinámico (es decir, generado a partir de los datos de la petición), es por ello que nace la gran necesidad de desarrollar páginas web dinámicas**.** (Ferri., 2010)

1.2. Definición de aplicaciones web dinámicas

Los sitios **Web dinámicos** son aquellos que permiten crear aplicaciones dentro de la propia Web, otorgando una mayor interactividad con el navegante, ya que

permite al usuario interactuar con ella debido a que posee contenido dinámico (tipos de letra, colores o fondos, tamaño de pantalla, imágenes, audio, video).

Las páginas dinámicas son páginas HTML(Hipertext Markup language) generadas a partir de lenguajes de programación (scripts) que son ejecutados en el propio servidor web. A diferencia de otros scripts, como el JavaScript, que se ejecutan en el propio navegador del usuario, los 'Server Side' scripts generan un código HTML desde el propio servidor web.

Las páginas web dinámicas utilizan recursos del servidor para generar nuevas hojas de contenido e información para el sitio web. Las páginas web estáticas están realizadas en HTML y están formadas por texto e imágenes con ausencia de movimiento y servicios al usuario, ya que el HTML no permite crear una hoja a partir de bases de datos u otras páginas.

Este código HTML puede ser modificado en función de una petición realizada por el usuario en una Base de Datos. Dependiendo de los resultados de la consulta en la Base de Datos, se generará un código HTML, mostrando diferentes contenidos.

Para crear una página de este tipo no basta con programar en HTML, ya que este lenguaje es muy limitado. Es necesario combinar HTML con otros lenguajes, como JavaScript, VBScript, Java, ASP, PHP, etc.

También puede hacerse uso de capas, de animaciones Flash, de applets java y de hojas de estilo CSS.

A la combinación de estos elementos se le conoce como **DHTML** (DinamicHipertextMarkupLAnguage).

El desarrollo de este tipo de Web son más complicadas pues requieren conocimientos específicos de lenguajes de programación así como creación y gestión de bases de datos, pero la enorme potencia y servicio que otorgan este tipo de

páginas hace que merezca la pena la inversión y esfuerzo invertidos respecto a los resultados obtenidos.

Como ejemplo se puede mencionar a: aplicaciones dinámicas como encuestas y votaciones, foros de soporte, libros de visita, envío de e-mails inteligentes, reserva de productos, pedidos on-line, atención al cliente personalizada.

Una página Web dinámica tiene las siguientes características:

- Gran número de posibilidades en su diseño y desarrollo.
- El visitante puede alterar el diseño, contenidos o presentación de la página a su gusto.
- En su realización se utilizan diversos lenguajes y técnicas de programación.
- El proceso de actualización es sumamente sencillo, sin necesidad de entrar en el servidor.
- Permite un gran número de funcionalidades tales como bases de datos, foros, contenido dinámico, etc.
- Pueden realizarse íntegramente con software de libre distribución.
- Cuenta con un gran número de soluciones prediseñadas de libre disposición.

Muchos diseñadores consideran el HTML un lenguaje de programación, cuando en realidad es un lenguaje definido de construcción de páginas web. Dada esta limitación del HTML ha sido necesario el uso de otros lenguajes de programación que generan tareas como por ejemplo editar los pedidos de diferentes productos de una página web virtual y permiten crear grandes páginas web o portales en internet. (chileunder, 2011)

Ventajas de las páginas web dinámicas frente a las páginas web estáticas

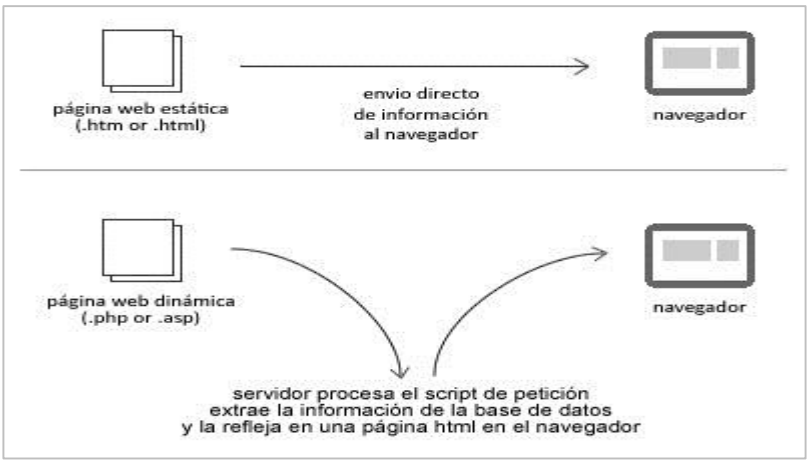

Figura 1. 1. Páginas estáticas vs páginas dinámicas

Las páginas web estáticas son difíciles de editar - para poder modificar el texto de una página web estática habría que descargar la página en HTML del servidor, modificar su contenido y guardarla de nuevo en el servidor web, tarea que sólo podría realizar el web master o la agencia de diseño que desarrolló la página web. Esto significa, que cada vez que necesite actualizar su página, tendrá que contactar con su agencia de diseño web y pagar por cada cambio que se ha realizado en el sitio web.

1. Las páginas web modernas y dinámicas vienen equipadas con un sistema de gestión de contenidos. Este software permite la edición de cada una de las hojas de la página web, es decir, la modificación tanto de texto como imágenes. Así mismo, también permite añadir más pestañas en el menú y submenús manteniendo la página web siempre actualizada.

El contenido de la página modificada es almacenado en la base de datos del servidor y generado en la página en tiempo real una vez requerido por el usuario.

Gracias al sistema de páginas dinámicas, se puede implementar un script que detecta los aspectos más relevantes de la página web, es decir, aquellos aspectos

importantes que el propietario de la página quiera resaltar y generar una lista dinámica y mostrarla en la página web.

A partir de esta definición ha ido evolucionando la forma de crear sitios web dinámicos, empezando por los CGI, APPLETS, SERVLETS, PHP,JSP,ASP, WEB SERVICES,AJAX, los mismos que se mencionan a continuación.

1.3. Técnicas de encriptación

1.3.1. CGI

Interfaz de entrada común (CommonGateWay Interface en inglés) es una importante tecnología de la World Wide Web que permite a un cliente (navegador web) solicitar datos de un programa ejecutado en un servidor web. CGI especifica un estándar para transferir datos entre el cliente y el programa. Es un mecanismo de comunicación entre el servidor web y una aplicación externa. Las aplicaciones que se ejecutan en el servidor reciben el nombre de CGIs.

Las aplicaciones CGI fueron una de las primeras prácticas de crear contenido dinámico para las páginas web. En una aplicación CGI, el servidor web pasa las solicitudes del cliente a un programa externo. Este programa puede estar escrito en cualquier lenguaje que soporte el servidor, aunque por razones de portabilidad se suelen usar lenguajes de script. La salida de dicho programa es enviada al cliente en lugar del archivo estático tradicional.

CGI ha hecho posible la implementación de funciones nuevas y variadas en las páginas web, de tal manera que esta interfaz rápidamente se volvió un estándar, siendo implementada en todo tipo de servidores web.

El funcionamiento de esta tecnología se basa en scripts que residen en el servidor, donde son llamados, ejecutados y regresan información de vuelta al usuario.

A continuación se describe la forma de actuación de un CGI de forma esquemática:

1. En primera instancia, el servidor recibe una petición (el cliente ha activado un URL que contiene el CGI), y comprueba si se trata de una invocación de un CGI.
2. Posteriormente, el servidor prepara el entorno para ejecutar la aplicación. Esta información procede mayoritariamente del cliente.
3. Seguidamente, el servidor ejecuta la aplicación, capturando su salida estándar.
4. A continuación, la aplicación realiza su función: como consecuencia de su actividad se va generando un objeto MIME que la aplicación escribe en su salida estándar.
5. Finalmente, cuando la aplicación finaliza, el servidor envía la información producida, junto con información propia, al cliente, que se encontraba en estado de espera.

Un programa CGI puede ser escrito en cualquier lenguaje de programación que produzca un fichero ejecutable. Entre los lenguajes más habituales se encuentran: C, C++, Perl, Java, Visual Basic... No obstante, debido a que el CGI recibe los parámetros en forma de texto será útil un lenguaje que permita realizar manipulaciones de las cadenas de caracteres de una forma sencilla, como por ejemplo Perl. Perl es un lenguaje interpretado que permite manipulaciones sencillas de ficheros y textos. (García, 2009)

```
// hello.c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
void main(void)
{ (MarcadorDePosición1)
        printf("Content-type: text/html\n\n");
```

```
        printf("<HTML><HEAD><TITLE>Hello World Wide Web</TITLE></HEAD>");
        printf("<BODYBGCOLOR=\"#FFFFFF\"><P ALIGN=CENTER>");
        printf("<H1>Primer CGI</H1>");
        printf("Hello World Wide Web");
        printf("</BODY></HTML>");
}
```

1.3.2. Applet

Un applet es un componente de una aplicación que se ejecuta en el contexto de otro programa, por ejemplo un navegador web. El applet debe ejecutarse en un contenedor, que lo proporciona un programa anfitrión, mediante un plugin, o en aplicaciones como teléfonos móviles que soportan el modelo de programación por 'applets'.

A diferencia de un programa, un applet no puede ejecutarse de manera independiente, ofrece información gráfica y a veces interactúa con el usuario, típicamente carece de sesión y tiene privilegios de seguridad restringidos. Un applet normalmente lleva a cabo una función muy específica que carece de uso independiente.

Ejemplos comunes de applets son las Java applets y las animaciones Flash. Otro ejemplo es el Windows Media Player utilizado para desplegar archivos de video incrustados en los navegadores como el Internet Explorer. Otros plugins permiten mostrar modelos 3D que funcionan con una applet.

Un Java applet es un código JAVA que carece de un método main, por eso se utiliza principalmente para el trabajo de páginas web, ya que es un pequeño programa que es utilizado en una página HTML y representado por una pequeña pantalla gráfica dentro de ésta.

Por otra parte, la diferencia entre una aplicación JAVA y un applet radica en cómo se ejecutan. Para cargar una aplicación JAVA se utiliza el intérprete de JAVA (pcGRASP de Auburn University, Visual J++ de Microsoft, Forte de Sunde Visual Café). En cambio, un applet se puede cargar y ejecutar desde cualquier explorador que soporte JAVA

Las applets son programas que se incluyen en las páginas Web. Las applets son ejecutadas en la máquina cliente, con lo que no existen ralentizaciones por la saturación del módem o del ancho de banda. Permiten cargar a través de la red una aplicación portable que se ejecuta en el navegador. Para que esto ocurra tan sólo hace falta que el navegador sea capaz de interpretar Java. A las páginas que contienen applets se las denomina páginas Java-Powered. Las applets pueden ser visualizadas con la herramienta appletviewer, incluido en el JDK de Java. Las applets (mini aplicación) son programas escritos en Java que sirven para "dar vida" a las páginas Web (interacción en tiempo real, inclusión de animaciones, sonidos...), de ahí su potencia. (García, USAL, 2010)

```
import java.awt.*;
import java.applet.*;
public class AppletDiagonal extends Applet {
  public void paint(Graphics g) {
    g.setColor( Color.red );
   g.drawLine(0, 0, getWidth(), getHeight() );
  }
}
```

1.3.3. Servlets

Los **servlets**, son objetos que corren dentro del contexto de un contenedor de servlets (ej: Tomcat) y extienden su funcionalidad.

La palabra servlet deriva de otra anterior, applet, que se refería a pequeños programas que se ejecutan en el contexto de un navegador web. Por contraposición, un servlet es un programa que se ejecuta en un servidor.

El uso más común de los servlets es generar páginas web de forma dinámica a partir de los parámetros de la petición que envíe el navegador web.

Un servlet es un objeto que se ejecuta en un servidor o contenedor JEE, especialmente diseñado para ofrecer contenido dinámico desde un servidor web, generalmente HTML. Otras opciones que permiten generar contenido dinámico son los lenguajes ASP, PHP, JSP (un caso especial de servlet), Ruby y Python. Forman parte de JEE (Java Enterprise Edition), que es una ampliación de JSE (Java Standard Edition).

Un servlet implementa la interfazjavax.servlet.Servlet o hereda alguna de las clases más convenientes para un protocolo específico (ej: javax.servlet.HttpServlet). Al implementar esta interfaz el servlet es capaz de interpretar los objetos de tipo HttpServletRequest y HttpServletResponse quienes contienen la información de la página que invocó al servlet.

Entre el servidor de aplicaciones (o web content) y el servlet existe un contrato que determina cómo han de interactuar. La especificación de éste se encuentra en los JSR (Java SpecificationRequests) del JCP (Java CommunityProcess).

El ciclo de vida de un Servlet se divide en los siguientes puntos:

1. El cliente solicita una petición a un servidor vía URL.
2. El servidor recibe la petición.
3. Si es la primera, se utiliza el motor de Servlets para cargarlo y se llama al método init().
4. Si ya está iniciado, cualquier petición se convierte en un nuevo hilo. Un Servlet puede manejar múltiples peticiones de clientes.

5. Se llama al método service() para procesar la petición devolviendo el resultado al cliente.
6. Cuando se apaga el motor de un Servlet se llama al método destroy(), que lo destruye y libera los recursos abiertos.
7. Son independientes del servidor utilizado y de su sistema operativo, lo que quiere decir que a pesar
8. de estar escritos en Java, el servidor puede estar escrito en cualquier lenguaje de programación, obteniéndose exactamente el mismo resultado que si lo estuviera en Java.
9. Los servletspueden llamar a otros servlets, e incluso a métodos concretos de otros servlets.
10. Permiten redireccionarpeticiones de servicios a otros servlets(en la misma máquina o en una máquina remota).
11. Los servletspueden obtener fácilmente información acerca del cliente (la permitida por el
12. protocoloHTTP), tal como su dirección IP, el puerto que se utiliza en la llamada, el método
13. utilizado (GET, POST, ...), etc.
14. Permiten además la utilización de cookies y sesiones, de forma que se puede guardar información.

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class ParamServlet extends HttpServlet {
        private static final long serialVersionUID = 1L;
        public void doPost(HttpServletRequest req, HttpServletResponse res)
throws ServletException, IOException {
        // Obtenemos un objeto Print Writer para enviar respuesta
                res.setContentType("text/html");
                PrintWriter pw = res.getWriter();
```

```
                    pw.println("<HTML><HEAD><TITLE>Leyendo
parámetros</TITLE></HEAD>");
                    pw.println("<BODY BGCOLOR=\"#CCBBAA\">");
                    pw.println("<H2>Leyendo parámetros desde un formulario
html</H2><P>");
                    pw.println("<UL>\n");
                    pw.println("Te llamas " + req.getParameter("NOM") + "<BR>");
                    pw.println("y tienes " + req.getParameter("EDA") + "
años<BR>");
                    pw.println("</BODY></HTML>");
                    pw.close();
          }
} (García, USAL, 2010)
```

Sin embargo se generan nuevas propuestas como son PHP,JSP,ASP.

1.3.4. PHP,JSP,ASP

PHP Es un lenguaje de programación utilizado para la creación de sitio web. PHP es un acrónimo recursivo que significa “PHP Hypertext Pre-processor”, (inicialmente se llamó Personal Home Page). Surgió en 1995, desarrollado por PHP Group.

PHP es un lenguaje de script interpretado en el lado del servidor utilizado para la generación de páginas web dinámicas, embebidas en páginas HTML y ejecutadas en el servidor. PHP no necesita ser compilado para ejecutarse. Para su funcionamiento necesita tener instalado Apache o IIS con las librerías de PHP. La mayor parte de su sintaxis ha sido tomada de C, Java y Perl con algunas características específicas. Los archivos cuentan con la extensión (php).

Las características principales son:

- Muy fácil de aprender
- Se caracteriza por ser un lenguaje muy rápido.
- Soporta en cierta medida la orientación a objeto. Clases y herencia.
- Es un lenguaje multiplataforma: Linux, Windows, entre otros.
- Capacidad de conexión con la mayoría de los manejadores de base de datos: MysSQL, PostgreSQL,Oracle, MS SQL Server, entre otras.
- Capacidad de expandir su potencial utilizando módulos.
- Posee documentación en su página oficial la cual incluye descripción y ejemplos de cada una de sus funciones.
- Es libre, por lo que se presenta como una alternativa de fácil acceso para todos.
- Incluye gran cantidad de funciones.
- No requiere definición de tipos de variables ni manejo detallado del bajo nivel.
- Se necesita instalar un servidor web.
- Todo el trabajo lo realiza el servidor y no delega al cliente.
- La legibilidad del código puede verse afectada al mezclar sentencias HTML y PHP.
- La programación orientada a objetos es aún muy deficiente para aplicaciones grandes.
- Dificulta la modularización.
- Dificulta la organización por capas de la aplicación.

```
<?php // Manual de PHP de WebEstilo.com session_register('contador'); echo '<a href="'.$PHP_SELF.'?'.$SID.'">Contador vale: '.++$_SESSION['contador'].'</a>';
```

JSP JSP Es un lenguaje para la creación de sitios web dinámicos, acrónimo de Java Server Pages. Está orientado a desarrollar páginas web en Java. JSP es un lenguaje multiplataforma. Creado para ejecutarse del lado del servidor. JSP fue desarrollado por Sun Microsystems. Comparte ventajas similares a las de ASP.NET, desarrollado para la creación de aplicaciones web potentes. Posee un motor de

páginas basado en los servlets de Java. Para su funcionamiento se necesita tener instalado un servidor Tomcat.

Las principales características son:

- Código separado de la lógica del programa
- Las páginas son compiladas en la primera petición
- Permite separar la parte dinámica de la estática en las páginas web
- Los archivos se encuentran con la extensión jsp.
- El código JSP puede ser incrustado en código HTML
- Código: puede ser incrustado código “”Java
- Directivas: permite controlar parámetros del servlet
- Acciones: permite alterar el flujo normal de la ejecución de una página
- Ejecución rápida de servlets.
- Crear páginas del lado del servidor.
- Multiplataforma
- Integridad con los módulos Java
- Código bien estructurado
- Permite la utilización de servlets.
- Complejidad en el aprendizaje

El código a continuación es un ejemplo.

<**form** action="enviar-array.jsp" method="post">
<**label** for="favoritos"> favoritas</**label**><**br**/>
<**select** multiple="multiple" id="favoritos" name="favoritos" size=9>
<**option** value="deportes">Deportes</**option**>
<**option** value="cine">Cine</**option**>
<**option** value="teatro">Teatro</**option**>
<**option** value="fotografía">Fotografía</**option**>
<**option** value="lectura">Lectura</**option**>
<**option** value="viajes">Viajes</**option**>

<**option** value="pintura">Pintura</**option**>
<**option** value="música">Música</**option**>
<**option** value="otros">Otros</**option**>
</**select**>
<**input** type="submit" value="Enviar"/>
</**form**>
<% **String**[] favoritos = request.getParameterValues("favoritos");%> (Valle, 2003)

ASP Es una tecnología del lado de servidor desarrollada por Microsoft para el desarrollo de sitio web dinámicos.

ASP significa en inglés (Active Server Pages), fue liberado por Microsoft en 1996. Las páginas web desarrolladas bajo este lenguaje es necesario tener instalado Internet Information Server (IIS).

ASP no necesita ser compilado para ejecutarse. Existen varios lenguajes que se pueden utilizar para crear páginas ASP. El más utilizado es VBScript, nativo de Microsoft. ASP se puede hacer también en Perl and Jscript (no JavaScript). El código ASP puede ser insertado junto con el código HTML. Los archivos cuentan con la extensión (asp).

Entre las principales características se destacan:

- Usa Visual Basic Script, siendo fácil para los usuarios.
- Comunicación óptima con SQL Server.
- Soporta el lenguaje JScript (Javascript de Microsoft).
- Código desorganizado.
- Se necesita escribir mucho código para realizar funciones sencillas.
- Tecnología propietaria.

Hospedaje de sitios web costosos. ASP.NET:

- Este es un lenguaje comercializado por Microsoft, y usado por programadores para desarrollar entre otras funciones, sitios web. ASP.NET es el sucesor de la tecnología ASP, fue lanzada al mercado mediante una estrategia de mercado denominada .NET.

El ASP.NET fue desarrollado para resolver las limitantes que brindaba tu antecesor ASP. Creado para desarrollar web sencillas o grandes aplicaciones. Para el desarrollo de ASP.NET se puede utilizar C#, VB.NET o J#. Los archivos cuentan con la extensión (aspx). Para su funcionamiento de las páginas se necesita tener instalado IIS(Internet Information Server) con el Framework .Net. Microsoft Windows 2003 incluye este framework, solo se necesitará instalarlo en versiones anteriores.

```
<html><body>
<% response.write("<h2>TITULO</h2>")
response.write("<p style='color:#0000ff'>PUEDES USAR HTML CON ASP RESPONSE.WRITE</p>")
%></body> </html>
```

1.3.5. Web Services

Un **servicio web** (en inglés, Web service) es un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Distintas aplicaciones de software desarrolladas en lenguajes de programación diferentes, y ejecutadas sobre cualquier plataforma, pueden utilizar los servicios web para intercambiar datos en redes de ordenadores como Internet. La interoperabilidad se consigue mediante la adopción de estándares abiertos. Las organizaciones OASIS y W3C son los comités responsables de la arquitectura y reglamentación de los servicios Web. Para mejorar la interoperabilidad entre distintas implementaciones de servicios Web se ha creado el organismo WS-I, encargado de desarrollar diversos perfiles para definir de manera más exhaustiva estos estándares.

```
Imports Ejemplo1.Clasificacion.BEU
Imports Ejemplo1.Clasificacion.BLL
Imports Ejemplo1..Clasificacion.Contratos
Namespace Ejemplo1.Clasificacion.Servicios
   Public Class Categoria
      Implements ICategoria
      Public Function ConsultarCategoria(ByVal pvoCategoria As BEUCategoria) As
BEUCategoria Implements ICategoria.ConsultarCategoria
   Dim objBLLCategoria As New BLLCategoria()
            Dim objBEUCategoria As New BEUCategoria()
   objBEUCategoria = bjBLLCategoria.ConsultarCategoria(pvoCategoria)
            BLLImpersonation.undoImpersonation()
            Return objBEUCategoria
         Catch ex As Exception
            Return Nothing
         End Try
      End Function
```

Las características se muestran a continuación:

- Aportan interoperabilidad entre aplicaciones de software independientemente de sus propiedades o de las plataformas sobre las que se instalen.
- Los servicios Web fomentan los estándares y protocolos basados en texto, que hacen más fácil acceder a su contenido y entender su funcionamiento.
- Permiten que servicios y software de diferentes compañías ubicadas en diferentes lugares geográficos puedan ser combinados fácilmente para proveer servicios integrados. (García, USAL, 2010)
- Para realizar transacciones no pueden compararse en su grado de desarrollo con los estándares abiertos de computación distribuida como CORBA (CommonObjectRequestBrokerArchitecture).
- Su rendimiento es bajo si se compara con otros modelos de computación distribuida, tales como RMI (RemoteMethodInvocation), CORBA o DCOM

(DistributedComponentObjectModel). Es uno de los inconvenientes derivados de adoptar un formato basado en texto. Y es que entre los objetivos de XML no se encuentra la concisión ni la eficacia de procesamiento.

- Al apoyarse en HTTP, pueden esquivar medidas de seguridad basadas en firewall cuyas reglas tratan de bloquear o auditar la comunicación entre programas a ambos lados de la barrera (Piñol., 2012)
- Debido a sus inconvenientes aparecen una nueva técnica de desarrollo que incluye varias tecnologías como lo es AJAX.

1.3.6. AJAX

Ajax, acrónimo de **A**synchronous**J**avaScript **A**nd **X**ML (JavaScript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (Rich Internet Applications). Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, lo que significa aumentar la interactividad, velocidad y usabilidad en las aplicaciones.

Ajax es una tecnología asíncrona, en el sentido de que los datos adicionales se requieren al servidor y se cargan en segundo plano sin interferir con la visualización ni el comportamiento de la página. JavaScript es el lenguaje interpretado (scripting language) en el que normalmente se efectúan las funciones de llamada de Ajax mientras que el acceso a los datos se realiza mediante XMLHttpRequest, objeto disponible en los navegadores actuales. En cualquier caso, no es necesario que el contenido asíncrono esté formateado en XML.

Ajax es una técnica válida para múltiples plataformas y utilizable en muchos sistemas operativos y navegadores, dado a que está basado en estándares abiertos como JavaScript y DocumentObjectModel (DOM).

AJAX, en resumen, es: Cargar y renderizar una página, luego mantenerse en esa página mientras scripts y rutinas van al servidor buscando, en background, los datos

que son usados para actualizar la página solo re-renderizando la página y mostrando u ocultando porciones de la misma.

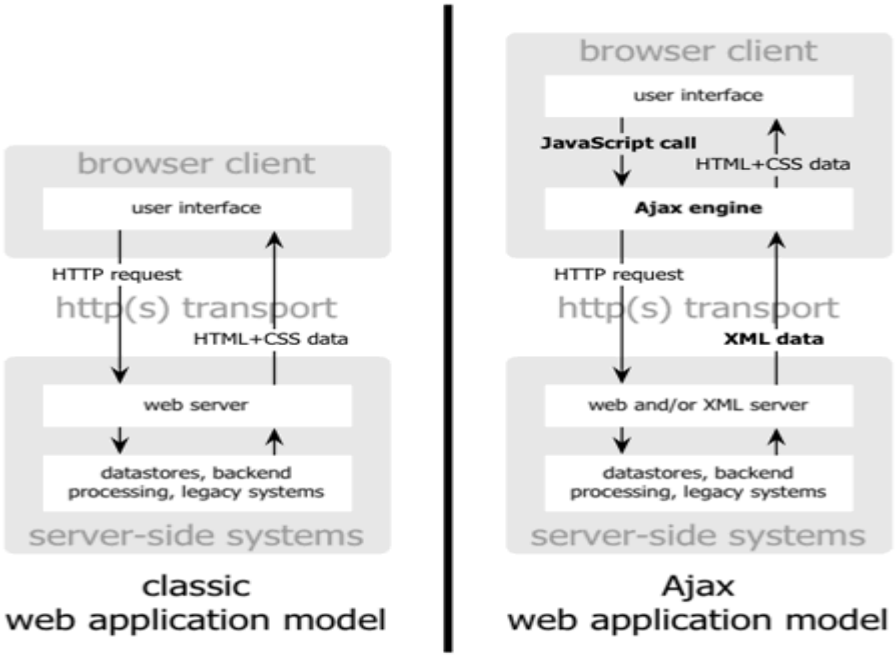

Figura 1. 2. Comparación gráfica del modelo tradicional y AJAX

```
<html>
<head>
<title>Ejemplo1</title>
<script language = "javascript"> var XMLHttpRequestObject = false;
    if (window.XMLHttpRequest) { XMLHttpRequestObject = new XMLHttpRequest(); }
    else if (window.ActiveXObject) {
     XMLHttpRequestObject = new ActiveXObject("Microsoft.XMLHTTP"); }
      function pedirDatos(fuenteDatos, divID){
       if(XMLHttpRequestObject) {
       var obj = document.getElementById(divID);
      XMLHttpRequestObject.open("GET", fuenteDatos);
       XMLHttpRequestObject.onreadystatechange = function(){
       if (XMLHttpRequestObject.readyState == 4 && XMLHttpRequestObject.status == 200) {
       obj.innerHTML = XMLHttpRequestObject.responseText;
```

```
        }
       }
       XMLHttpRequestObject.send(null);
      }
     }
  </script>
  </head>
  <body>
    <H1>Mostrando datos con AJAX</H1>
    <form>  <input type = "button" value = "Mostrar mensaje" onclick =
"pedirDatos('datos.txt','targetDiv')">
    </form>
    <div id="targetDiv" style="background-color:#99FF66;">
    <p>Aqu&iacute; aparecer&aacute; texto</p>
    </div>  </body>  </html>
```

Tecnologías incluidas en Ajax

Ajax es una combinación de cuatro tecnologías ya existentes:

- XHTML (o HTML) y hojas de estilos en cascada (CSS) para el diseño que acompaña a la información.
- DocumentObjectModel (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. El objeto XMLHttpRequest para intercambiar datos de forma asíncrona 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.
- XML es el formato usado generalmente para la transferencia de datos solicitados al servidor, aunque cualquier formato puede funcionar, incluyendo HTML preformateado, texto plano, JSON y hasta EBML.

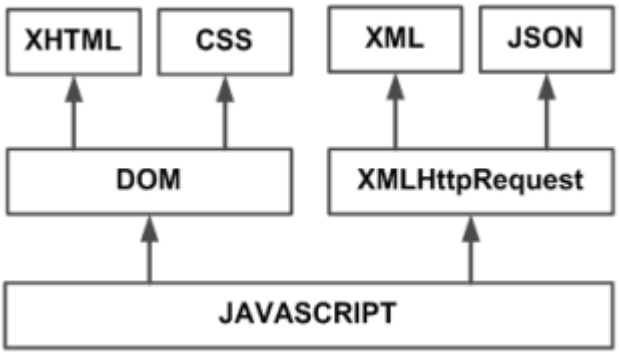

Figura 1. 3. Tecnologías bajo el concepto de AJAX

Fuente: (Eguiluz, 2010)

CAPÍTULO II

2. AJAX NUEVO PARADIGMA EN APLICACIONES WEB DINÁMICAS

2.1. Introducción

Con la rápida evolución de aplicaciones web dinámicas y la creciente necesidad de los usuarios de obtener información mediante la web en forma ágil y oportuna sin tener que quedarse en espera durante mucho tiempo, se ha considerado a AJAX como un paradigma nuevo que permite que el tiempo de espera en línea sea menor, debido a que ya no se recarga toda la página cada vez que se envía una petición al servidor (como por ejemplo una validación de datos, edición de datos en la memoria e incluso algo de navegación) sino por el contrario únicamente la información necesaria. Esto ha hecho que AJAX sea una nueva forma de trabajo la misma que engloba diferentes tecnologías web ya existentes como, lenguaje de marcas de hipertexto (HTML), hojas de estilo en cascada (CSS) y javascript las mismas que fusionadas permiten hacer un trabajo más cercano a la funcionalidad de las aplicaciones de escritorio.

2.2. Definición de AJAX

Ajax, acrónimo de Asynchronous JavaScript y XML (JavaScript asíncrono y XML), es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (Rich Internet Applications). Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas o sea sin tener que actualizar toda la página completa, lo que significa aumentar la interactividad, velocidad y usabilidad en las aplicaciones.

Ajax es una tecnología asíncrona, en el sentido de que los datos adicionales se requieren al servidor y se cargan en segundo plano sin interferir con la visualización

ni el comportamiento de la página. JavaScript es el lenguaje interpretado (scripting language) en el que normalmente se efectúan las funciones de llamada de Ajax mientras que el acceso a los datos se realiza mediante XMLHttpRequest, objeto disponible en los navegadores actuales. En cualquier caso, no es necesario que el contenido asíncrono esté formateado en XML.

Ajax es una técnica válida para múltiples plataformas y utilizable en muchos sistemas operativos y navegadores, dado que está basado en estándares abiertos como JavaScript y Document Object Model (DOM). (Blogs, 2010)

2.3. Las tecnologías que forman AJAX

Desarrollar aplicaciones AJAX requiere un conocimiento avanzado de todas y cada una de las tecnologías anteriores.

2.3.1. HTML Y CSS

HTML, siglas de HyperText Markup Language (Lenguaje de Marcado de Hipertexto), es el lenguaje de marcado predominante para la elaboración de páginas web. Es usado para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes. HTML también puede describir, hasta un cierto punto, la apariencia de un documento, y puede incluir un script (por ejemplo Javascript), el cual puede afectar el comportamiento de navegadores web y otros procesadores de HTML.

HTML también es usado para referirse al contenido del tipo de MIME (extensiones multipropósito de correo de internet) o todavía más ampliamente como un término genérico para el HTML, ya sea en forma descendida del XML (o en forma descendida directamente de SGML (Estándar de Lenguaje de Marcado Generalizado). (W3Schools, 2012)

HTML (HyperText Markup Language) es un lenguaje muy sencillo que permite describir hipertexto, es decir, texto presentado de forma estructurada y agradable, con enlaces (hyperlinks) que conducen a otros documentos o fuentes de información relacionadas, y con inserciones multimedia (gráficos, sonido...) La descripción se basa en especificar en el texto la estructura lógica del contenido (títulos, párrafos de texto normal, enumeraciones, definiciones, citas, etc) así como los diferentes efectos que se quieren dar (especificar los lugares del documento donde se debe poner cursiva, negrita, o un gráfico determinado). (Universidad Politécnica de Madrid, España, 2005)

CSS Las **hojas de estilo en cascada** (en inglés Cascading Style Sheets), CSS es un lenguaje usado para definir la presentación de un documento estructurado escrito en HTML o XML (y por extensión en XHTML). El W3C (World Wide Web Consortium) es el encargado de formular la especificación de las hojas de estilo que servirán de estándar para los agentes de usuario o navegadores.

La idea que se encuentra detrás del desarrollo de CSS es separar la estructura de un documento de su presentación**.** (HTMLQUICK, 2009)

Las hojas de estilo vienen a intentar volver a separar en un documento el estilo lógico del estilo físico, dejando este último en bloques de definición de estilos separados de la estructura del documento.

CSS se trata de una especificación sobre los estilos físicos aplicables a un documento HTML, y trata de dar la separación definitiva de la lógica (estructura) y el físico (presentación) del documento.

a) Características y ventajas de las CSS

El modo de funcionamiento de las CSS consiste en definir, mediante una sintaxis especial la forma de presentación que le aplicaremos a un sitio web entero, de modo que se puede definir la forma de todo el web de una sola vez.

Un documento HTML o página se puede definir la forma, en un pequeño trozo de código en la cabecera a toda la página. Una porción del documento aplicando estilos visibles en un trozo de la página. Una etiqueta en concreto, llegando incluso a definir varios estilos diferentes para una sola etiqueta. Podemos definir por ejemplo varios tipos de párrafos: en rojo, en azul, con márgenes o sin ellos.

La potencia de la tecnología salta a la vista. Pero no solo se queda aquí, ya que además esta sintaxis CSS permite aplicar al documento formato de modo mucho más exacto. Si antes el HTML se nos quedaba corto para maquetar las páginas y teníamos que utilizar trucos para conseguir nuestros efectos, ahora tenemos muchas más herramientas que nos permiten definir esta forma: Es posible definir la distancia entre líneas del documento. Se puede aplicar identado a las primeras líneas del párrafo.

Es posible colocar elementos en la página con mayor precisión, y sin lugar a errores.

Y mucho más, como definir la visibilidad de los elementos, márgenes, subrayados, tachados.

b) Tres tipos de estilos

La información CSS se puede proporcionar por varias fuentes, ya sea adjunto como un documento por separado o incorporado en el documento HTML, y dentro de estas posibilidades destacan tres formas de dar estilo a un documento web:

c) Hoja de Estilo Externa

La Hoja de Estilo Externa se almacena en un archivo diferente al del archivo con el código HTML el cual está vinculado a través del elemento link que debe ir situado en la sección head. Es la manera de programar más eficiente ya que separa

completamente las reglas de formato para la página HTML de la estructura básica de la página.

d) Hoja de Estilo Interna

La Hoja de Estilo Interna está incorporada a un documento HTML, a través del elemento <<style>> dentro de la sección <<head>>, consiguiendo de esta manera separar la información del estilo del código HTML.

e) Estilo en Línea

El Estilo en Línea sirve para insertar el lenguaje de estilo directamente dentro de la sección <<body>> con el elemento <<style>>. Sin embargo, este tipo de estilo no se recomienda pues se debe intentar siempre separar el contenido de la presentación.

f) Versiones CSS

Existen varias versiones: CSS1 y CSS2, la CSS3 está todavía en desarrollo por el CSS WG (Cascading Style Sheets Working Group).

g) Ventajas de CSS

a) La principal ventaja es que el estilo se puede guardar completamente por separado del contenido siendo posible por ejemplo, almacenar todos los estilos de presentación para una web de 10.000 páginas en un sólo archivo de CSS.
b) CSS permite un mejor control en la presentación de un sitio web que los elementos de HTML, agilizando su actualización.
c) Aumento de la accesibilidad de los usuarios gracias a que pueden especificar su propia hoja de estilo, permitiéndoles modificar el formato de un sitio web según sus necesidades.
d) El ahorro global en el ancho de banda es notable, ya que la hoja de estilo se almacena en cache después de la primera solicitud y se puede volver a usar para cada página del sitio, no se tiene que descargar con cada página web.

e) Una página puede tener diferentes hojas de estilo para mostrarse en diferentes dispositivos, como pueden ser impresoras, lectores de voz, o móviles

```
<p>
Esto es un párrafo en varias palabras <SPAN style="color:green">en color verde</SPAN>.
resulta muy fácil.
</p>
<p style="color:#990000">
Esto es un párrafo de color rojo.
</p>
<p style="color:#000099">
Esto es un párrafo de color azul.
</p>
<html> <head>
<title>Ejemplo de estilos para toda una página</title>
<STYLE type="text/css">
<!--
H1 {text-decoration: underline; text-align:center}
P {font-Family:arial,verdana; color: white; background-color: black}
BODY {color:black;background-color: #cccccc; text-indent:1cm}
</STYLE>
</head>
<body>
<h1>Página con estilos</h1>
Bienvenidos...
<p>Siento ser tan hortera, pero esto es un ejemplo sin m&aacute;s importancia</p>
</body>
</html>
```

(WXVC, 2011)

2.3.2. DOM Y JAVASCRIPT

DOM Acrónimo de Document Object Model (Modelo de Objetos de Documento). Es una plataforma que proporciona un conjunto estándar de objetos a través de la cual se pueden crear documentos HTML y XML, navegar por su estructura y, modificar, añadir y borrar tanto elementos como contenidos. Al no apoyarse en un lenguaje de programación en particular, DOM facilita el diseño de páginas web activas, proporcionando una interfaz estándar para que otro software manipule los documentos**.** (w3c, 2010)

El DOM es una jerarquía de objetos predefinidos que describen los elementos de la página web que está mostrando el navegador, así como otras características del proceso de navegación (como son el historial, el tamaño de la ventana de navegación o el contenido de la barra de estado del navegador). Un objeto es en el fondo, un conjunto de variables y funciones que actúa sobre dichas variables, encapsuladas en un mismo paquete.

El acceso a las funciones y a las variables se realiza mediante una interfaz bien definida que aleja al programador de la necesidad de conocer cómo están implementadas internamente dichas funciones.

De este modo, la programación orientada a objetos resulta muy intuitiva, y más próxima al conocimiento humano.

DOM define qué atributos son asociados con cada objeto y cómo los objetos y los atributos pueden ser manipulados. El HTML dinámico (DHTML) se basa en el DOM para cambiar dinámicamente la apariencia de las páginas web después de que han sido descargadas por un navegador.

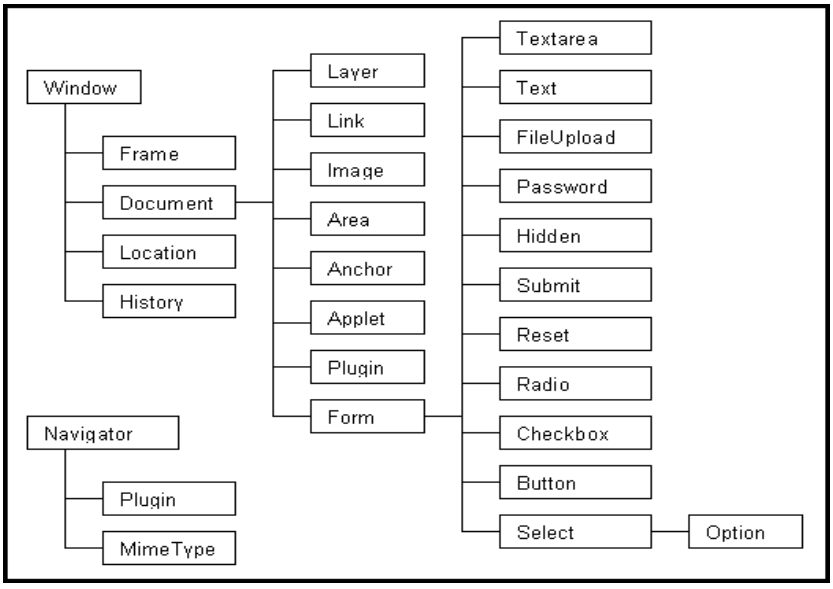

Figura 2. 1. DOM según Netscape

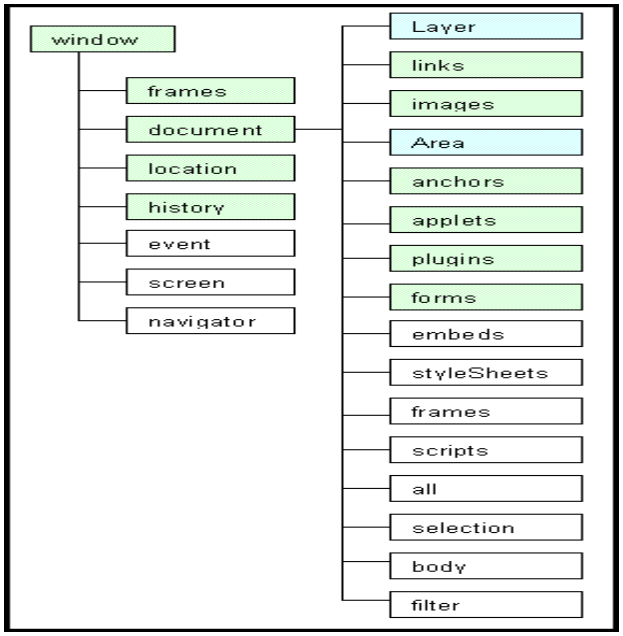

Figura 2. 2. DOM según Microsoft

Fuente:(ElCodigo, 2011)

JavaScript es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos[] , basado en prototipos, imperativo, débilmente tipado y dinámico.

Se utiliza principalmente en su forma del lado del cliente (client-side), implementado como parte de un navegador web permitiendo mejoras en la interfaz de usuario y páginas web dinámicas, aunque existe una forma de JavaScript del lado del servidor (Server-side JavaScript o SSJS). Su uso en aplicaciones externas a la web, por ejemplo en documentos PDF, aplicaciones de escritorio (mayoritariamente widgets) es también significativo.

JavaScript se diseñó con una sintaxis similar al C, aunque adopta nombres y convenciones del lenguaje de programación Java. Sin embargo Java y JavaScript no están relacionados y tienen semánticas y propósitos diferentes. Todos los navegadores modernos interpretan el código JavaScript integrado en las páginas web. Para interactuar con una página web se provee al lenguaje JavaScript de una implementación del Document Object Model (DOM). (W3Schools, 2010)

JavaScript no es un lenguaje de programación propiamente dicho. Es un lenguaje script u orientado a documento, como pueden ser los lenguajes de macros que tienen muchos procesadores de texto.

JavaScript y Java son dos cosas distintas. Principalmente porque Java es un lenguaje de programación completo. Lo único que comparten es la sintaxis. Un lenguaje de comandos multiplataforma del WWW desarrollado por Netscape Communications es el código de JavaScript, este se inserta directamente en una página HTML.

a) Características de JavaScript

JavaScript comparte muchos elementos con otros lenguajes de alto nivel. Hay que tener en cuenta que este lenguaje es muy semejante a otros como C, Java o PHP, tanto en su formato como en su sintaxis, aunque por supuesto tiene sus propias características definitorias.

JavaScript es un lenguaje que diferencia entre mayúsculas y minúsculas, por lo que si escribimos alguna expresión en minúsculas, deberemos mantener esa expresión en minúsculas a lo largo de todo el programa.

Si se escribe esa misma expresión en mayúsculas, será una expresión diferente a la primera. Esto es así en la mayoría de los lenguajes de este tipo, como PHP.

Otra característica es que podemos encerrar las expresiones que escribamos con una serie de caracteres especiales. Estos caracteres se denominan operadores y sirven tanto para encerrar expresiones como para realizar trabajos con ellas, como operaciones matemáticas o de texto. Los operadores que permiten encerrar expresiones deben abrirse siempre. '(', '{' y '[' y deben cerrarse con sus correspondientes ')', '}' y ']', respectivamente.

Como JavaScript es un lenguaje de formato libre, podemos escribir las líneas de código de la forma que consideremos mejor, aunque por supuesto debemos escribir siempre de la forma correcta. Por ejemplo, podemos escribir las líneas con un número variable de espacios:

```
variable = "hola";
variable="hola";
   variable ="hola";
variable= "hola"   ;
```

Esto significa que podemos añadir tabuladores al inicio de la línea para justificar los párrafos de código. También podemos romper las líneas de texto si son demasiado largas:

```
document.write("Muy \    buenas");
```

Pero no podemos hacer esto:

```
document.write \    ("Muy buenas");
```

Sólo podemos romper cadenas de texto, no instrucciones.

Otro aspecto importante de JavaScript es la necesidad o no de utilizar el punto y coma ';' al final de las instrucciones. Este operador sólo sirve para delimitar las instrucciones, pero aunque su uso es obligatorio en la mayoría de los lenguajes, es opcional en JavaScript. Si usamos este operador, podemos incluir varias sentencias en la misma línea de código, y si no lo usamos, sólo podemos escribir una sentencia cada vez. De todas formas, aconsejamos usar el punto y coma porque en otros lenguajes como PHP o Java, este operador es obligatorio.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Ejercicio 2 - Mostrar mensajes complejos</title>
 <script type="text/javascript">
  var mensaje = "Hola Mundo! \n Qué facil es incluir \'comillas simples\' \n y \"comillas dobles\" ";
  alert(mensaje);
</script> </head> <body>
<p>Esta página muestra un mensaje complejo</p>
</body> </html>
```

2.3.3. XML

XML, siglas en inglés de e**X**tensible **M**arkup **L**anguage ('lenguaje de marcas extensible'), es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C). Es una simplificación y adaptación del SGML y permite definir la gramática de lenguajes específicos (de la misma manera que HTML es a su vez un lenguaje definido por SGML). Algunos de estos lenguajes que usan XML para su definición son XHTML, SVG, MathML. XML no ha nacido sólo para su aplicación en Internet, sino que se propone como un estándar para el

intercambio de información estructurada entre diferentes plataformas. Se puede usar en bases de datos, editores de texto, hojas de cálculo.

XML es una tecnología sencilla que tiene a su alrededor otras que la complementan y la hacen mucho más grande y con unas posibilidades mucho mayores. Tiene un papel muy importante en la actualidad ya que permite la compatibilidad entre sistemas para compartir la información de una manera segura, fiable y fácil**.** (Quin, 2013)

En la práctica corresponde a un estándar que permite a diferentes aplicaciones interactuar con facilidad a través de la red. XML fue creado al amparo del World Wide Web Consortium (W3C) organismo que vela por el desarrollo de WWW partiendo de las amplias especificaciones de SGML.

XML es un formato basado en texto, específicamente diseñado para almacenar y transmitir datos. Un documento XML se compone de elementos XML, cada uno de los cuales consta de una etiqueta de inicio, de una etiqueta de fin y de los datos comprendidos entre ambas etiquetas. Al igual que los documentos HTML, un documento XML contiene texto anotado por etiquetas. Sin embargo, a diferencia de HTML, XML admite un conjunto ilimitado de etiquetas, no para indicar el aspecto que debe tener algo, sino lo que significa.

La particularidad más importante del XML es que no posee etiquetas prefijadas con anterioridad, ya que es el propio diseñador el que las crea a su antojo, dependiendo del contenido del documento.

De esta forma, los documentos XML con información sobre libros deberían tener etiquetas como <AUTOR>, <EDITORIAL>, <Nº_DE_PÁGINAS>, <PRECIO>, etc., mientras que los documentos XML relacionados con educación incluyen etiquetas del tipo de <ASIGNATURA>, <ALUMNO>, <CURSO>, <NOTA>, etc.

Por ejemplo en la siguiente tabla se muestra la información incluida por un código típico HTML y su versión equivalente en XML. Se puede apreciar en este

ejemplo, que es mucho más fácil de entender la representación en XML. (Ramon, 2011) (Microsoft, 2011)

Tabla 2. 1.

Comparación entre HTML y XML

HTML	XML
<TABLE>	<LIBROS>
<TR>	<LIBRO>
<TD>Título</TD>	<TITULO>AutoSketch</TITULO>
<TD>Autor</TD>	<AUTOR>Ramón Montero</AUTOR>
<TD>Precio</TD>	<PRECIO>33</PRECIO>
	</LIBRO>
</TR>	<LIBRO>
<TR>	<TITULO>Windows 98</TITULO>
<TD>AutoSketch</TD>	<AUTOR>Jaime Perez</AUTOR>
<TD>Ramón Montero</TD>	<PRECIO>3.250</PRECIO>
<TD>33</TD> </TR>	</LIBRO>
<TR>	<LIBRO>
<TD>Windows 98</TD>	<TITULO>Web Graphics</TITULO>
<TD>Jaime Perez</TD>	<AUTOR>Ron Wodaski</AUTOR>
<TD>3.250</TD></TR>	<PRECIO>8.975</PRECIO>
<TR>	</LIBRO>
<TD>Web Graphics</TD>	</LIBROS>
<TD>Ron Wodaski</TD>	
<TD>8.975</TD> </TR> </TABLE>	

a) **Características Principales**

b) **Extensible**

Dentro de XML se pueden definir un conjunto ilimitado de etiquetas. Mientras que las etiquetas de HTML pueden utilizarse para desplegar una palabra en negrita o itálicas, el XML proporciona un marco de trabajo para etiquetado de datos estructurados. Un elemento de XML puede declarar que sus datos asociados sean el precio de venta al público, un impuesto de venta, el título de un libro o cualquier otro elemento de datos deseado. Al irse adoptando las etiquetas XML a lo largo de una intranet de alguna organización y a lo ancho de la Internet, habrá una correspondiente habilidad para buscar y manipular datos sin importar las aplicaciones dentro de las cuales se encuentre.

c) Representación estructural de los datos.

El XML proporciona una representación estructural de los datos que ha probado ser ampliamente implementable y fácil de distribuir. Las implementaciones industriales en la comunidad del SGML y en otros lugares han demostrado que la calidad intrínseca y la fortaleza industrial del formato de datos con estructura de árbol del XML. El XML es un subconjunto del SGML que está optimizado para su transmisión por Web; al estar definido por el Consorcio de la World Wide Web, asegura que los datos estructurados serán uniformes e independientes de aplicaciones o compañías. Esta interoperabilidad resultante está dando el impuso de inicio a una nueva generación de aplicaciones de Web para comercio electrónico [MSDN en línea. Introducción al XML].

El lenguaje XML proporciona un estándar de datos que puede codificar el contenido, la semántica y el esquema de una amplia variedad de casos que van desde simples a complejos, por ejemplo XML puede ser utilizado para marcar lo siguiente:

a) Un documento ordinario.
b) Un registro estructurado, tal como un registro de citas u órdenes de compra.
c) Un registro de datos, tal como el resultado de una consulta.
d) Metacontenido acerca de un sitio Web, tal como un Formato de Definición de Canal (Channel Definition Format, CDF).
e) Presentaciones gráficas, tales como la interface de usuario de una aplicación.

Una vez que los datos estén en el escritorio del cliente, pueden ser manipulados, editados, y presentados de una gran variedad de maneras, sin viajes de regreso al servidor. Los servidores se pueden convertir ahora en más escalables, debido a las menores cargas de cálculo y ancho de banda. Además, dado que los datos son intercambiados en el formato XML, pueden ser fácilmente mezclados desde diferentes fuentes.

d) Los datos son separados de la presentación y el proceso.

El poder y la belleza del XML es que mantiene la separación entre la interface de usuario y los datos estructurados. El XML separa los datos de la presentación y el proceso, permitiendo desplegar y procesar los datos tal como usted desee, al aplicar diferentes hojas de estilo y aplicaciones.

Esta separación de datos de la presentación permite una integración de datos perfecta de fuentes diversas. La información de clientes, órdenes de compra, resultados de investigaciones, pagos de facturas, registros médicos, datos de catálogo y cualquier otra información se puede convertir a XML, permitiendo a los datos ser intercambiados en línea tan fácilmente como las páginas de HTML despliegan datos hoy. Los datos codificados en XML pueden ser transmitidos sobre la Web hasta el escritorio. No es necesario retroajustar información en formatos propietarios almacenados en bases de datos o documentos de mainframes y, debido a que se usa el HTTP para transmitir documentos XML sobre la red, no se necesitan cambios para esta función.

e) Conversión de los datos XML en autodescriptivos.

Los datos codificados en XML son autodescriptivos, pues las etiquetas descriptivas están entremezcladas con los datos. El formato abierto y flexible utilizado por XML permite su uso en cualquier lugar donde sea necesario intercambiar y transferir información. Dado que el XML es independiente del HTML, se puede insertar código XML en documentos HTML. El W3C ha definido un formato mediante el cual se pueden encapsular en páginas HTML los datos basados en XML. Al incrustar datos XML en una página HTML, se pueden generar varias vistas a partir de los datos entregados, utilizando los datos semánticos que contiene el XML**.** (Monografías, 2012)

f) Características de XML

a) XML impone una sintaxis más rígida para las marcas, que permite su proceso de forma más eficiente.
b) En XML, las marcas de término no pueden ser omitidas (a diferencia de la marca P en HTML, por ejemplo).
c) Marcas sin contenido, como IMG o BR en HTML, terminan con un /> para indicar que allí acaban. XML también distingue entre minúsculas y mayúsculas.
d) También, cualquier valor de un atributo en una marca debe ir entre comillas (es decir, no se pueden omitir). Esto significa que interpretar XML sin conocer el conjunto válido de marcas es mucho más sencillo. En particular, definir entonces el tipo del documento (lo que en SGML y XML se llama.
e) DTD de document type declaration) no es obligatorio. En este caso, las marcas se obtienen a medida que se interpreta el documento.

XML permite definir lenguajes de marcas para cualquier fin y tiene capacidades de validación de datos.

g) Usos de XML

Actualmente XML está siendo usado para muchos fines diferentes. La lista a continuación son sólo algunos de los más importantes:

- XSL: el eXtensible Style sheet Languaje es la contraparte de XML a CSS (Cascading Style Sheets), el lenguaje de estilo que permite separar la visualización del documento y su estructura semántica.
- XLL: el eXtensible Link Language permite definir distintos tipos de enlaces (links) entre documentos, ya sean externos o internos.
- RDF: el Resource Description Framework es un estándar de la Web para proveer interoperabilidad entre aplicaciones y descripciones de metadatos.

- MathML: dos conjuntos de marcas definidas usando XML para presentar fórmulas y expresar la semántica de expresiones matemáticas.
- SMIL: el Synchronized Multimedia Integration Language es un lenguaje para sincronizar presentaciones multimediales en la Web, donde la posición y tiempo de activación de distintos objetos puede ser especificada. (Chile, 2012)

2.4. Utilización de ajax en .Net Framework

ASP.NET AJAX, anteriormente llamado Atlas, es un conjunto de extensiones para ASP.NET desarrollado por Microsoft para implementar la funcionalidad de Ajax.

Mediante componentes del lado del cliente y del servidor, ASP.NET AJAX permite al desarrollador crear aplicaciones web en ASP.NET 2.0 que pueden actualizar datos en la página web sin un recarga completa de la misma. La tecnología clave que permite esta funcionalidad es el objeto XMLHttpRequest, junto con Javascript y DHTML.

ASP.NET AJAX fue liberado en enero de 2007 después de un largo periodo de pruebas. Fue subsecuentemente incluido con la versión 3.5 del .NET Framework, que fue liberada junto con Visual Studio 2008 en noviembre de 2007.

El 11 de septiembre de 2006, Scott Guthrie, el director general a cargo de la plataforma.NET, anuncio que ATLAS seria renombrado[] y lanzado como tres productos a finales del año.

Los nuevos productos son llamados Microsoft AJAX Library, que contiene las bibliotecas javascript, ASP.NET 2.0 AJAX Extensions, que contiene el código.NET del lado del servidor, y ASP.NET AJAX Control Toolkit, que incluye controles de código compartido que pueden ser utilizados con ASP.NET AJAX.

Microsoft AJAX Library es una colección autónoma de clases en JavaScript estandarizadas incluidas con ASP.NET AJAX. Es admitida por la mayoría de los navegadores más populares y puede ser usada para construir aplicaciones web centradas en el cliente que integradas con un proveedor de datos.

La plataforma .Net AJAX presenta una plataforma de trabajo extraordinariamente fácil de usar, la misma que simplifica el trabajo de los desarrolladores de AJAX, permitiendo pasar más tiempo en detalles de implementación y menos tiempo en código XML.

El Framework de AJAX.Net utiliza JSON(Javascript Object Notation)

Ajax agrupa estas tecnologías y provee un receso entre el modelo tradicional que requiere un retroceso al servidor en orden de ejecución de los métodos del lado del servidor. Ajax permite al navegador hacer llamadas al código del lado del servidor y recibir una respuesta del servidor todo esto sin usar el retroceso de ASP.Net, lo cual transfiere muchas páginas de información con cada requerimiento. (Giardina, 2011)

2.5. Trabajo con Ajax.Net Framework

El framework de Ajax.Net es una nueva forma de trabajo para la interacción entre el servidor y el navegador. La principal diferencia entre Ajax.net es que hay una capa de trabajo alrededor de las tecnologías para crear el método Ajax. Sin la capa de trabajo podría codificar manualmente la comunicación Cliente/servidor, lo cual generaría muchos errores y tiempo consumido.

El framework de Ajax.Net dinámicamente crea un objeto proxy que actúa como un mecanismo de comunicación entre el servidor y el navegador. Cuando un desarrollador crea un método detrás del código, éste método puede ser arreglado con un atributo que diga al motor Ajax.Net lo que necesita para crear el proxy Javascript.

El proxy Javascript podría tener una interface similar al los objetos que existen en el servidor. Utilizando el framework Ajax.Net los objetos del lado del cliente

podrían tener las mismas propiedades como contraparte del lado del servidor. El proxy es simplemente un objeto de transferencia de datos y puede no implementar la lógica del lado del servidor. (DEVX, 2006)

2.6. Diferenciación entre aplicaciones web tradicionales y AJAX

En las aplicaciones web tradicionales, las acciones del usuario en la página (pinchar en un botón, seleccionar un valor de una lista, etc.) desencadenan llamadas al servidor. Una vez procesada la petición del usuario, el servidor devuelve una nueva página HTML al navegador del usuario.

Esta técnica tradicional para crear aplicaciones web hace que el usuario tenga que esperar a que se recargue la página con los cambios solicitados. Si la aplicación debe realizar peticiones continuas, su uso se convierte en algo molesto AJAX permite mejorar completamente la interacción del usuario con la aplicación, evitando las recargas constantes de la página, ya que el intercambio de información con el servidor se produce en un segundo plano.

Las aplicaciones construidas con AJAX eliminan la recarga constante de páginas mediante la creación de un elemento intermedio entre el usuario y el servidor. La nueva capa intermedia de AJAX mejora la respuesta de la aplicación, ya que el usuario nunca se encuentra con una ventana del navegador vacía esperando la respuesta del servidor.

El siguiente esquema muestra la diferencia más importante entre una aplicación web tradicional y una aplicación web creada con AJAX. La imagen superior muestra la interación síncrona propia de las aplicaciones web tradicionales. La imagen inferior muestra la comunicación asíncrona de las aplicaciones creadas con AJAX.

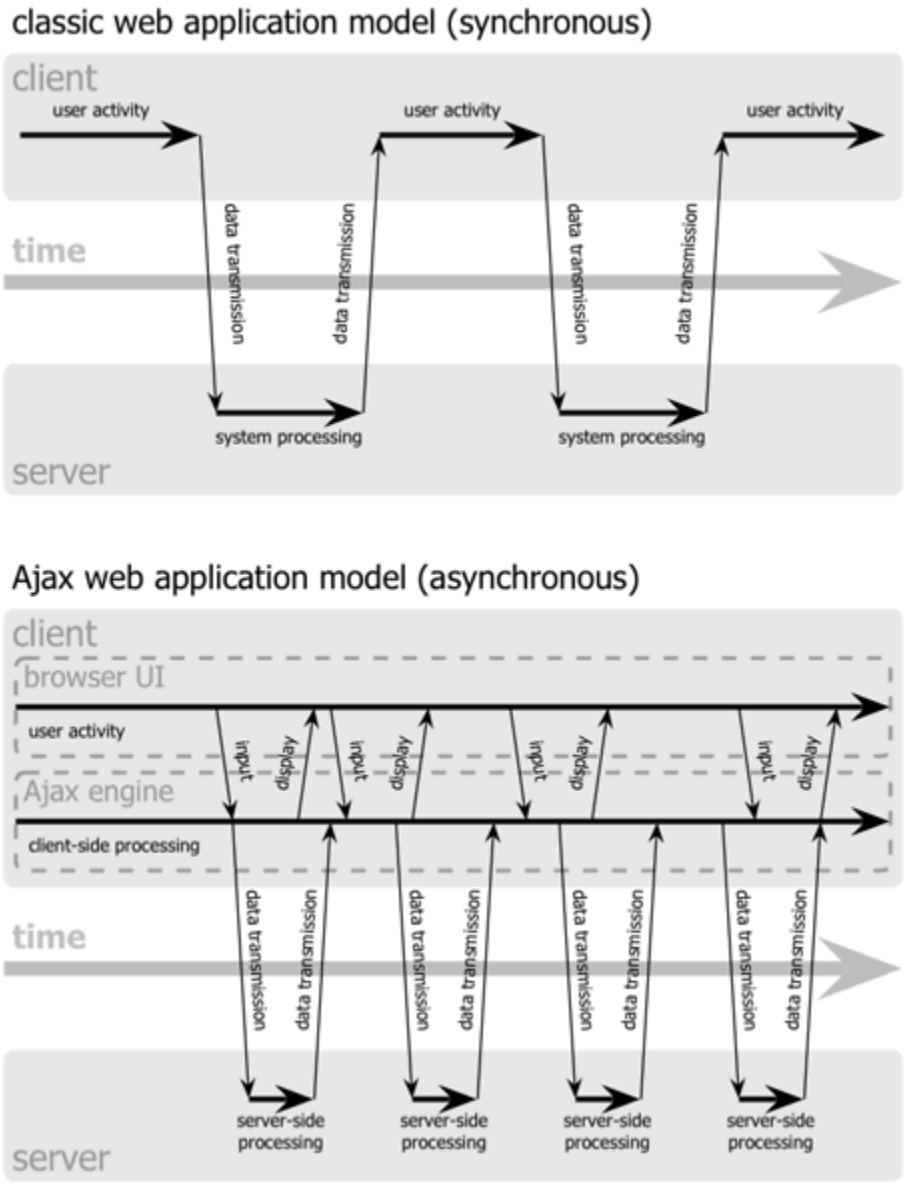

Figura 2. 3. Comparación entre las comunicaciones síncronas de las aplicaciones web tradicionales y las comunicaciones asíncronas de las aplicaciones AJAX (Imagen original creada por Adaptive Path y utilizada con su permiso)

Las peticiones HTTP al servidor se sustituyen por peticiones JavaScript que se realizan al elemento encargado de AJAX. Las peticiones más simples no requieren intervención del servidor, por lo que la respuesta es inmediata. Si la interacción requiere una respuesta del servidor, la petición se realiza de forma asíncrona mediante AJAX. En este caso, la interacción del usuario tampoco se ve interrumpida

por recargas de página o largas esperas por la respuesta del servidor. (LibrosWeb, 2010)

2.7. Ventajas y desventajas

Ventajas

a) 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.

b) 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.

c) Ajax pueden ser de cualquier tamaño, desde las funciones simples a las muy complejas.

d) 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.

e) Válido en cualquier plataforma y navegador.

f) Así como AJAX funciona en cualquier navegador, es perfectamente compatible con cualquier tipo de servidor estándar y lenguaje de programación Web.

g) Se acerca más a la funcionalidad de una aplicación de escritorio.

h) El tiempo de espera para una petición se reduce

Desventajas

a) Problemas si el usuario ha deshabilitado el uso de JavaScript en su navegador.

b) A más Ajax, más uso de código JavaScript del lado del browser.

c) Se pierde la posibilidad de volver a la página anterior.

CAPÍTULO III

3. DESARROLLO DE LA APLICACIÓN WEB DINÁMICA

3.1. Análisis de la metodología UWE(UML-Based Web Engineering)

UWE es un método de ingeniería del software para el desarrollo de aplicaciones web basado en UML

En requisitos separa las fases de captura, definición y validación y hace además una clasificación y un tratamiento especial dependiendo del carácter de cada requisito.

Consiste en una notación y en un método:

- La notación se basa en UML para aplicaciones Web en general y para aplicaciones adaptativas en particular.
- UWE establece una separación estricta de aspectos en las fases tempranas de desarrollo e implementa un proceso de desarrollo manejado por modelos.

Los modelos se detallan a continuación:

3.1.1. Especificación de requisitos

El primer paso es la especificación de requerimientos de la aplicación, que en UWE se indica mediante un modelo de requerimientos. UWE propone dos niveles de detalle para modelar los requerimientos: primero un descripción general de los casos de uso y luego una descripción detallada.

3.1.2. Sobrevista de casos de uso

Los diagramas de casos de uso muestran las funcionalidades que tendrá el sistema y se construye con elementos UML: actor y caso de uso.

Los usuarios se un sistema web son: anónimos (usuario), usuarios registrados y el administrador del sistema web.

Un caso de uso es una descripción de los pasos o las actividades que deberán realizarse para llevar a cabo algún proceso. Los personajes o entidades que participarán en un caso de uso se denominan actores. En el contexto de ingeniería del software, un caso de uso es una secuencia de interacciones que se desarrollarán entre un sistema y sus actores en respuesta a un evento que inicia un actor principal sobre el propio sistema. Los diagramas de casos de uso sirven para especificar la comunicación y el comportamiento de un sistema mediante su interacción con los usuarios y/u otros sistemas. O lo que es igual, un diagrama que muestra la relación entre los actores y los casos de uso en un sistema. Una relación es una conexión entre los elementos del modelo, por ejemplo la especialización y la generalización son relaciones. Los diagramas de casos de uso se utilizan para ilustrar los requerimientos del sistema al mostrar cómo reacciona a eventos que se producen en su ámbito o en él mismo**.** (Riko, 2010)

3.1.3. Definición de contenido

Es el proceso de construcción de un modelo de los datos utilizados en una organización, independientemente de las consideraciones físicas y sirve para la construcción de un modelo conceptual de los datos de acuerdo a los requisitos, para ello es necesario realizar las siguientes acciones.

1. Identificar los tipos de entidad mediante la examinación de las especificaciones de requisitos de usuario.
2. Identificar los tipos de relación para lo cual se deben tomar en cuenta consideraciones gramaticales para identificar las relaciones a partir de la

especificación de requisitos, las mismas que se hacen mediante expresiones verbales.

3. Identificar y asociar los atributos con los tipos de entidad y relación para resaltar las diferencias entre entidades y determinar una superclase con atributos comunes.
4. Determinar los dominios de los atributos
5. Determinar los atributos de clave candidata, principal y alternativa. (Gonzaga, 2007)

3.1.4. Establecimiento de la estructura de navegación

Basados en el establecimiento de requerimientos y modelo de contenido se modela la estructura de navegación de una aplicación web. Las clases de navegación presentan nodos navegables, los enlaces de navegación muestran enlaces directos entre las clases de navegación, las rutas de navegación alternativa se muestran con menús. (Turmero, 2012)

3.1.5. Modelamiento de presentación

Este Modelo indica cuáles son las clases de presentación y de proceso que pertenecen a una página web. Es posible usar un Diagrama de Presentación con el fin de proveer esta información.

El modelo de presentación presenta una visión abstracta de la interfaz del usuario (UI) de la aplicación web. Está basado en el modelo de navegación y hace una abstracción de los aspectos concretos como son los colores, fuentes, y el sitio donde serán colocados los elementos en la página web, en su lugar el modelo describe la estructura básica de la interfaz detallando los elementos como anclas, formularios, imágenes, textos, botones.

3.2. Análisis y especificación de requisitos

Introducción

La Especificación de requerimientos de Software (ERS), es un documento base para el desarrollo del Portal Web de la Unidad Educativa "Liceo Oxford" Este documento se desarrolla siguiendo el estándar IEEE 830-1998 "Estándar IEEE Práctica Recomendada para la Especificación de Requisito Software".

3.2.1. Análisis del problema

A través de la presentación de un portal web y recolección de requisitos; en la Unidad Educativa "Liceo Oxford" el problema principal radica en la presentación de información a la comunidad en general la misma que debe ser actualizada constantemente por lo cual se propone como una aplicación dinámica.

Puesto que es una Institución educativa es necesario presentar información sobre sus servicios, instalaciones, proyecciones, ubicación y niveles.

3.2.2. Ámbito del Sistema

El sistema a realizar se denominará POWELIOX Portal Web Liceo Oxford, el cual deberá permitir ver toda información de acceso público a través de este, la gestión de áreas, materias, docentes, tareas, paralelos, méritos, becas, unidades de producción y la presentación de formularios para inscripciones en diferentes ámbitos como aplicación para inscripciones a nivel pre-escolar, escolar y secundaria.

3.2.3. Definiciones, acrónimos y abreviaturas

3.2.4. Definiciones

Tabla 3. 1.

Definiciones

Término	**Descripción**
Administrador	Usuario con privilegios para establecer altas, bajas y cambios o configuraciones necesarias en la gestión del potal web.
Responsable del control	Persona que inicializa el sistema y realiza el control diario de los formularios aplicados
Usuario consumidor	Persona que accede al portal público.
Alumno	Persona que estudia dentro de la institución
Docente	Persona que trabaja en la institución, puede ingresar planificaciones, tareas.

3.2.5. Acrónimos y abreviaturas

Tabla 3. 2.

Acrónimos y abreviaturas.

Acrónimo/abreviatura	Definición
POWELIOX	Portal Web Liceo Oxford
ERS	Especificación de requisitos de Software.
UI	Interfaz de usuario
UWE	UML Basado en Ingeniería Web
IEEE	Instituto de Ingenieros Eléctricos y Electrónicos.
UML	Lenguaje Unificado de Modelado

3.2.6. Referencias

Estándar IEEE 830-1998 (IEEE Recomended Practice for Software Requirements Specification).

3.2.7. Visión general del documento

El documento consta de tres secciones; La primera contiene una visión general del sistema a desarrollar. La segunda sección describe el funcionamiento del sistema, gestión de datos asociados y factores que inciden el sistema. Y en la última sección se definen los requisitos que debe satisfacer el sistema.

3.2.8. Descripción general

En esta sección se detalla de forma general el sistema, con el objetivo de conocer las principales funciones que realizará, tanto con los datos, restricciones, y cualquier factor que afecte al desarrollo del mismo.

3.2.9. Perspectiva del producto

El sistema debe presentar información publicitaria a la comunidad.

El funcionamiento esencial del portal es de presentar datos informativos sobre la Unidad educativa, recolectar datos proporcionados por los profesores y padre de familia que acceden al portal.

3.2.10. Funciones del Sistema

En forma general el sistema deberá dar y soportar las siguientes gestiones:

- Gestión de estudiantes
- Gestión de docentes
- Gestión de eventos
- Gestión de cronogramas
- Gestión de becas
- Gestión de galería
- Gestión de méritos
- Gestión de áreas académicas
- Gestión de materias
- Gestión de tareas
- Gestión de pasantías
- Gestión de la Unidad de Producción
- Gestión de productos
- Gestión de rasgos generales

- Gestión de cursos
- Gestión de paralelos
- Gestión de usuarios

Gestión de estudiantes

Permitirá el ingreso y modificación de los datos de los estudiantes y el acceso a ciertas funciones como las tareas.

Gestión de docentes

Permitirá el ingreso de datos de docentes, los mismos que ingresan las tareas para los estudiantes y cargan las planificaciones de la correspondiente materia.

Gestión de eventos

Permitirá la gestión de eventos como son ingreso, modificación o, estos según la programación de la dirección o rectorado.

Gestión de cronogramas de actividades

Permitirá gestionar los datos del cronograma permitiendo así el ingreso, modificación o eliminación, según la revisión y autorización del rectorado.

Gestión de becas

Permitirá hacer un seguimiento de las aplicaciones a becas llenadas por los estudiantes o sus representantes, para esto permitirán ingresar, o modificar.

Las becas estarán asociadas a un factor por el cual se aplica a la beca pudiendo ser éstos, académicos, deportivos o sociales.

Gestión de galería

Permitirá administrar la galería gráfica por eventos para esto se ingresa, o modifica y se asignan archivos que contiene las imágenes de la galería.

Gestión de méritos

Permitirá administrar datos de méritos estudiantiles y del personal, permitiendo así publicar dichos méritos alcanzados por periodos o por eventos para ello se realizará el ingreso, modificación o eliminación de dichos registros.

Gestión áreas académicas

Permitirá ingresar, modificar o eliminar las áreas académicas que existan dentro de la institución, las mismas que tendrán materias relacionadas.

Gestión de Materia

Permitirá el ingreso, modificación de materias, cada una de estas materias pertenecen a un área académica.

Gestión de Tareas

Permitirá la gestión de las tareas que pueden ser ingresadas, modificadas o eliminadas por el docente.

Gestión de pasantía

Permitirá gestionar la disponibilidad de pasantías según las áreas, para esto se ingresarán y modificarán los datos de las pasantías.

Gestión de Unidad de Producción

Permitirá el ingreso y modificación de una unidad de producción así como un archivo anexo de la jerarquía directiva de la misma (organigrama)

Gestión de producto

Se permitirá el ingreso, modificación de productos.

Gestión de la información básica de la unidad educativa

Permitirá ingresar o modificar la información básica de la institución como su misión, visión, ubicación geográfica e historia.

Gestión de Cursos y paralelos

Permitirá ingresar o modificar la información de los cursos con sus respectivos paralelos.

Gestión de usuarios

Permitirá administrar la creación o eliminación de usuarios del sistema.

3.2.11. Requisitos específicos

Requisitos funcionales

I. **Gestión de estudiantes**

(Req1). Ingresar un nuevo estudiante con los siguientes datos: cédula, nombres, apellidos, dirección, teléfono, e-mail, nombre del representante, fecha nacimiento, año lectivo, paralelo, código del curso o grado y un código que es autogenerado.

(Req2). Modificar los datos del estudiante a excepción del código

II. Gestión de eventos

(Req3). Ingresar eventos a llevarse a cabo con los datos. Código que es autogenerado, fecha publicación, fecha de inicio, fecha de finalización, lugar, hora, involucrados, nombre, descripción.
(Req4). Modificar evento a excepción del código
(Req5). Lista todos los eventos.

III. Gestión cronogramas de actividades

(Req6). Ingresa cronogramas con los datos: descripción, fecha de publicación, detalle del cronograma con los datos (fecha de inicio, fecha de finalización, descripción), periodo, código que es autogenerado.
(Req7). Modificar cronograma
(Req8) Lista un cronograma con sus respectivas actividades.
(Req9) Eliminar un registro de cronogramas.
(Req10) Si han sido eliminadas todas las actividades de un cronograma, dicho cronograma será eliminado también

IV. Gestión de becas

(Req11). Se ingresa una beca con los datos: nombre del estudiante, factor de beca, descripción, código de beca (autogenerado)
(Req12) Se permitirá la modificación de los datos de una beca.

V. Gestión de galería

(Req13). Ingresar galería con los datos: nombre del evento de la galería, urlGalería, código(autogenerado), descripción.
(Req14). Modificar la galería excepto el código.

VI. Gestión de méritos

(Req15). Ingresar mérito con los datos: tipo de mérito (académico, deportivo, científico, profesional, cultural), nombre del acreedor del mérito (estudiante o docente), código del mérito (autogenerado), y una foto.
(Req16). Modificar el mérito

VII. Gestión de áreas académicas

(Req17). Ingresar área con los siguientes datos: nombre del área, y código (autogenerado).
(Req18). Modificar el área excepto el código.
(Req19). Para el ingreso de un área se deberá considerar que las materias que están dentro de esta tengan relación.

VIII. Gestión de materia

(Req20). Ingresar la materia con los datos: nombre, cantidad de horas, área.
(Req21). Modificar los datos de la materia.
(Req22) La materia corresponde a un curso y tiene una planificación, y está dentro de un horario por cursos.

IX. Gestión de tarea

(Req23) Se ingresa la tarea con los datos: nombre de la materia, nombre del profesor, grado o curso, fecha de actual, fecha de entrega, puntuación, descripción de la tarea y código (autogenerado).
(Req24). Modificar los datos de la tarea excepto el código
(Req25). Eliminar la tarea, previa confirmación.

X. Gestión de pasantía

(Req26). Ingresar la pasantía con los datos: área, descripción, fecha inicio y código (autogenerado), observaciones.
(Req27). Modificar los datos de la pasantía excepto el código

XI. Gestión de paralelo

(Req28). Ingresar el paralelo con los datos: código autogenerado, curso, descripción, url de horario, periodo lectivo.
(Req29). Modificar los datos del paralelo.

XII. Gestión de unidad de producción

(Req30). Ingresar una unidad de producción con los datos: código (autogenerado), fecha de creación, ubicación física, responsable a cargo, dirección url del archivo que contiene el organigrama.
(Req31). Modificar los datos de la unidad de producción

XIII. Gestión de producto

(Req32). Ingresar el producto con los datos: descripción, código de la unidad, código del producto (autogenerado).
(Req33). Modificar los datos del producto excepto el código

XIV. Gestión de docentes

(Req34) Se ingresará un docente con los datos: código (autogenerado), cedula, nombres, apellidos, dirección, teléfono, e-mail, título, área a la que corresponde.
(Req35) Modificar datos de docente excepto el código

XV. Gestión de Cursos

(Req36) Ingresar la descripción de un curso y su código autogenerado.
(Req37) Modificar los datos correspondientes a un curso.

XVI. Gestión de Rasgos generales

(Req38) Permitirá añadir la información básica de la unidad educativa: misión, visión, objetivo, url de historia, ubicación, nombre del rector, teléfono, e-mail, descripción de extensiones; en caso que no exista al menos una ingresada.
(Req39) Modificar la información básica de la unidad si ya existe previamente ingresada.

XVII. Gestión de Usuarios

(Req40) Permite registrar un nuevo usuario del sistema, teniendo en cuenta qué tipo de usuario es si administrador o docente, un nombre de usuario, una clave y el código que se autogenera.
(Req41) Eliminar usuarios de la lista que se muestra.

Requisitos visuales

(Req42) Se permitirá ver en la pantalla principal la misión, visión, objetivo, información de contacto, nombre de la primera autoridad.

Requisitos de seguridad

(Req 43) Se controlará el acceso a ciertas pantallas según el usuario para poder ingresar, modificar o eliminar información.

3.2.12. Requisitos De Interfaces Externas

a. Interfaces de Usuario

Las interfaces de usuario se desarrollan en un ambiente de páginas web

b. Interfaces de Hardware

Se trabajara en plataforma cliente/servidor.

3.2.13. Requisitos De Rendimiento

Que las páginas no tarden en cargar la información.

3.2.14. Requisitos De Desarrollo

El ciclo de vida para el desarrollo del producto será el secuencial.

3.2.15. Requisitos Tecnológicos

a. Hardware: Pc. Pentium IV o superior
b. Sistema Base: Sistema Operativo Windows 7
c. Base de datos: SQL server R2
d. Lenguaje de programación: Visual Basic.Net
e. Diseño de pantallas ASP.Net con AJAX

3.2.16. Atributos del sistema

a. Seguridad

Cuando un usuario ingrese al portal web puede visualizar la información de la unidad, si un usuario registrado desea ingresar para realizar algún cambio debe ingresar el usuario y la clave o password, si los datos ingresados no son los correctos se le indicara un mensaje de error.

3.3. INGENIERÍA DE REQUERIMIENTOS DEL POWELIOX

3.3.1. Modelo de requerimientos

a) Lista de actores

Tabla 3. 3.

Lista de actores

Actor	Descripción
Administrador	Es el usuario que tiene acceso a todas las funciones del sistema, por lo tanto tiene permisos para gestionar: usuarios, ingreso o modificación de datos, además de realizar el mantenimiento y actualización continua de los privilegios asignados.
Docente	Es el usuario que puede ingresar a ciertas funcionalidades de ingreso o modificación de datos según su usuario registrado.
Estudiante	Son todos los Usuarios que acceden a una funcionalidad limitada del portal pero total de visualización de información.
Usuario	Son los que pueden ingresar al portal y visualizar toda la información básica del mismo.

b) Diagramas de casos de uso

Figura 3. 1. Diagrama de CU de gestión de estudiante

Figura 3. 2. Diagrama de CU de gestión de docente

Figura 3. 3. Diagrama de CU de gestión de evento

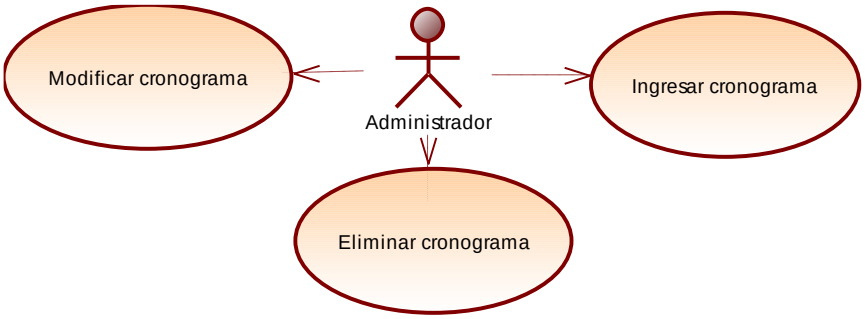

Figura 3. 4. Diagrama de CU de gestión de cronograma

Figura 3. 5. Diagrama de CU de gestión de beca

Figura 3. 6. Diagrama de CU de gestión de galería

Figura 3. 7. Diagrama de CU de gestión de mérito

Figura 3. 8. Diagrama de CU de gestión de área

Figura 3. 9. Diagrama de CU de gestión de materia

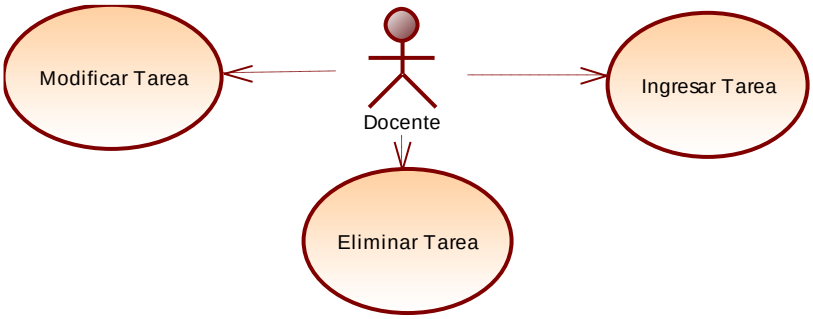

Figura 3. 10. Diagrama de CU de gestión de tarea

Figura 3. 11. Diagrama de CU de gestión de pasantía

Figura 3. 12. Diagrama de CU de gestión de unidad de producción

Figura 3. 13. Diagrama de CU de gestión de producto

Figura 3. 14. Diagrama de CU de gestión de rasgos generales

Figura 3. 15. Diagrama de CU de gestión de curso

Figura 3. 16. Diagrama de CU de gestión de paralelo

Figura 3. 17. Diagrama de CU de gestión de usuarios

1. Casos de uso detallados

Tabla 3. 4.

Descripción del CU "Ingresar estudiante"

Nombre: CU01	**Ingresar estudiante**
Descripción:	**Permite que el administrador pueda ingresar datos de un estudiante.**
Propósito:	**Ingresar los datos de un estudiante**
Actores:	**Usuario Administrador**
Precondiciones:	**Debe identificarse como usuario administrador.**
Flujo Normal:	
Actor	**Sistema**
1) El caso de uso inicia cuando el actor selecciona la opción nuevo del menú estudiante **3) Llena los campos con los datos de un estudiante.** **4) Selecciona la opción Guardar**	2) Muestra la pantalla de ingreso con los campos que debe llenar: cédula, nombres, apellidos, año lectivo, fecha de nacimiento, dirección, número de teléfono, correo electrónico, nombre del representante, el curso y el paralelo. 5) Valida si todos los campos están llenos y si los datos son correctos. 6) Almacena en la base de datos 7) Muestra un mensaje que indica que los datos fueron almacenados con éxito. 8) Se vacían los campos.
Flujo Alternativo: **5. Si hay campos vacíos o incorrectos se muestra un mensaje con el error correspondiente.** **6. No almacenó retorna un mensaje que indica que el registro no fue almacenado.**	
Post condiciones: **El sistema debe permitir almacenar el registro.**	
Requisito: Req1	

Tabla 3. 5.

Descripción del CU "Modificar estudiante"

Nombre: CU02	**Modificar estudiante**
Descripción:	**Permite que el administrador pueda modificar los datos de un estudiante.**
Propósito:	**Modificar los datos de un estudiante.**
Actores:	**Usuario Administrador**
Pre condiciones:	
• **Debe haberse identificado como administrador.**	
• **Deben haber registros almacenados.**	
Flujo Normal:	
Actor	**Sistema**
1) El caso de uso inicia cuando el actor selecciona la opción modificar del menú estudiante.	2) Muestra un listado con los registros de los estudiantes.
3) Selecciona el registro que desea del listado.	4) Muestra una pantalla con los campos: cédula, nombres, apellidos, dirección, teléfono, e-mail, nombre del representante, curso, paralelo, fecha de nacimiento, año lectivo.
5) Cambia los datos que sean necesarios.	6) Valida si todos los campos están llenos y son correctos los datos.
7) Selecciona la opción Guardar.	8) Modifica el registro en la base de datos.
	9) Muestra un mensaje de Registro modificado con éxito.
	10) Retorna al listado.
Flujo Alternativo:	
2. No hay estudiantes registrados	
6. Si hay campos vacíos o incorrectos se muestra un mensaje con el error Correspondiente.	
9. No modificó el registro en la base de datos.	
Post condiciones:	
El sistema debe permitir la modificación de los registros.	
Requisito: Req2	

Tabla 3. 6.

Descripción del CU "Ingresar docente"

Nombre:CU03	**Ingresar docente**
Descripción:	**Permite que el administrador pueda ingresar datos de un docente.**
Propósito:	**Ingresar un nuevo registro de docentes.**
Actores:	**Usuario Administrador**
Pre condiciones:	**El usuario debe haberse identificado como administrador**
Flujo Normal:	
Actor	**Sistema**
1) El caso de uso inicia cuando el actor selecciona la opción ingresar del menú docente.	2) Muestra la pantalla de ingreso con los campos: nombres, apellidos, dirección, teléfono, e-mail, área, título.
3) Llena los campos con los datos de un docente	4) Valida si todos los campos están llenos y si los datos son correctos.
4) Selecciona la opción guardar.	5) Almacena en la base de datos.
	6) Muestra un mensaje de ingresado con éxito.
Flujo Alternativo:	
4. Si hay campos vacíos o incorrectos se muestra un mensaje con el error correspondiente.	
6. Si no se pudo ingresar en la base de datos.	
Post condiciones:	
El sistema debe permitir el ingreso de nuevos registros..	
Requisito: Req34	

Tabla 3. 7.

Descripción del CU "Ingresar docente"

Nombre:CU04	**Modificar docente**
Descripción:	**Permite que el administrador pueda modificar los datos de un docente.**
Propósito:	**Modificar los datos de un docente registrado.**
Actores:	**Usuario Administrador**
Flujo Normal:	
Actor	**Sistema**
1) El caso de uso inicia cuando el actor selecciona la opción modificar del menú docente. **3) Selecciona un registro del listado.** **5) Cambia los datos que sean necesarios.** **7) Selecciona la opción guardar.**	2) Muestra un listado con los registros encontrados 4) Muestra los campos: cédula, nombres, apellidos, dirección, teléfono, e-mail, título y área. 6) Valida si todos los campos están llenos y si los datos son correctos. 8) Modifica el registro en la base de datos. 9) Muestra un mensaje de modificación con éxito. 10) Retorna al listado.
Flujo Alternativo:	
2. Que no existan registros de docentes	
6. Si hay campos vacíos o incorrectos se muestra un mensaje con el error correspondiente.	
9. Si no se puedo almacenar en la base de datos muestra un mensaje.	
Post condiciones:	
El sistema debe permitir la modificación de los datos de un docente.	
Requisito: Req35	

Tabla 3. 8.

Descripción del CU "Modificar docente"

Nombre:CU04	**Modificar docente**
Descripción:	**Permite que el administrador pueda modificar los datos de un docente.**
Propósito:	**Modificar los datos de un docente registrado.**
Actores:	**Usuario Administrador**
Flujo Normal:	
Actor	**Sistema**
1) El caso de uso inicia cuando el actor selecciona la opción modificar del menú docente. **3) Selecciona un registro del listado.** **5) Cambia los datos que sean necesarios.** **7) Selecciona la opción guardar.**	2) Muestra un listado con los registros encontrados 4) Muestra los campos: cédula, nombres, apellidos, dirección, teléfono, e-mail, título y área. 6) Valida si todos los campos están llenos y si los datos son correctos. 8) Modifica el registro en la base de datos. 9) Muestra un mensaje de modificación con éxito. 10) Retorna al listado.
Flujo Alternativo:	
2. Que no existan registros de docentes	
6. Si hay campos vacíos o incorrectos se muestra un mensaje con el error correspondiente.	
9. Si no se puedo almacenar en la base de datos muestra un mensaje.	
Post condiciones:	
El sistema debe permitir la modificación de los datos de un docente.	
Requisito: Req35	

Tabla 3. 9.

Descripción del CU "Ingresar evento"

Nombre: CU05	**Ingresar evento**
Descripción:	**Permite que el administrador pueda ingresar un nuevo registro de un evento.**
Propósito:	**Ingresar un nuevo evento.**
Pre condiciones:	**El usuario debe haberse identificado como administrador.**
Actores:	**Usuario Administrador**
Flujo Normal:	
Actor	**Sistema**
1) El caso de uso inicia cuando el actor selecciona la opción nuevo del menú evento. **3) Llena los campos con los datos de un evento.** **4) Seleccionar la opción guardar.**	2) Muestra la pantalla de ingreso con los campos nombre del evento, descripción, fecha de inicio, fecha de finalización, lugar del evento, involucrados (docentes, alumnos, padres, comunidad, personal administrativo, todos). 4) Valida si todos los campos están llenos y si los datos son correctos. 5) Almacena en la base de datos. 6) Muestra un mensaje de almacenado con éxito.
Flujo Alternativo:	
4. Si hay campos vacíos o incorrectos se muestra un mensaje con el error correspondiente.	
5. Si no se puedo almacenar en la base de datos muestra un mensaje.	
Post condiciones:	
El sistema debe permitir el ingreso de nuevos eventos.	
Requisito: Req3	

Tabla 3. 10.

Descripción del CU "Modificar evento"

Nombre:CU06	**Modificar evento**
Descripción:	**Permite que el administrador pueda modificar datos de un registro de evento.**
Propósito:	**Modificar los datos de un evento.**
Actores:	**Usuario Administrador**
Pre condiciones:	**El usuario debe haberse identificado como administrador.**
Flujo Normal:	
Actor	**Sistema**
1) El caso de uso inicia cuando el actor selecciona la opción modificar del menú eventos. **3) Selecciona un registro del listado.** **5) Cambia los datos que desea modificar.** **7) Selecciona la opción guardar.**	2) Muestra un listado con los registros encontrados según el criterio de búsqueda aplicado. 4) Muestra una pantalla con los datos correspondientes al registro seleccionado. 6) Valida si todos los campos están llenos y si los datos son correctos. 8) Modifica en la base de datos. 9) Muestra un mensaje de modificación exitosa.
Flujo Alternativo:	
2. Si no existen registros.	
6. Si hay campos vacíos o incorrectos se muestra un mensaje con el error correspondiente.	
9. Si no se puedo modificar en la base de datos el sistema el sistema muestra un mensaje de error en la modificación.	
Post condiciones:	
El sistema debe permitir modificar un registro.	
Requisito: Req4	

Tabla 3. 11.

Descripción del CU "Ingresar cronograma"

Nombre:CU07	**Ingresar cronograma**
Descripción:	**Permite que el administrador pueda ingresar datos de cronograma.**
Propósito:	**Ingresar un cronograma con su detalle correspondiente.**
Actores:	**Usuario Administrador**
Pre condiciones: **El usuario debe haberse identificado como administrador**	
Flujo Normal:	
Actor	**Sistema**
1) El caso de uso inicia cuando el actor elige la opción nuevo del menú cronogramas. **3) Llena los campos con los datos del cronograma.** **4) Selecciona la opción guardar.** **8) Llena los campos del detalle** **9) Selecciona la opción guardar**	2) Muestra la pantalla de ingreso de cronogramas con los campos periodo lectivo y descripción. 5) Valida si todos los campos están llenos y si los datos son correctos. 6) Guarda el registro en la base de datos y se muestra un mensaje que indica que se deben ingresar las actividades del cronograma 7) Muestra una pantalla de los detalles del cronograma con los campos descripción, fecha de inicio y fecha de finalización. 10) Valida si todos los campos están llenos. 11) Almacena en la base de datos. 12) Muestra un mensaje de ingreso exitoso.
Flujo Alternativo: **5,10. Si hay campos vacíos o incorrectos se muestra un mensaje con el error correspondiente.** **6,11. Si no se puedo ingresar en la base de datos se muestra un mensaje de error en el ingreso.**	
Post condiciones: **El sistema debe permitir ingresar un nuevo cronograma.**	
Requisito: Req6	

Tabla 3. 12.

Descripción del CU "Modificar cronograma"

Nombre:CU08	**Modificar cronograma**
Descripción:	**Permite que el administrador pueda modificar datos de un registro de cronograma.**
Propósito:	**Ingresar un cronograma.**
Actores:	**Usuario Administrador**
Pre condiciones:	**El usuario debe haberse identificado como administrador.**
Flujo Normal:	
Actor	**Sistema**
1) El caso de uso inicia cuando el actor selecciona la opción modificar del menú cronogramas. **3) Selecciona un registro del listado.** **5) Selecciona una actividad.** **8) Cambia los datos que va a modificar.** **9) Selecciona la opción guardar.**	2) Muestra el listado de registros encontrados. 4) Muestra el listado de actividades del cronograma seleccionado. 7) Muestra los datos: descripción, fecha inicio, fecha de finalización. 10) Valida si todos los campos están llenos y si los datos son correctos. 11) Modifica el registro en la base de datos. 12) Muestra un mensaje de modificación exitosa.
Flujo Alternativo: **2,4. Si no existen registros asociados al criterio de búsqueda, muestra un mensaje.** **10. Si hay campos vacíos o incorrectos se muestra un mensaje con el error correspondiente.** **12. Si no se puedo modificar en la base de datos muestra un mensaje de error en la modificación.**	
Post condiciones:	

El sistema debe permitir la modificación de un registro.
Requisito: Req7

Tabla 3. 13.

Descripción del CU "Eliminar cronograma

Nombre:CU09	**Eliminar cronograma**
Descripción:	**Permite que el administrador pueda eliminar un registro de cronograma.**
Propósito:	**Eliminar un cronograma de actividades.**
Actores:	**Usuario Administrador**
Pre condiciones:	**El usuario debe haberse identificado como administrador.**
Flujo Normal:	
Actor	**Sistema**
1) El caso de uso inicia cuando el actor selecciona la opción eliminar del menú cronogramas. **3) Selecciona un registro del listado.** **5) Selecciona la actividad a eliminar** **8) Selecciona la opción Aceptar de la ventana emergente.**	2) Muestra un listado de registros encontrados según el criterio de búsqueda aplicado. 4) Muestra la lista de actividades del cronograma. 7) Muestra un mensaje de confirmación de borrado de la actividad seleccionada. Si han sido borradas todas las actividades el cronograma también se elimina. 9) Elimina el registro de la base de datos. 10) Muestra un mensaje de eliminación exitosa.
Flujo Alternativo:	
2,4. Si no existen registros el sistema muestra un mensaje. **8. Si da clic sobre el botón cancelar no se elimina el registro seleccionado.** **10. Si no se pudo eliminar el registro de la base de datos se muestra un mensaje de error.**	
Post condiciones:	
El sistema debe permitir la eliminación de un cronograma.	
Requisito: Req8	

Descripción del CU "Ingresar beca"

Nombre:CU10	**Ingresar beca**
Descripción:	**Permite que el administrador pueda ingresar datos de una beca.**
Propósito:	**Ingresar datos correspondientes a una beca.**
Actores:	**Usuario Administrador**
Pre condiciones:	**El usuario debe haberse identificado como administrador.**
Flujo Normal:	
Actor	**Sistema**
1) El caso de uso inicia cuando el actor selecciona la opción nuevo del menú becas. **3) Llena los campos con los datos de la beca.** **4) Selecciona la opción guardar.**	2) Muestra la pantalla de ingreso de becas con los campos nombres del estudiante, motivo de la beca, descripción. 5) Valida si todos los campos están llenos y si los datos son correctos. 6) Almacena en la base de datos. 7) Muestra un mensaje de ingreso exitoso.
Flujo Alternativo:	
5. Si hay campos vacíos o incorrectos se muestra un mensaje con el error correspondiente. **7. Si no se pudo guardar en la base de datos muestra un mensaje de error en guardado.**	
Post condiciones:	
El sistema debe permitir el ingreso de nuevos registros.	
Requisito: Req11	

Tabla 3. 14.

Descripción del CU "Modificar beca"

Nombre:CU11	**Modificar beca**
Descripción:	**Permite que el administrador pueda modificar datos de un registro de becas.**
Propósito:	**Modificar los datos correspondientes a una beca.**
Actores:	**Usuario Administrador**
Flujo Normal:	
Actor	**Sistema**
1) El caso de uso inicia cuando el actor selecciona la opción modificar del menú Becas. **3) Selecciona un registro del listado.** **5) Cambia los datos que va a modificar.** **6) Selecciona la opción guardar.**	2) Muestra un listado con los registros encontrados. 4) Muestra los datos nombre del estudiante, motivo de la beca, descripción. 7) Valida si todos los campos están llenos y si los datos son correctos. 8) Modifica el registro en la base de datos. 9) Muestra un mensaje de modificación exitosa.
Flujo Alternativo:	
2. Si no existen registros se muestra un mensaje. **7. Si hay campos vacíos o incorrectos se muestra un mensaje con el error correspondiente.** **9. Si no se pudo modificar en la base de datos se muestra un mensaje.**	
PoPost condiciones:	
El sistema debe permitir la modificación de una beca.	
Requisito: Req12	

Tabla 3. 15.

Descripción del CU "Ingresar galería"

Nombre:CU12	**Ingresar galería**
Descripción:	**Permite que el administrador pueda ingresar datos de una galería.**
Propósito:	**Ingresar una galería fotográfica de un evento.**
Actores:	**Usuario Administrador**
Pre condiciones:	
• **El usuario debe haberse identificado como administrador.** • **Debe haber un evento previamente ingresado.**	
Flujo Normal:	
Actor	**Sistema**
1) El caso de uso inicia cuando el actor selecciona la opción nuevo del menú galería. **3) Llena los campos con los datos de la galería.** **4) Selecciona la opción guardar.**	2) Muestra la pantalla de ingreso de galería con los campos nombre del evento, descripción, url de fotos. 5) Valida si todos los campos están llenos y si los datos son correctos. 6) Almacena en la base de datos. 7) Muestra un mensaje de ingreso exitoso.
Flujo Alternativo:	
5. Si hay campos vacíos o incorrectos se muestra un mensaje con el error correspondiente. **7. Si no se pudo ingresar en la base de datos se muestra un mensaje.**	
Post condiciones:	
El sistema debe permitir que se ingrese nuevos registros.	
Requisito: Req13	

Tabla 3. 16.

Descripción del CU "Modificar galería"

Nombre:CU13	**Modificar galería**
Descripción: **Permite que el administrador pueda modificar datos de un registro de galería.**	
Pre condiciones: • **Modificar los datos correspondientes a una galería.** • **Debe haber registros previamente ingresados.**	
Actores: **Usuario Administrador**	
Pre condiciones: **El usuario debe haberse identificado como administrador.**	
Flujo Normal:	
Actor	**Sistema**
1) El caso de uso inicia cuando el actor selecciona la opción modificar del menú galería **3) Selecciona un registro del listado.** **5) Cambia los datos que va a modificar.** **6) Selecciona la opción guardar.**	2) Muestra un listado con los registros. 4) Muestra los datos nombre del evento, descripción, url de fotografías. 7) Valida si todos los campos están llenos y si los datos son correctos. 8) Modifica el registro en la base de datos. 9) Muestra un mensaje de modificación exitosa.
Flujo Alternativo: **2. Si no existen registros asociados al criterio de búsqueda se muestra un mensaje.** **7. Si hay campos vacíos o incorrectos se muestra un mensaje con el error correspondiente.** **9. Si no se pudo modificar el registro se muestra un mensaje.**	
Post condiciones: **El sistema debe permitir modificar los datos de un registro de galería.**	
Requisito: Req14	

Tabla 3. 17.

Descripción del CU "Ingresar mérito"

Nombre:CU14	**Ingresar mérito**
Descripción:	**Permite que el administrador pueda ingresar datos de mérito.**
Propósito:	**Ingresar datos respectivos a un mérito.**
Actores:	**Usuario Administrador**
Pre condiciones: **El usuario debe haberse identificado como administrador.**	
Flujo Normal:	
Actor	**Sistema**
1) El caso de uso inicia cuando el actor selecciona la opción nuevo del menú mérito. **3) Llena los campos con los datos del mérito** **4) Selecciona la opción guardar.**	2) Muestra la pantalla de ingreso de méritos con los campos tipo de mérito, nombre del estudiante o nombre del docente, url fotografía, descripción. 5) Valida si todos los campos están llenos y si los datos son correctos. 6) Almacena el registro en la base de datos. 7) Muestra un mensaje de ingreso exitoso.
Flujo Alternativo: **5. Si hay campos vacíos o incorrectos se muestra un mensaje con el error correspondiente.** **8. Si no se puedo ingresar muestra un mensaje.**	
Post condiciones: **El sistema debe permitir el ingreso de un nuevo registro de méritos.**	
Requisito: Req17	

Tabla 3. 18.

Descripción del CU "Modificar mérito"

Nombre:CU15	**Modificar mérito**
Descripción:	**Permite que el administrador pueda modificar datos de un registro de mérito.**
Propósito:	**Modificar los datos de un registro de méritos.**
Actores:	**Usuario Administrador**
Pre condiciones	
• **El usuario debe haberse identificado como administrador.**	
• **Deben haber registros previamente ingresados.**	
Flujo Normal:	
Actor	**Sistema**
1) El caso de uso inicia cuando el actor selecciona la opción modificar del menú méritos.	2) Muestra el listado de registros encontrados.
3) Selecciona un registro del listado.	4) Muestra los datos tipo de mérito, nombre del docente o nombre del docente, fotografía, descripción.
5) Cambia los datos que va a modificar.	7) Valida si todos los campos están llenos y si los datos son correctos.
6) Selecciona la opción guardar.	8) Modifica el registro en la base de datos.
	9) Muestra un mensaje de modificación exitosa.
Flujo Alternativo:	
2. Si no existen registros, se muestra un mensaje indicando que no hay resultados que mostrar.	
7. Si hay campos vacíos o incorrectos se muestra un mensaje con el error correspondiente.	
9. Si no se pudo modificar el registro en la base de datos se muestra un mensaje.	
Requisito: Req16	

Tabla 3. 19.

Descripción del CU "Ingresar área"

Nombre:CU16	**Ingresar área**
Descripción:	**Permite que el administrador pueda ingresar datos de un área.**
Propósito:	**Ingresar datos de un área.**
Actores:	**Usuario Administrador**
Pre condiciones: **El usuario debe haberse identificado como administrador.**	
Flujo Normal:	
Actor	**Sistema**
1) El caso de uso inicia cuando el actor selecciona la opción nuevo del menú área.	2) Muestra la pantalla de ingreso de área, con el campo nombre del área.
3) Llena el campo nombre del área.	5) Valida si está lleno el campo.
4) Selecciona la opción guardar.	6) Almacena en la base de datos.
	7) Muestra mensaje de ingreso exitoso.
Flujo Alternativo:	
5. Si el campo está vacío se muestra un mensaje con el error correspondiente.	
6. Si no se puedo almacenar en la base de datos muestra un mensaje.	
Post Condiciones:	
El sistema debe permitir el ingreso de un área.	
Requisito: Req19, Req 17	

Tabla 3. 20.

Descripción del CU "Modificar área"

Nombre:CU17	**Modificar área**
Descripción:	**Permite que el administrador pueda modificar datos de un registro de área.**
Propósito:	**Modificar los datos de un área.**
Actores:	**Usuario Administrador**
Precondiciones: **El usuario debe haberse identificado como administrador.**	
Flujo Normal:	
Actor	**Sistema**
1) El caso de uso inicia cuando el actor selecciona la opción modificar del menú áreas.	2) Muestra un listado con los registros encontrados.
3) Selecciona un registro del listado.	4) Muestra los datos correspondientes al registro seleccionado.
5) Cambia los datos que va a modificar	7) Valida si los campos están llenos.
6) Selecciona el botón guardar.	8) Modifica el registro en la base de datos.
	9) Muestra un mensaje de modificación exitosa.
Flujo Alternativo:	
2. Si no existen registros.	
7. Si hay campos vacíos o incorrectos se muestra un mensaje con el error correspondiente.	
9. Si no se pudo modificar en la base de datos se muestra un mensaje.	
Post condiciones:	
El sistema debe permitir la modificación de datos de un área.	
Requisito: Req18	

Tabla 3. 21.

Descripción del CU "Ingresar materia"

Nombre:CU18	**Ingresar materia**
Descripción:	**Permite que el administrador pueda ingresar datos de materia.**
Propósito:	**Ingresar datos correspondientes a una materia**
Actores:	**Usuario Administrador**
Pre condiciones: **El usuario debe haberse identificado como administrador.**	
Flujo Normal:	
Actor	**Sistema**
1) El caso de uso inicia cuando el actor selecciona la opción nuevo del menú materia.	2) Muestra la pantalla de ingreso de materias con los campos nombre, área, cantidad de horas.
3) Llena los campos con los datos de la materia	5) Valida si todos los campos están lleno y si los datos son correctos.
4) Selecciona el botón guardar.	6) Almacena en la base de datos.
	7) Muestra un mensaje de ingreso exitoso.
Flujo Alternativo:	
5. Si hay campos vacíos o incorrectos se muestra un mensaje con el error correspondiente. 6. Si no se pudo almacenar en la base de datos se muestra un mensaje.	
Post condiciones:	
El sistema debe permitir el ingreso de materias.	
Requisito: Req20	

Tabla 3. 22.

Descripción del CU "Modificar materia"

Nombre:CU19	**Modificar materia**
Descripción: **Permite que el administrador pueda modificar datos de un registro de materia.**	
Propósito: **Modificar los datos correspondientes a una materia.**	
Actores: **Usuario Administrador**	
Pre condiciones: **El usuario debe haberse identificado como administrador.**	
Flujo Normal:	
Actor	**Sistema**
1) El caso de uso inicia cuando el actor selecciona la opción modificar del menú materias.	2) Muestra un listado con los registros encontrados.
3) Selecciona un registro del listado.	4) Muestra los datos nombre de la materia, área, cantidad de horas.
5) Cambia los datos que va a modificar	7) Valida si todos los campos están llenos y si los datos son correctos.
6) Selecciona el botón guardar.	8) Modifica el registro en la base de datos.
	9) Muestra un mensaje de modificación exitosa.
Flujo Alternativo:	
2. Si no existen registros.	
7. Si hay campos vacíos o incorrectos se muestra un mensaje con el error correspondiente.	
9. Si no se pudo modificar en la base de datos se muestra un mensaje.	
Post condiciones:	
El sistema debe permitir la modificación de los datos de una materia.	
Requisito: Req21	

Tabla 3. 23.

Descripción del CU "Ingresar tarea"

Nombre:CU20	**Ingresar tarea**
Descripción: **Permite que el docente pueda ingresar datos de una tarea.**	
Propósito: **Ingresar datos referentes a una tarea.**	
Actores: **Usuario Docente**	
Pre condiciones: **El usuario debe haberse identificado como docente.**	
Flujo Normal:	
Actor	**Sistema**
1) El caso de uso inicia cuando el actor selecciona la opción nuevo del menú tareas.	2) Muestra la pantalla de ingreso de tareas con los campos fecha actual, fecha de entrega, puntuación, paralelo, nombre del docente, descripción, nombre de la materia.
3) Llena los campos con los datos de la tarea	5) Valida si todos los campos están llenos y si los datos son correctos.
4) Selecciona el botón guardar.	6) Almacena en la base de datos.
	7) Muestra un mensaje de ingreso exitoso.
Flujo Alternativo:	
5. Si hay campos vacíos o incorrectos se muestra un mensaje con el error correspondiente.	
6. Si no se puedo guardar el registro se muestra un mensaje.	
Post condiciones:	
El sistema debe permitir el ingreso de nuevos registros.	
Requisito: Req23	

Tabla 3. 24.

Descripción del CU "Modificar tarea"

Nombre:CU21	**Modificar tarea**
Descripción:	**Permite que el docente pueda modificar datos de un registro de tareas.**
Propósito:	**Modificar los datos de una tarea.**
Actores:	**Usuario Docente**
Pre condiciones: **El usuario debe haberse identificado como docente.**	
Flujo Normal:	
Actor	**Sistema**
1) El caso de uso inicia cuando el actor selecciona la opción modificar del menú tareas. **3) Selecciona un registro del listado.** **5) Cambia los datos que va a modificar a excepción de la fecha actual.** **6) Selecciona el botón guardar.**	2) Muestra un listado con los registros encontrados según los criterios de búsqueda aplicados. 4) Muestra los datos fecha, fecha de entrega, puntuación, curso, nombre del docente, descripción, nombre de la materia. 7) Valida si todos los campos están llenos y si los datos son correctos. 8) Modifica el registro en la base de datos. 9) Muestra un mensaje de modificación exitosa.
Flujo Alternativo: **2. Si no existen registros.** **7. Si hay campos vacíos o incorrectos se muestra un mensaje con el error correspondiente.** **9. Si no se pudo modificar el registro se muestra un mensaje.**	
Post condiciones: **El sistema debe permitir la modificación de los datos de una tarea.**	
Requisito: Req24	

Tabla 3. 25.

Descripción del CU "Eliminar tarea"

Nombre:CU22	**Eliminar tarea**
Descripción:	**Permite que el docente pueda eliminar un registro de las tareas.**
Propósito:	**Eliminar un registro de tareas.**
Actores:	**Usuario Docente**
Pre condiciones: **El usuario debe haberse identificado como docente.**	
Flujo Normal:	
Actor	**Sistema**
1) El caso de uso inicia cuando el actor selecciona la opción eliminar del menú tareas. **3) Selecciona un registro del listado.**	2) Muestra un listado de registros encontrados según los criterios de búsqueda aplicados. 4) Elimina el registro de la base de datos. 5) Muestra un mensaje de eliminación exitosa
Flujo Alternativo: **7. Si no pudo eliminar de la base de datos se muestra un mensaje.**	
Post condiciones: **El sistema debe permitir la eliminación de una tarea.**	
Requisito: Req25	

Tabla 3. 26.

Descripción del CU "Ingresar pasante"

Nombre:CU23	**Ingresar pasantía**
Descripción: **Permite que el administrador pueda ingresar datos referentes a una pasantía.**	
Propósito: **Ingresar datos referentes a una pasantía**	
Actores: **Usuario Administrador**	
Pre condiciones: **El usuario debe haberse identificado como administrador.**	
Flujo Normal:	
Actor	**Sistema**
1) El caso de uso inicia cuando el actor selecciona la opción nuevo del menú pasantías. **3) Llena los campos con los datos del pasante.** **4) Selecciona el botón guardar.**	2) Muestra la pantalla de ingreso de pasantes con los campos área, fecha de inicio, descripción, nivel (inicial, básica, bachillerato). 5) Valida si todos los campos están llenos y si los datos son correctos. 6) Almacena en la base de datos. 7) Muestra un mensaje de ingreso exitoso.
Flujo Alternativo: **5. Si hay campos vacíos o incorrectos se muestra un mensaje con el error correspondiente.** **6. Si no se pudo ingresar en la base de datos se muestra un mensaje.**	
Post condiciones: **El sistema debe permitir el ingreso de nuevos registros en pasante.**	
Requisito: Req26	

Tabla 3. 27.

Descripción del CU "Modificar pasante"

Nombre:CU24	**Modificar pasantía**
Descripción: **Permite que el administrador pueda modificar datos de un registro de pasantía.**	
Propósito: **Modificar datos de una pasantía.**	
Actores: **Usuario Administrador**	
Pre condiciones: **El usuario debe haberse identificado como administrador.**	
Flujo Normal:	
Actor	**Sistema**
1) El caso de uso inicia cuando el actor selecciona la opción modificar del menú pasantía. **3) Selecciona un registro del listado.** **5) Cambia los datos que va a modificar.** **6) Selecciona el botón guardar.**	2) Muestra un listado con los registros encontrados. 4) Muestra los datos: fecha de inicio, área, descripción, nivel. 7) Valida si todos los campos están llenos y si los datos son correctos 8) Modifica el registro en la base de datos. 9) Muestra un mensaje de modificación exitosa.
Flujo Alternativo: **2. Si no se encontraron registros.** **7. Si hay campos vacíos o incorrectos se muestra un mensaje con el error correspondiente.** **9. Si no se pudo modificar el registro se muestra un mensaje.**	
Post condiciones: **El sistema debe permitir la modificación de un registro.**	
Requisito: Req27	

Tabla 3. 28.

Descripción del CU "Ingresar Unidad de producción"

Nombre:CU25	**Ingresar unidad de producción**
Descripción: **Permite que el administrador pueda ingresar datos referentes a una unidad de producción.**	
Propósito: **Ingresar datos referentes a una unidad de producción.**	
Actores: **Usuario Administrador**	
Pre condiciones: **El usuario debe haberse identificado como administrador.**	
Flujo Normal:	
Actor	**Sistema**
1) El caso de uso inicia cuando el actor selecciona la opción nuevo del menú Unidad de producción.	2) Muestra la pantalla de ingreso de unidades de producción con los campos descripción, ubicación, responsable, fecha de creación, url del organigrama..
3) Llena los campos con los datos de la unidad de producción	5) Valida si todos los campos están llenos y si lo datos son correctos.
4) Selecciona el botón guardar.	6) Almacena en la base de datos.
	7) Muestra un mensaje de ingreso exitoso.
Flujo Alternativo: **5. Si hay campos vacíos o incorrectos se muestra un mensaje con el error correspondiente.** **7. Si no se pudo guardar los datos se muestra un mensaje.**	
Post condiciones: **El sistema debe permitir que se ingrese los datos de una unidad de producción.**	
Requisito: Req30	

Tabla 3. 29.

Descripción del CU "Modificar unidad de producción"

Nombre:CU26	**Modificar unidad de producción**
Descripción: **Permite que el administrador pueda modificar datos de una Unidad de producción.**	
Propósito: **Modificar los datos referentes a una unidad de producción.**	
Actores: **Usuario Administrador**	
Pre condiciones: **El usuario debe haberse identificado como administración.**	
Flujo Normal:	
Actor	**Sistema**
1) El caso de uso inicia cuando el actor selecciona modificar del menú unidad de producción.	2) Muestra un listado con los registros encontrados.
3) Selecciona un registro del listado.	4) Muestra los datos descripción, ubicación, responsable, fecha de creación y url del organigrama.
5) Cambia los datos que va a modificar	7) Valida si todos los campos están llenos y si los datos son correctos
6) Selecciona el botón guardar.	8) Modifica el registro en la base de datos.
	9) Muestra un mensaje de modificación exitosa.
Flujo Alternativo: **2. Si no existen registros.** **7. Si hay campos vacíos o incorrectos se muestra un mensaje con el error correspondiente.** **9. Si no se pudo modificar los datos en la base de datos se muestra un mensaje.**	
Post condiciones: **El sistema debe permitir la modificación de datos de una unidad de producción.**	
Requisito: Req31	

Tabla 3. 30.

Descripción del CU "Ingresar producto"

Nombre:CU27	**Ingresar producto**
Descripción: **Permite que el administrador pueda ingresar datos referentes a un producto.**	
Propósito: **Ingresar datos de un producto.**	
Actores: **Usuario Administrador**	
Pre condiciones: • **El usuario debe haberse identificado como administrador.** • **Es necesario que haya al menos una unidad de producción registrada.**	
Flujo Normal:	
Actor	**Sistema**
1) El caso de uso inicia cuando el actor selecciona la opción nuevo del menú productos. **3) Llena los campos con los datos del pasante** **4) Selecciona el botón guardar.**	2) Muestra la pantalla de ingreso de productos con los campos descripción, composición, nombre de la unidad a la que pertenece. 5) Valida si todos los campos están llenos y si los datos son correctos. 6) Almacena en la base de datos. 7) Muestra un mensaje de ingreso exitoso.
Flujo Alternativo **5. Si hay campos vacíos o incorrectos se muestra un mensaje con el error correspondiente.** **7. Si no se pudo guardar en la base de datos se muestra un mensaje.**	
Post condiciones: **El sistema debe permitir el ingreso de nuevos productos.**	
Requisito: Req32	

Tabla 3. 31.

Descripción del CU "Modificar producto"

Nombre:CU28	**Modificar producto**
Descripción: **Permite que el administrador pueda modificar datos de un producto.**	
Propósito: **Modificar los datos de un producto.**	
Actores: **Usuario Administrador**	
Pre condiciones: • **El usuario debe haberse identificado como administrador** • **Debe haber al menos una unidad de producción registrada.**	
Flujo Normal:	
Actor	**Sistema**
1) El caso de uso inicia cuando el actor selecciona la opción modificar del menú productos. **3) Selecciona un registro del listado.** **5) Cambia los datos que va a modificar.** **6) Selecciona el botón guardar.**	2) Muestra un listado con los registros encontrados. 4) Muestra los datos correspondientes al registro seleccionado. 7) Valida si todos los campos están llenos y si los datos son correctos. 8) Modifica el registro en la base de datos. 9) Muestra un mensaje de modificación exitosa.
Flujo Alternativo: **2. Si no existen registros.** **7. Si hay campos vacíos o incorrectos se muestra un mensaje con el error correspondiente.** **9. Si no se pudo guardar en la base de datos se muestra un mensaje.**	
Post condiciones: **El sistema debe permitir la modificación de un registro de productos.**	
Requisito: Req33	

Tabla 3. 32.

Descripción del CU "Ingresar rasgos generales"

Nombre:CU29	**Ingresar Rasgos Generales**
Descripción: **Permite que el administrador pueda ingresar datos referentes a la información de la unidad educativa.**	
Propósito: **Ingresar datos referentes a la unidad educativa.**	
Actores: **Usuario Administrador**	
Pre condiciones: **El usuario debe haberse identificado como administrador.**	
Flujo Normal:	
Actor	**Sistema**
1) El caso de uso inicia cuando el actor selecciona la opción General del menú. **3) El actor elige una opción.** **5) Modifica los datos que desea.** **6) Selecciona el botón guardar.**	2) Muestra la pantalla con 2 opciones: una de nuevo y otra de modificar. 4) a.- Si es nuevo, verifica si ya existe un registro previo, y despliega un mensaje que indica que no se puede agregar uno nuevo. b.-Si es modificar , verifica si existe un registro previamente ingresado y muestra la pantalla con los datos: visión, misión, objetivo, información de contacto, máxima autoridad. 7) Valida si todos los campos están llenos y si los datos son correctos. 8) Almacena en la base de datos. 9) Muestra un mensaje de ingreso exitoso.
Flujo Alternativo: **7. Si hay campos vacíos o incorrectos se muestra un mensaje con el error correspondiente.** **9. Si no se pudo modificar el registro muestra un mensaje.**	
Post condiciones: **El sistema debe permitir el ingreso o modificación de los datos básicos de la unidad educativa.**	
Requisito: Req38,39	

Tabla 3. 33.

Descripción del CU "Ingresar curso"

Nombre:CU30	**Ingresar curso**
Descripción: **Permite que el administrador pueda ingresar datos referentes a un curso.**	
Propósito: **Ingresar datos referentes a un curso.**	
Actores: **Usuario Administrador**	
Pre condiciones: **El usuario debe haberse identificado como administrador.**	
Flujo Normal:	
Actor	**Sistema**
1) El caso de uso inicia cuando el actor selecciona la opción nuevo del menú curso. **3) Llena los campos con los datos del curso.** **4) Selecciona el botón guardar.**	2) Muestra la pantalla de ingreso de curso con el campo: descripción y nivel (Inicial, Básica, bachillerato). 5) Valida si todos los campos están llenos y si los datos son correctos. 6) Almacena en la base de datos. 7) Muestra un mensaje de ingreso exitoso.
Flujo Alternativo: **5. Si hay campos vacíos o incorrectos se muestra un mensaje con el error correspondiente.** **7. Si no su pudo ingresar los datos de un curso se muestra un mensaje.**	
Post condiciones: **El sistema debe permitir el ingreso de datos de un curso.**	
Requisito: Req36	

Tabla 3. 34.

Descripción del CU "Modificar curso"

Nombre:CU31	**Modificar curso**
Descripción:	**Permite que el administrador pueda modificar datos de un curso.**
Propósito:	**Modificar los datos de un curso.**
Actores:	**Usuario Administrador**
Pre condiciones:	• **El usuario debe haberse identificado como administrador.** • **Es necesario que existan registros de paralelo previamente ingresados.**
Flujo Normal:	
Actor	**Sistema**
1) El caso de uso inicia cuando el actor selecciona la opción modificar del menú curso. **3) Selecciona un registro del listado.** **5) Cambia la información.** **6) Selecciona el botón guardar.**	2) Muestra un listado con los registros encontrados. 4) Muestra los campos descripción y nivel. 7) Valida si los campos están llenos y si son correctos. 8) Almacena en la base de datos. 9) Muestra un mensaje de modificación exitosa.
Flujo Alternativo:	**2. Si no existen registros.** **7. Si el campo está vacío o si es incorrecto se muestra un mensaje con el error correspondiente.** **9. Si no se pudo modificar el registro se muestra un mensaje.**
Post condiciones:	**El sistema debe permitir la modificación de un registro de un curso.**
Requisito: Req37	

Tabla 3. 35.

Descripción del CU "Ingresar paralelo"

Nombre:CU32	**Ingresar paralelo**
Descripción:	**Permite que el administrador pueda ingresar datos referentes a un paralelo.**
Propósito:	**Ingresar datos referentes a un paralelo.**
Actores:	**Usuario Administrador**
Pre condiciones:	**El usuario debe haberse identificado como administrador.**
Flujo Normal:	
Actor	**Sistema**
1) El caso de uso inicia cuando el actor selecciona la opción nuevo del menú paralelo. **3) Llena los campos con los datos del paralelo.** **4) Selecciona el botón guardar.**	2) Muestra la pantalla de ingreso de paralelo con los campos: curso, descripción, url de horario, periodo lectivo. 5) Valida si todos los campos están llenos y si los datos son correctos. 6) Almacena en la base de datos. 7) Muestra un mensaje de ingreso exitoso.
Flujo Alternativo:	**5. Si hay campos vacíos o incorrectos se muestra un mensaje con el error correspondiente.** **7. Si no su pudo ingresar los datos de un paralelo se muestra un mensaje.**
Post condiciones:	**El sistema debe permitir el ingreso de datos de un paralelo.**
Requisito: Req28	

Tabla 3. 36.

Descripción del CU "Modificar paralelo"

Nombre:CU33	**Modificar paralelo**
Descripción:	**Permite que el administrador pueda modificar datos de un paralelo.**
Propósito:	**Modificar los datos de un paralelo.**
Actores:	**Usuario Administrador**
Pre condiciones: • **El usuario debe haberse identificado como administrador.** • **Es necesario que existan registros de paralelo previamente ingresados.**	
Flujo Normal:	
Actor	**Sistema**
1) El caso de uso inicia cuando el actor selecciona la opción modificar del menú paralelo. **3) Selecciona un registro del listado.** **5) Cambia la información.** **6) Selecciona el botón guardar.**	2) Muestra un listado con los registros encontrados. 4) Muestra los campos: descripción, periodo, curso y url del horario. 7) Valida si los campos están llenos y si son correctos correcto. 8) Almacena en la base de datos. 9) Muestra un mensaje de modificación exitosa.
Flujo Alternativo: **2. Si no existen registros** **7. Si el campo está vació o si es incorrecto se muestra un mensaje con el error correspondiente.** **9. Si no se pudo modificar el registro se muestra un mensaje.**	
Post condiciones: **El sistema debe permitir la modificación de un registro de paralelo.**	
Requisito: Req29	

Tabla 3. 37.

Descripción del CU "Ingresar usuario"

Nombre:CU34	**Ingresar usuario**
Descripción:	**Permite que el administrador pueda ingresar datos referentes a un usuario.**
Propósito:	**Ingresar datos referentes a un usuario.**
Actores:	**Usuario Administrador**
Pre condiciones: **El usuario debe haberse identificado como administrador.**	
Flujo Normal:	
Actor	**Sistema**
1) El caso de uso inicia cuando el actor selecciona la opción nuevo del menú Registrar usuarios. **3) Llena los campos con los datos del usuario.** **4) Selecciona el botón guardar.**	2) Muestra la pantalla de ingreso de usuarios con los campos: tipo de personal, nombre de usuario y clave. 5) Valida si todos los campos están llenos y si los datos son correctos. 6) Almacena en la base de datos. 7) Muestra un mensaje de ingreso exitoso.
Flujo Alternativo **5. Si hay campos vacíos o incorrectos se muestra un mensaje con el error correspondiente.** **7. Si no su pudo ingresar los datos de un usuario se muestra un mensaje.**	
Post condiciones: **El sistema debe permitir el ingreso de datos de un usuario.**	
Requisito: Req40	

Tabla 3. 38.

Descripción del CU "Eliminar usuario"

Nombre:CU35	**Eliminar usuario**
Descripción:	**Permite que el administrador pueda eliminar un registro de usuarios.**
Propósito:	**Eliminar un registro seleccionado.**
Actores:	**Usuario Administrador**
Pre condiciones:	**El usuario debe haberse identificado como administrador.**
Flujo Normal:	
Actor	**Sistema**
1) El caso de uso inicia cuando el actor selecciona la opción eliminar del menú registrar usuario.	2) Muestra un listado con los registros encontrados.
3) Selecciona un registro del listado.	4) Muestra un mensaje de confirmación de borrado del registro seleccionado.
5) De la ventana emergente selecciona el botón aceptar.	6) Elimina el registro seleccionado de la base de datos.
	7) Muestra un mensaje de eliminación exitosa.
Flujo Alternativo:	
5. Si no existen registros.	
7. Si selecciona el botón cancelar de la ventana emergente no se ejecuta la eliminación.	
10. Si no se pudo eliminar de la base de datos.	
Post condiciones:	
El sistema debe permitir la eliminación de un registro del usuario.	
Requisito: Req41	

3.4. Modelo de análisis

En esta parte se desarrollarán los modelos de contenido, navegación y presentación propuestos por UWE.

a) Modelo de contenido

Es un modelo conceptual que muestra las entidades del sistema web.

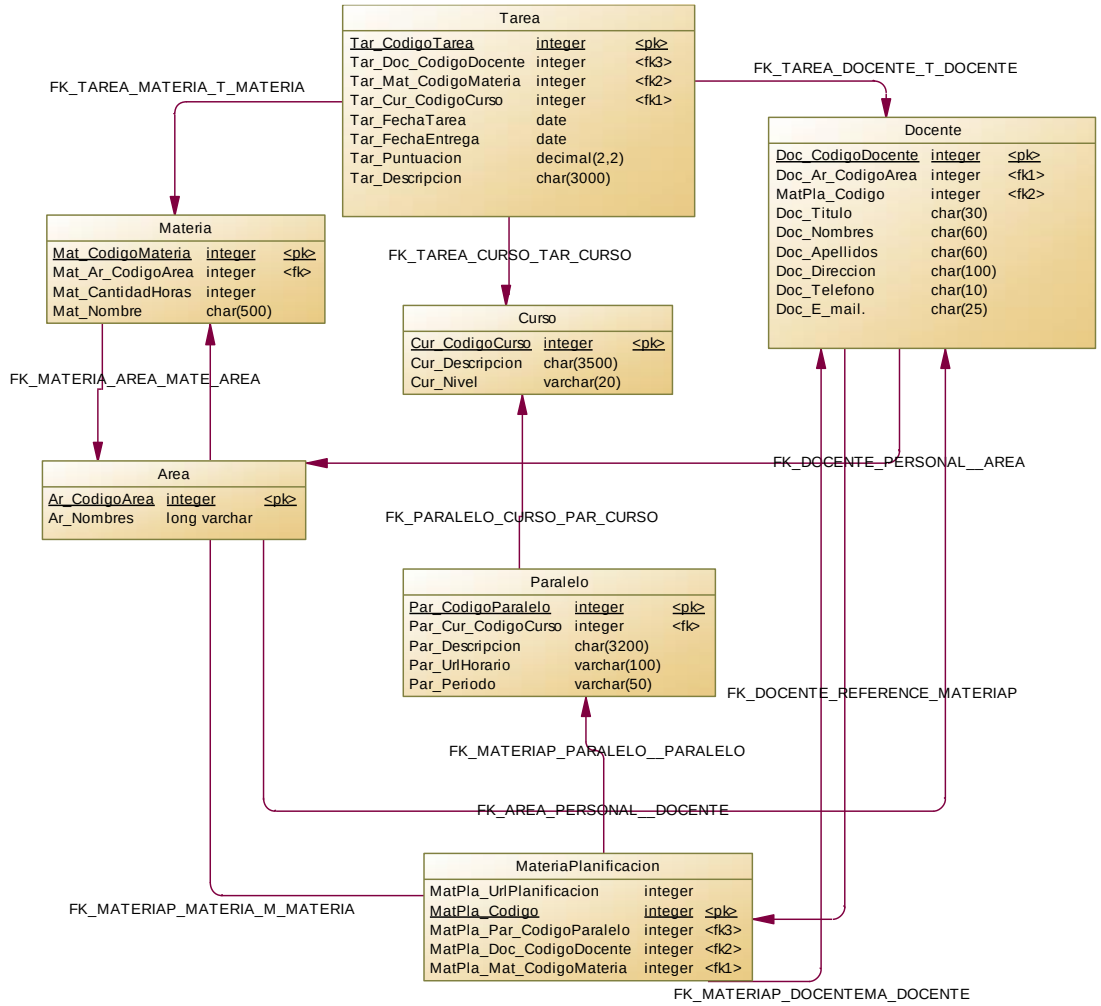

Figura 3. 18. Figura Clases de gestión de tareas

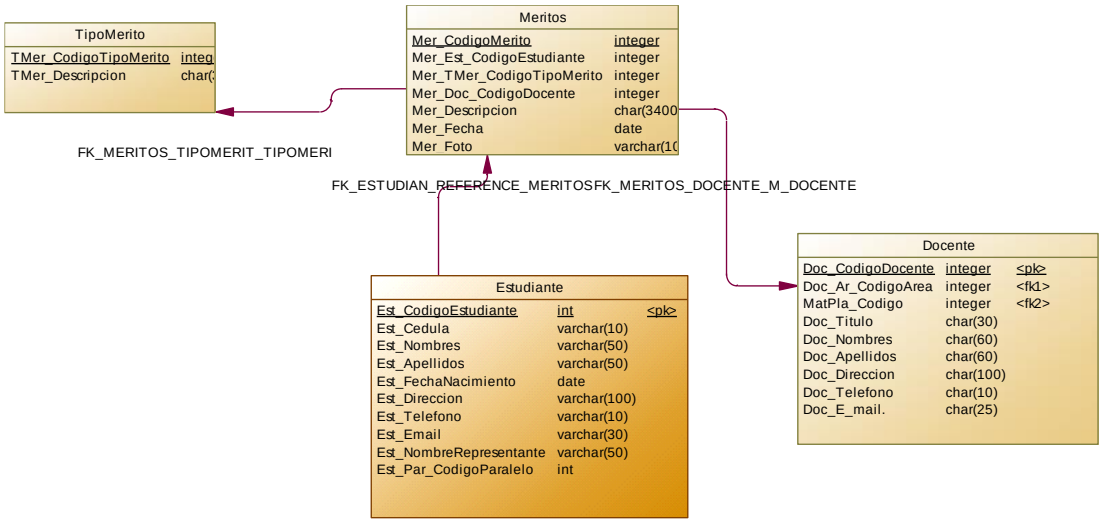

Figura 3. 19. Clases de gestión de méritos

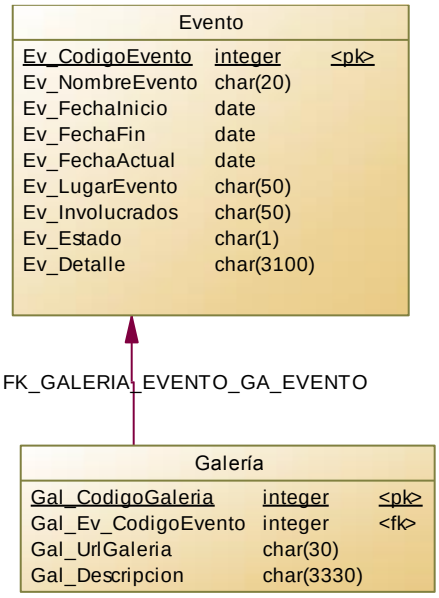

Figura 3. 20. Clases de la gestión de eventos y galería

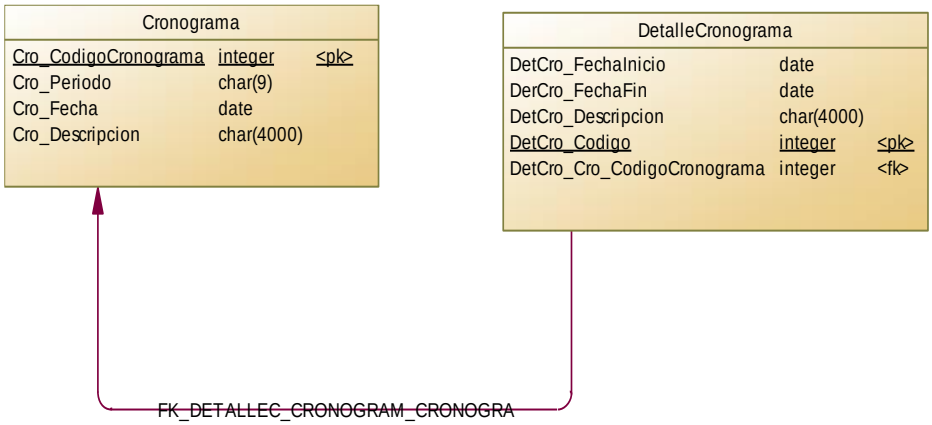

Figura 3. 21. Clases de la gestión cronogramas

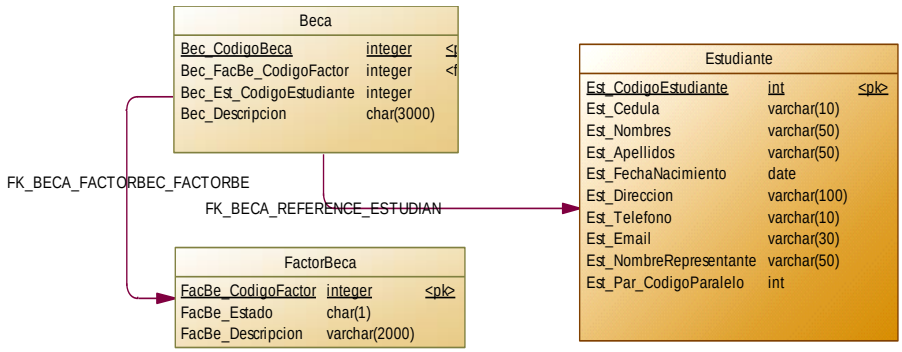

Figura 3. 22. Clases de la gestión becas

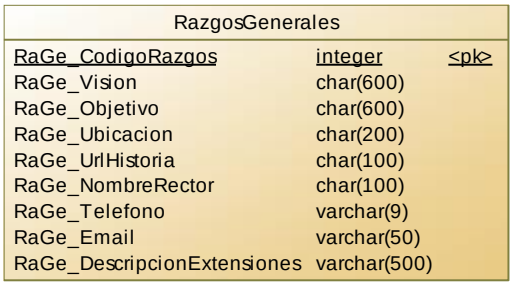

Figura 3. 23. Clases de la gestión de rasgos generales

Figura 3. 24. Clases de la gestión de productos

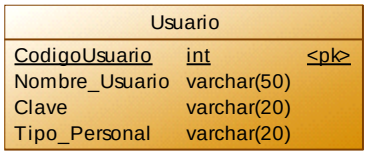

Figura 3. 25. Clases de la gestión de usuarios

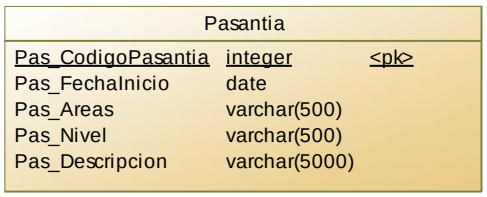

Figura 3. 26. Clases de la gestión de pasantías

b) Modelo de navegación

La tabla expuesta a continuación muestra los símbolos que se utilizan en el modelo de navegación.

Tabla 3. 39.

Notación UWE para el modelo de navegación

Elemento	Estereotipo UWE	Símbolo
Clase de navegación	<<Clase de navegación>>	
Menú	<<menú>>	
Índice	<<índice>>	
Consulta	<<consulta>>	
Clase de proceso	<<clase de proceso>>	

Se presenta el diagrama de navegación para las clases relevantes del sistema.

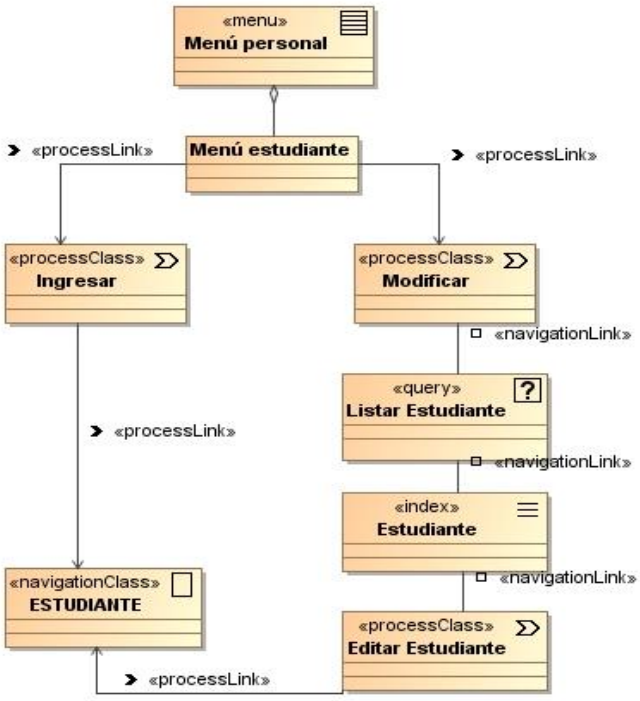

Figura 3. 27. Diagrama de navegación de configuración de clases básicas

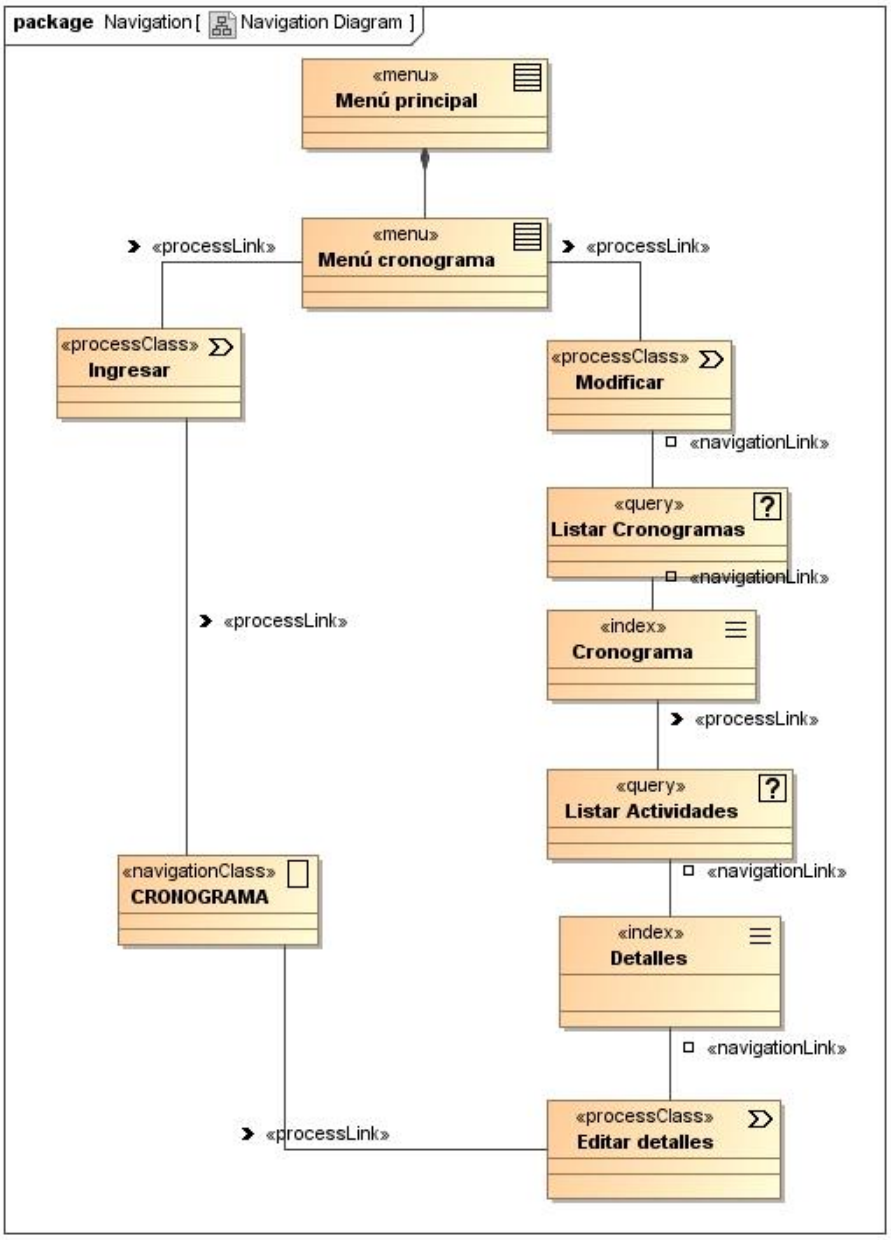

Figura 3. 28. Diagrama de navegación de configuración de clase cronograma y sus detalles

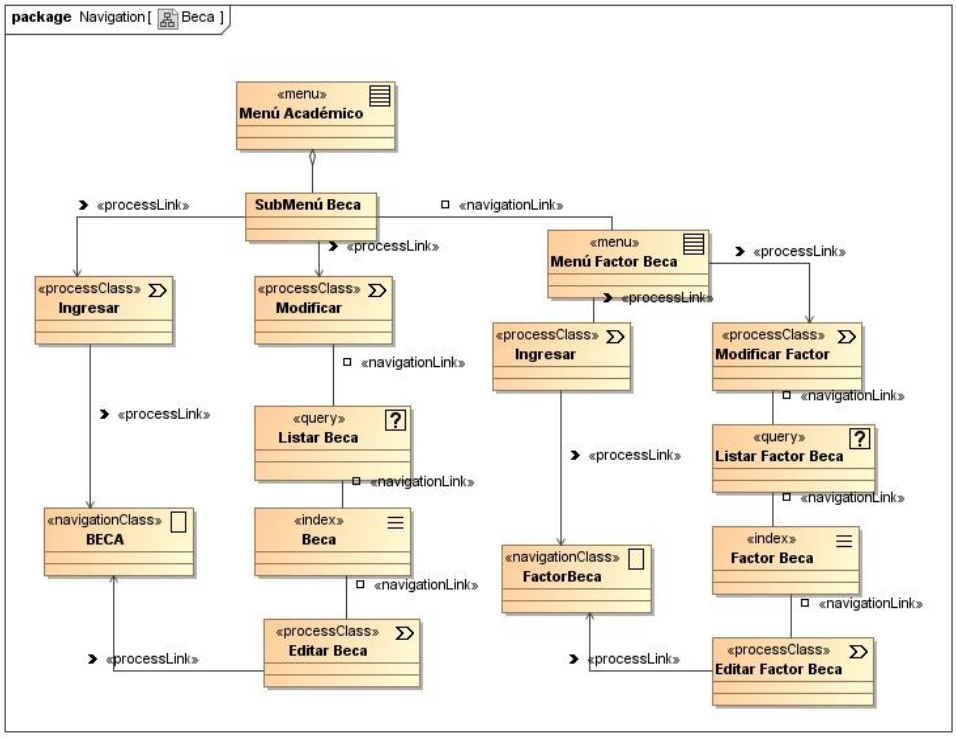

Figura 3. 29. Diagrama de navegación una clase que requiere de un tipo en otra

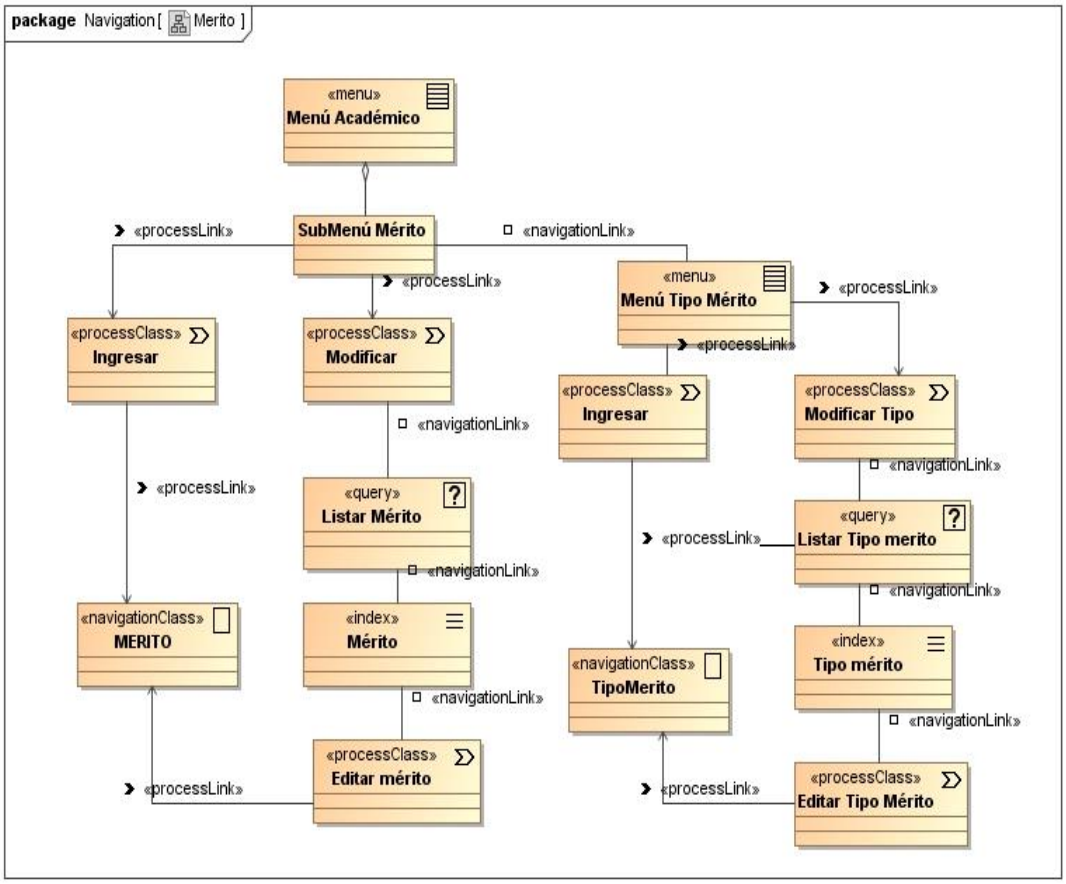

Figura 3. 30. Diagrama de navegación de mérito y tipo de mérito

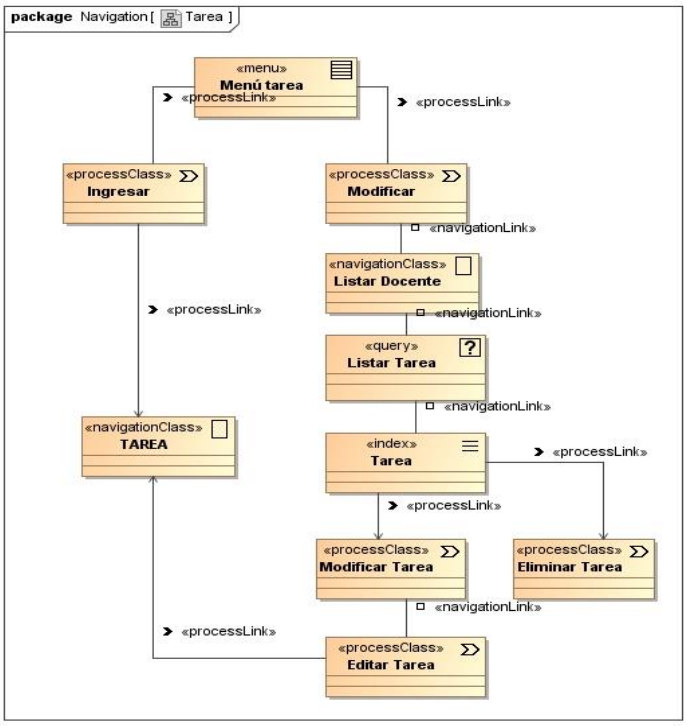

Figura 3. 31. Diagrama de navegación de la clase tarea

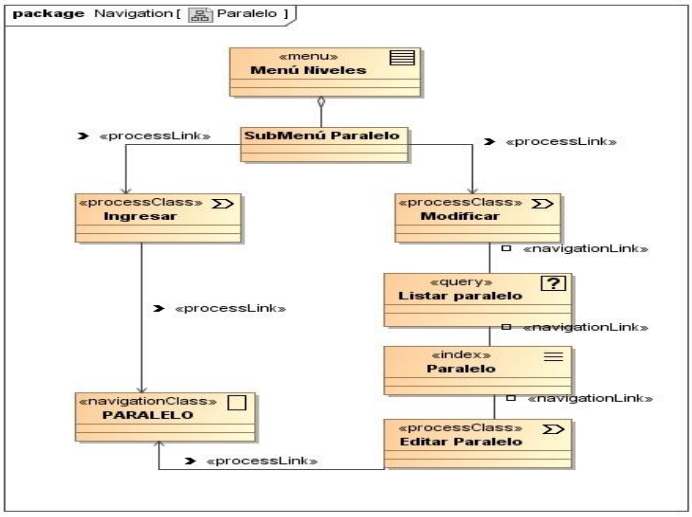

Figura 3. 32. Diagrama de navegación de la clase paralelo

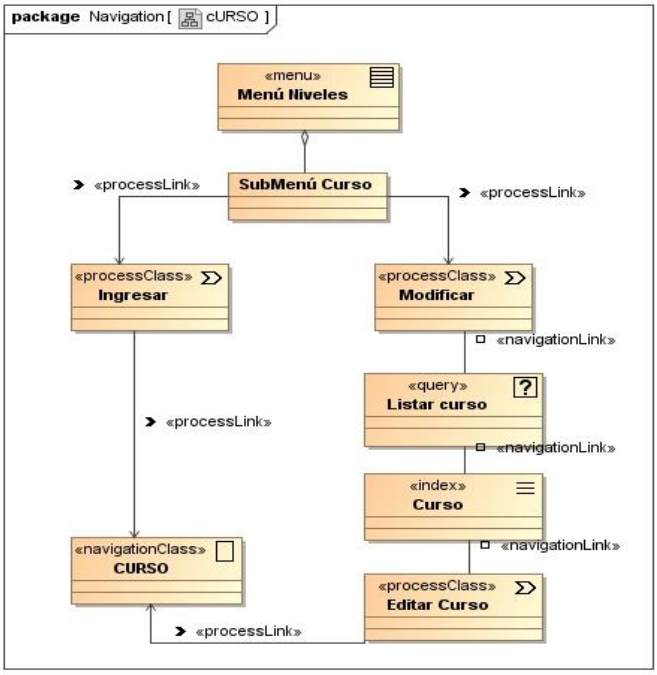

Figura 3. 33. Diagrama de navegación de la clase curso

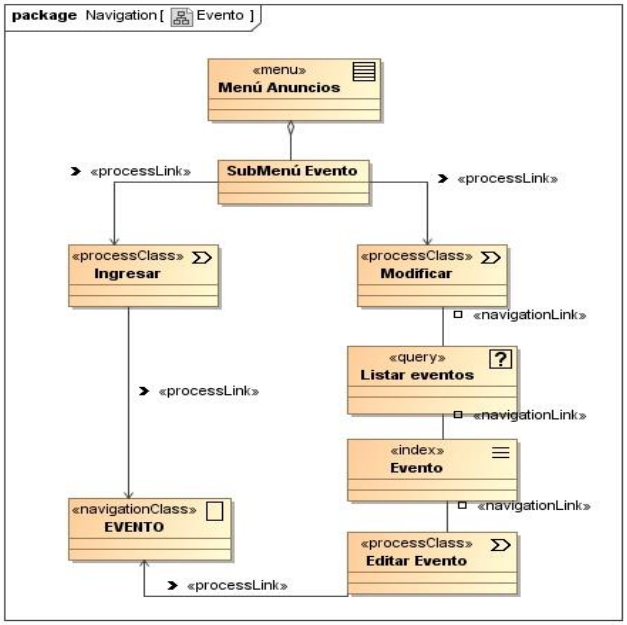

Figura 3. 34. Diagrama de navegación de la clase evento

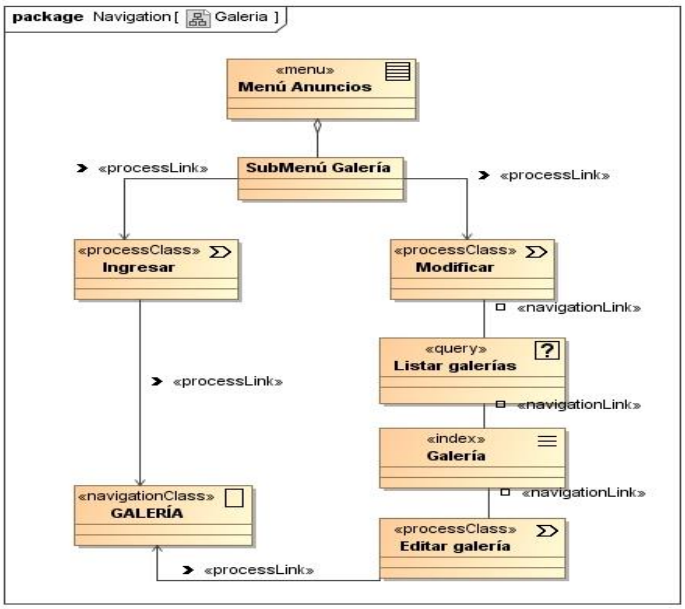

Figura 3. 35. Diagrama de navegación de la clase galería

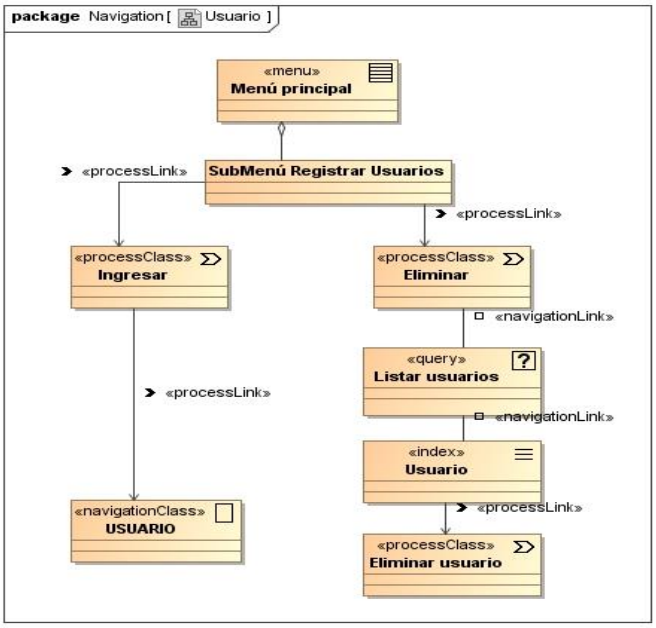

Figura 3. 36. Diagrama de navegación de la clase usuario

c) Modelo de presentación

La notación UWE para los elementos de clases de presentación se muestra en la Tabla

Tabla 3. 40.

Notación UWE para el modelo de presentación

Elemento	Estereotipo UWE	Símbolo
Clase de presentación	<<Clase de presentación>>	
Texto	<<texto>>	
Ancla	<<ancla>>	
Botón	<<botón>>	
Formulario	<<formulario>>	
Página de presentación	<<página de presentación >>	
Colección de anclas	<<colección de anclas>>	
Imagen	<<imagen>>	

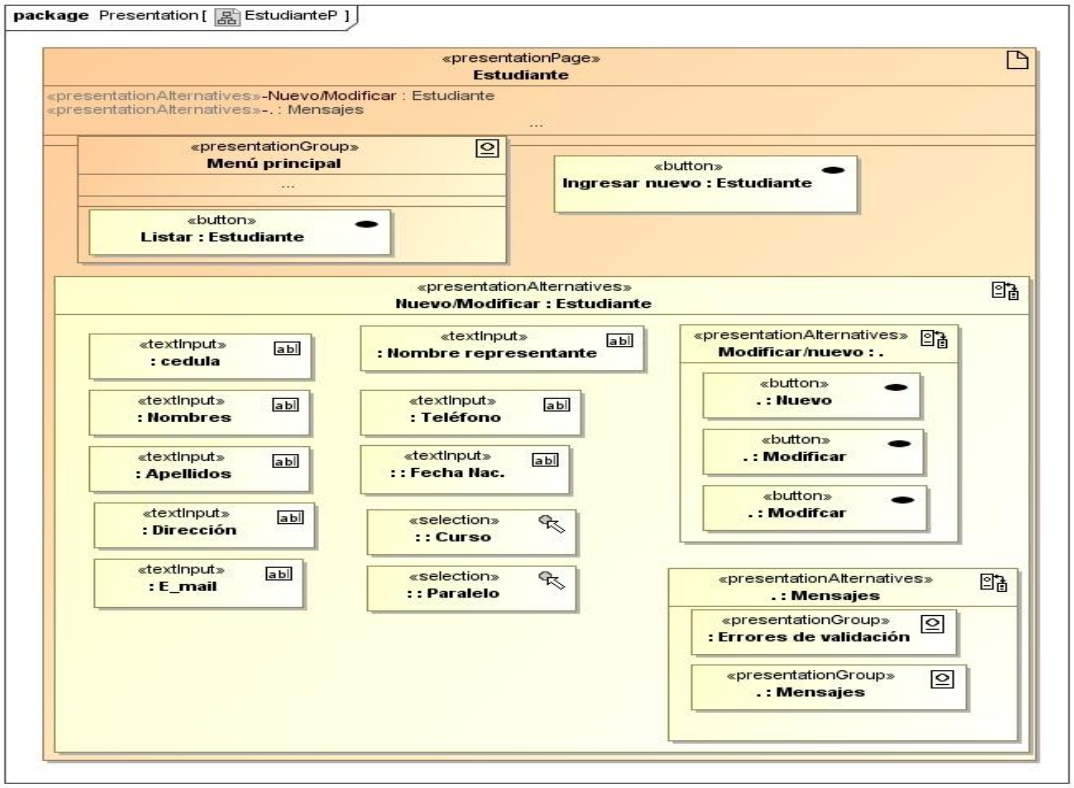

Figura 3. 37. Diagrama de presentación para configuración de la clase estudiante

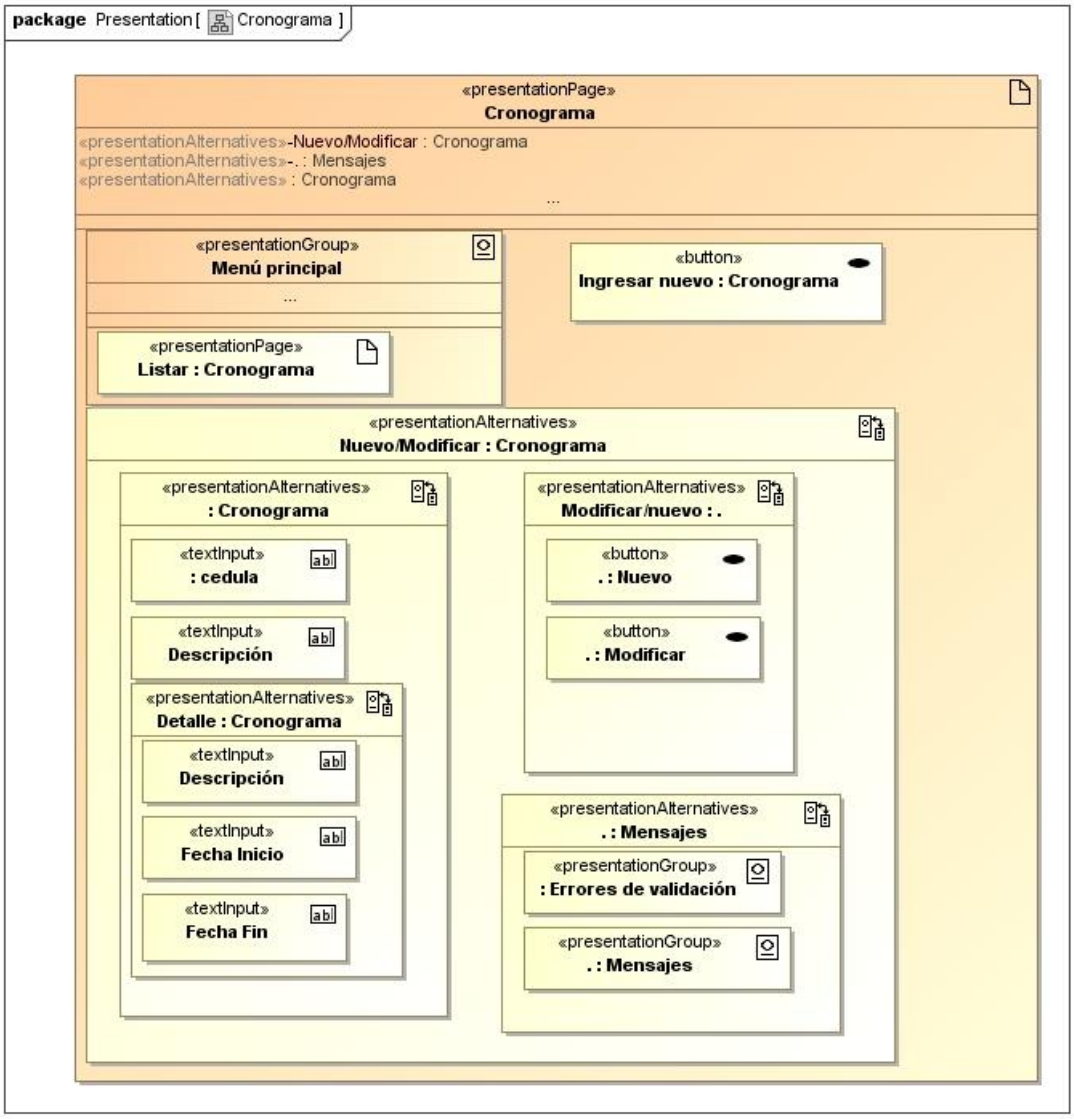

Figura 3. 38. Diagrama de presentación para la clase cronograma

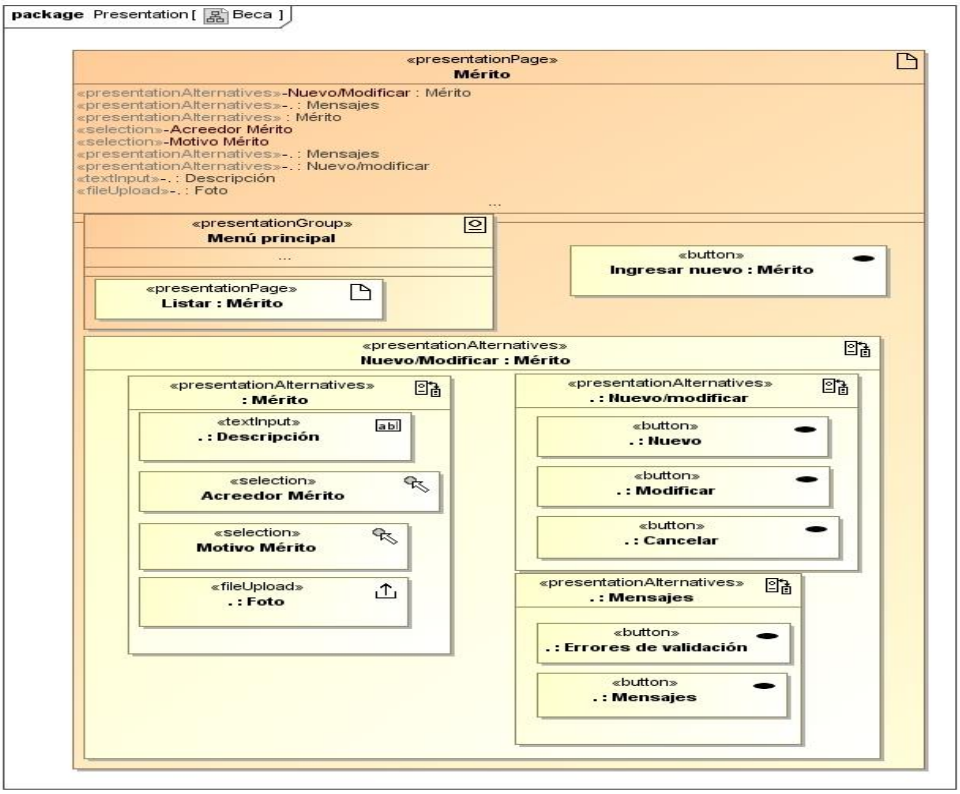

Figura 3. 39. Diagrama de presentación de la clase mérito

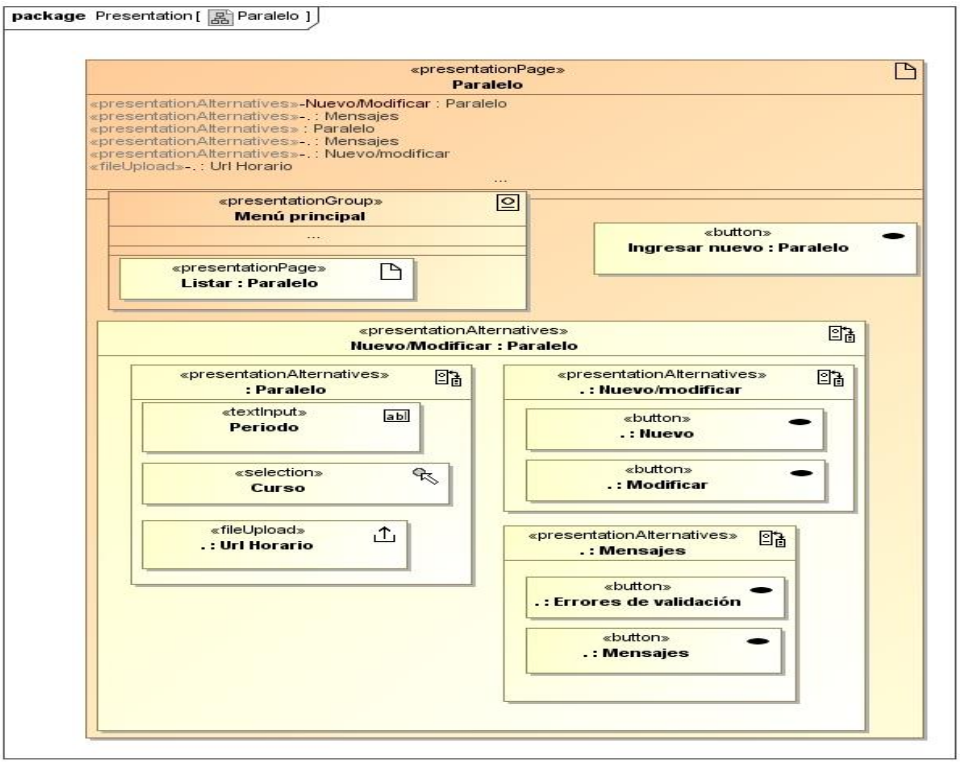

Figura 3. 40. Diagrama de presentación de la clase Paralelo

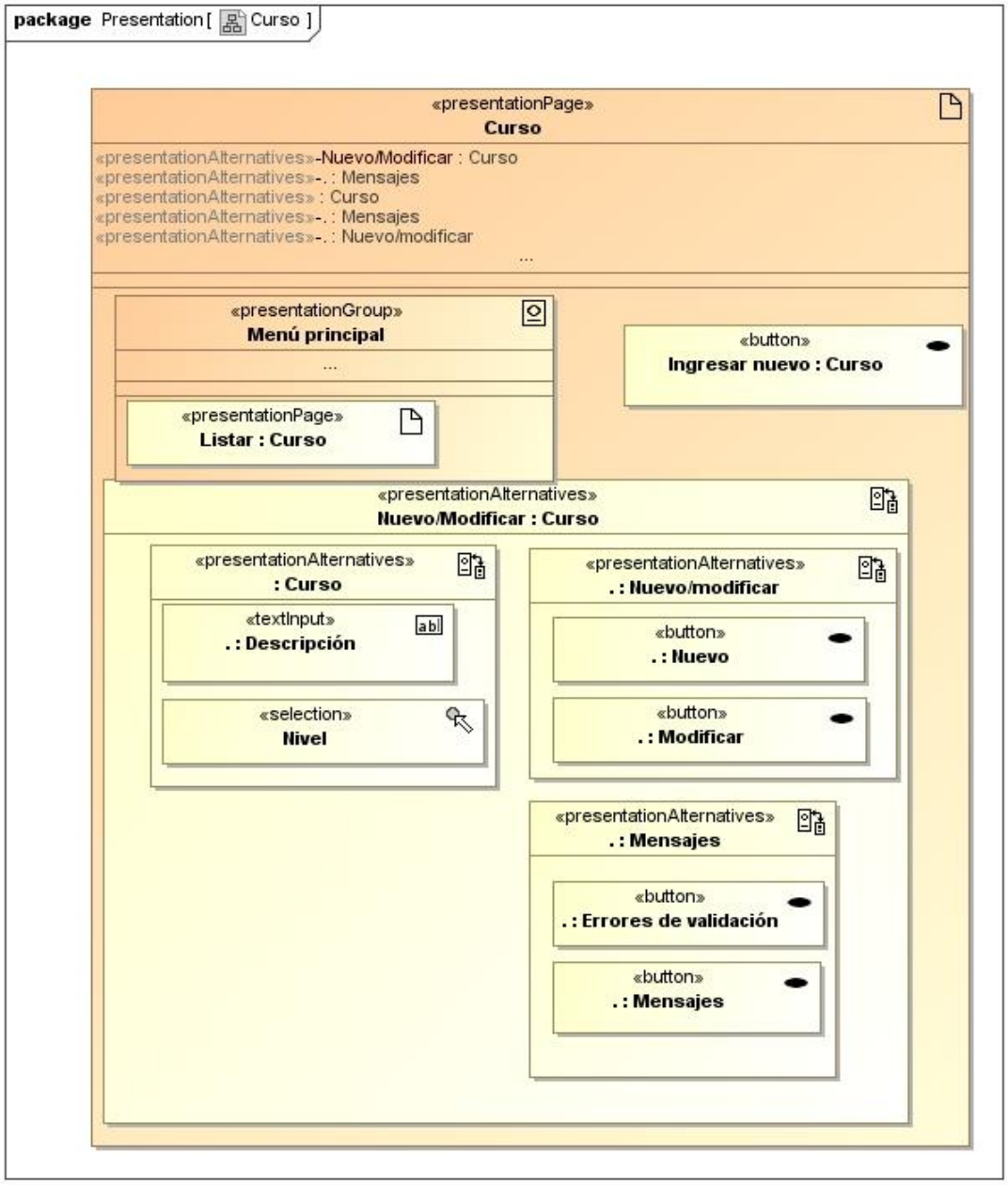

Figura 3. 41. Diagrama de presentación para la clase Curso

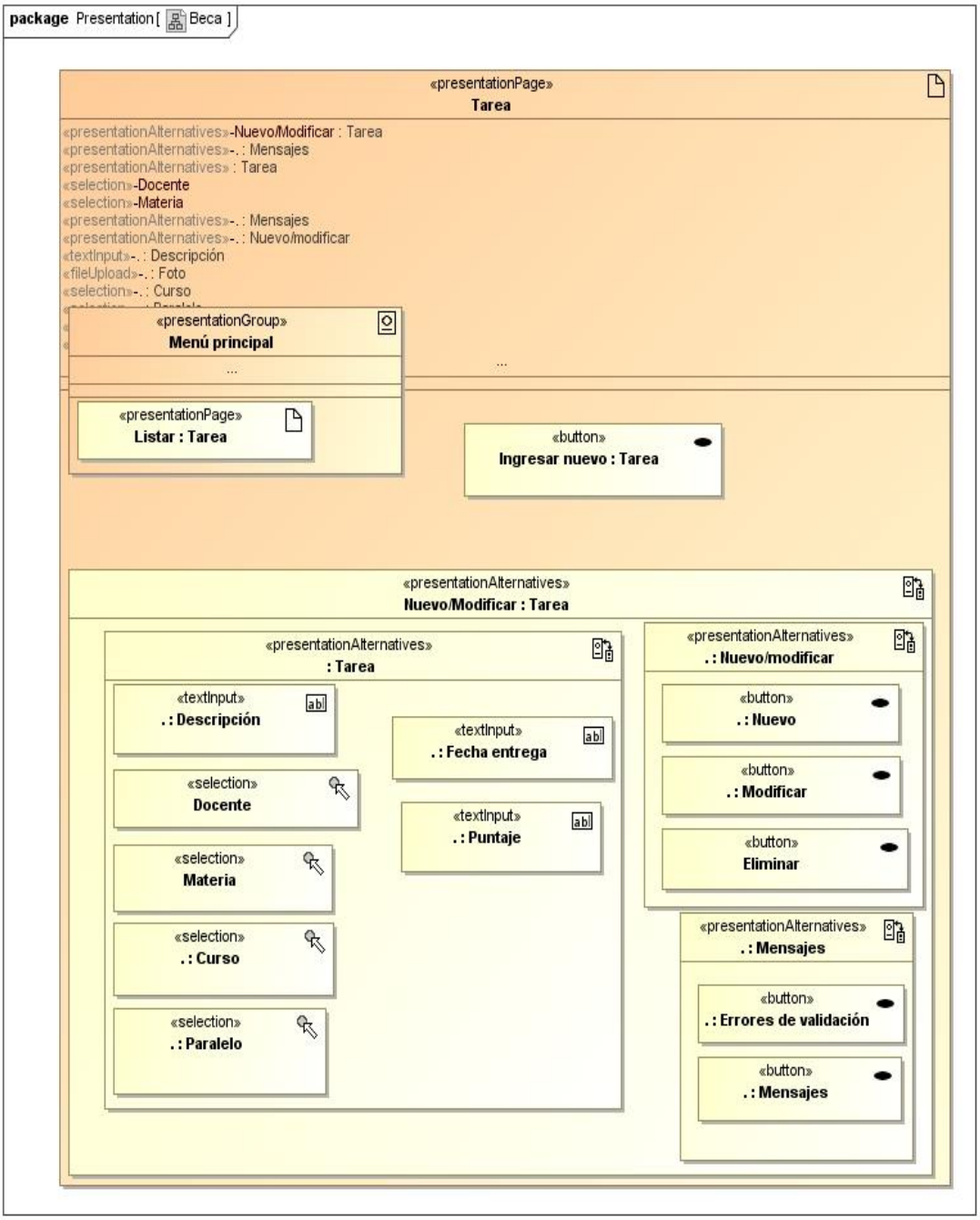

Figura 3. 42. Diagrama de presentación para la clase Tarea

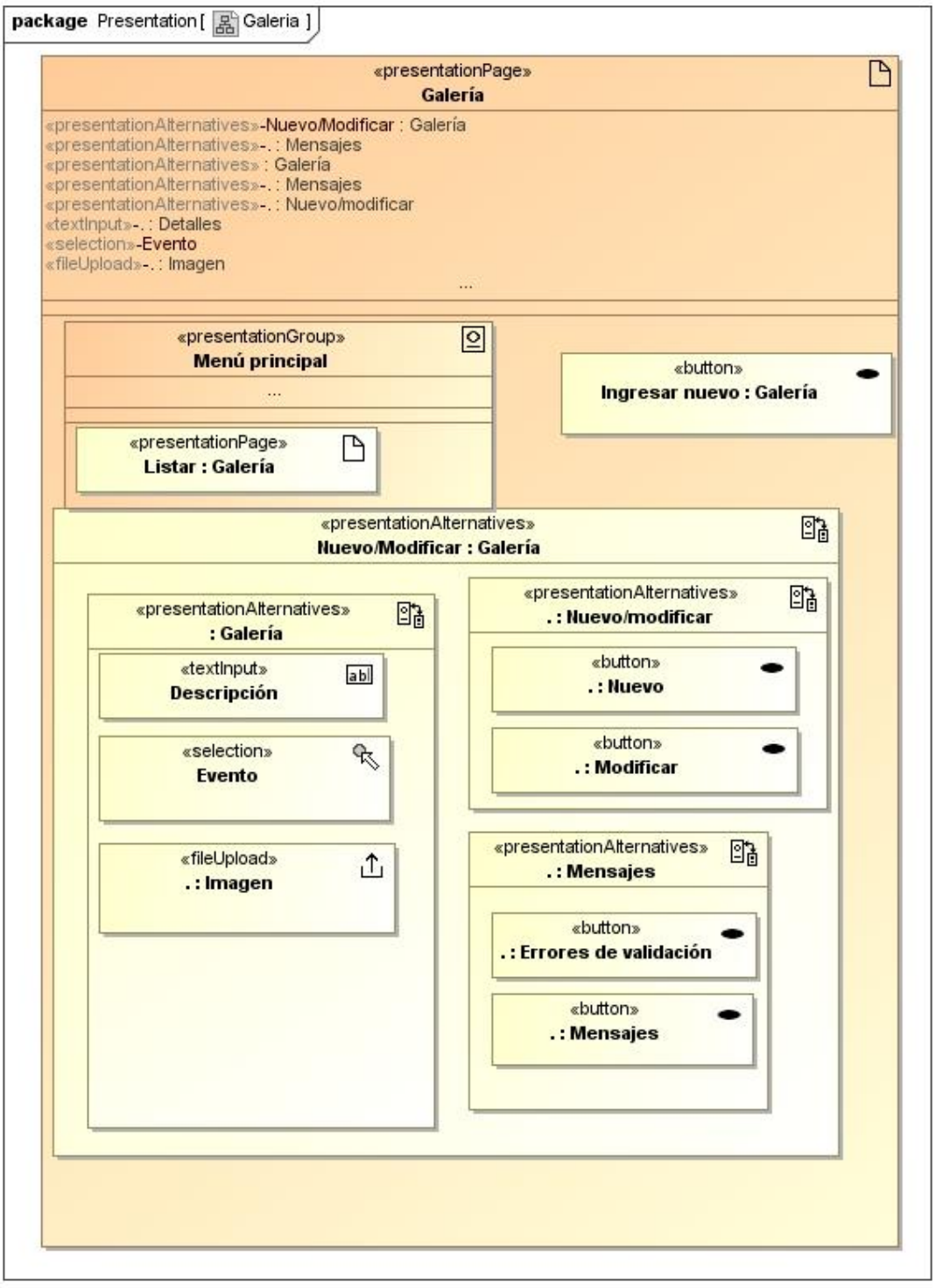

Figura 3. 43. Diagrama de presentación para la clase Galería

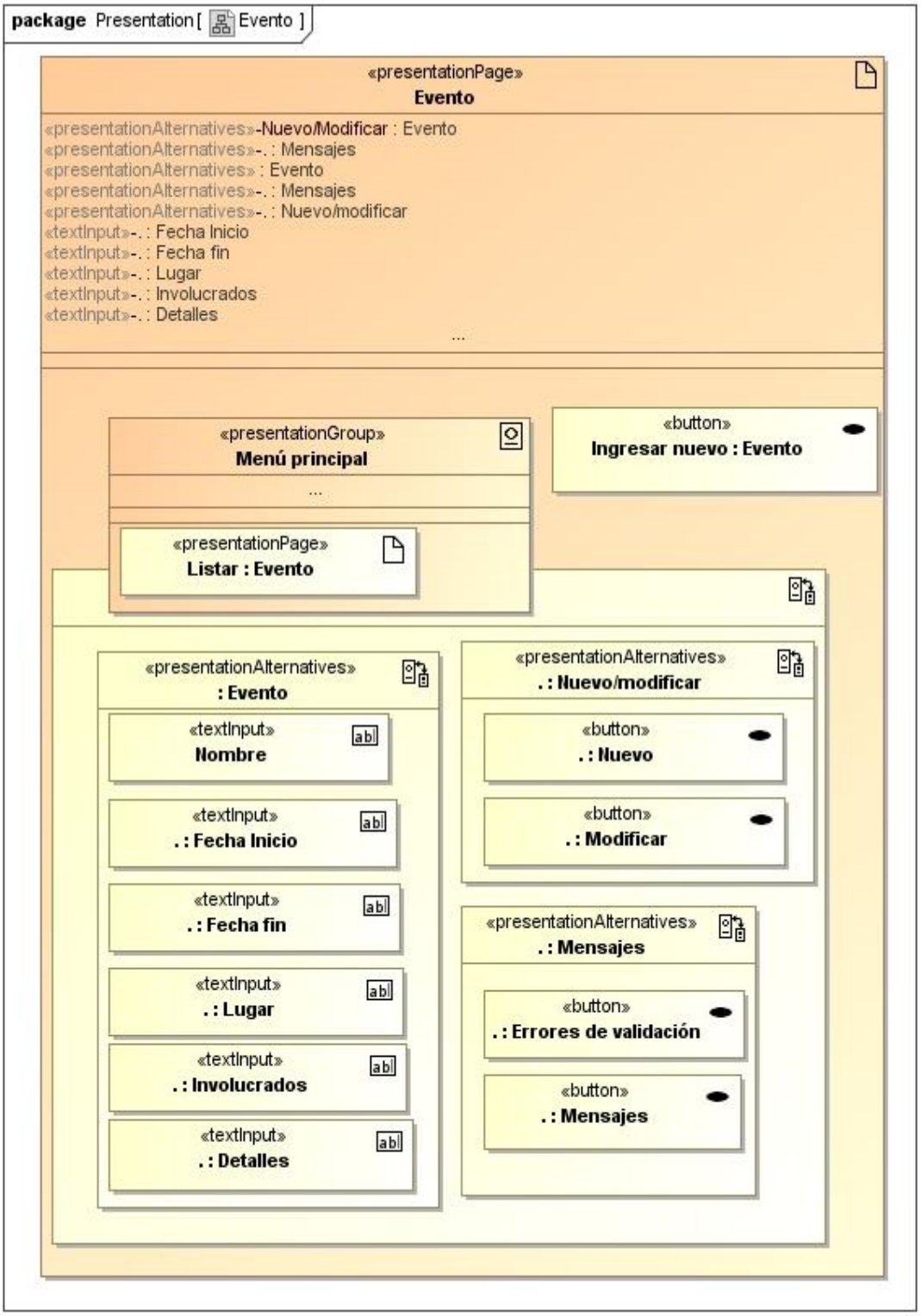

Figura 3. 44. Diagrama de presentación para la clase evento

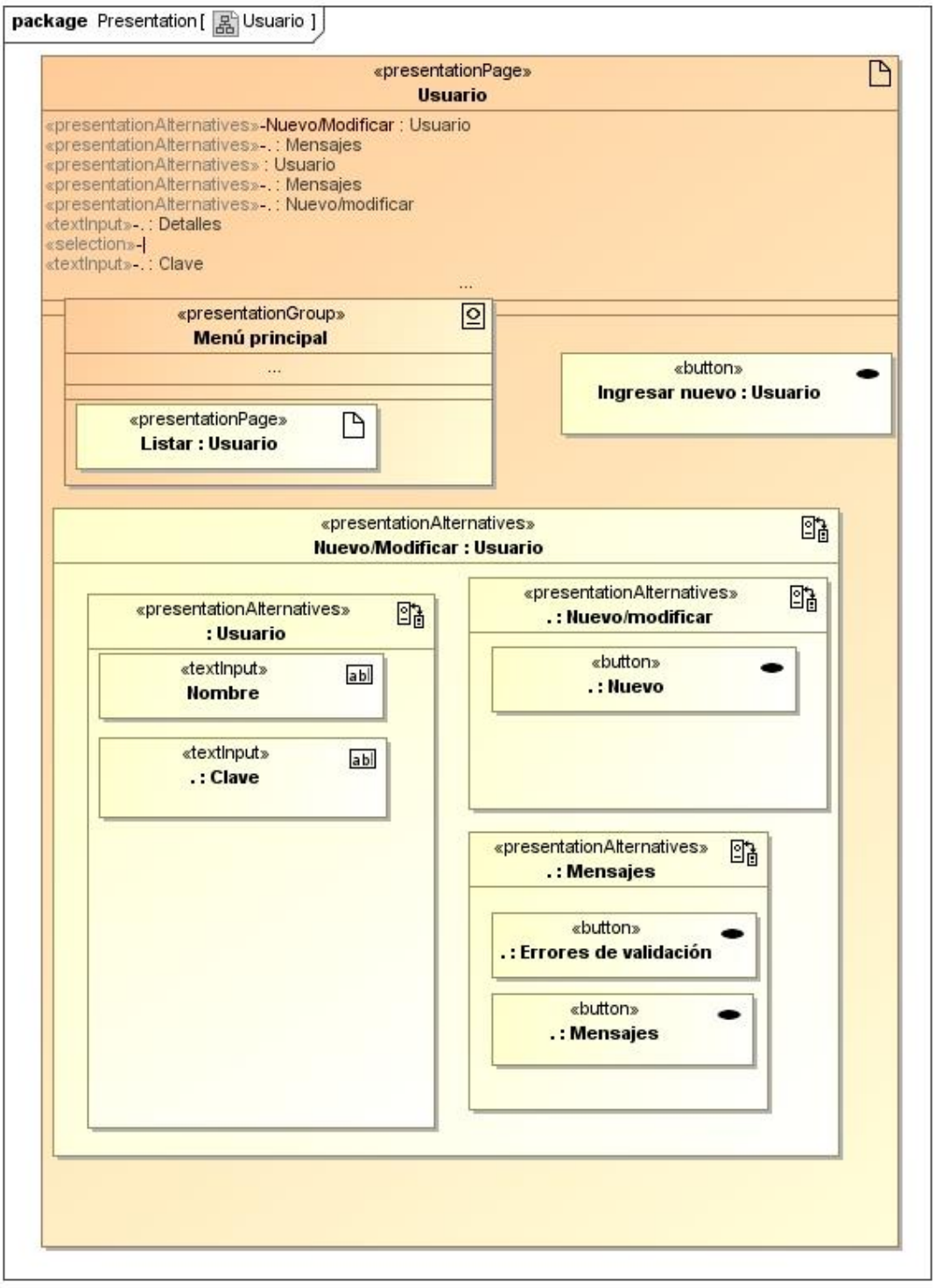

Figura 3. 45. Diagrama de presentación para la clase usuario

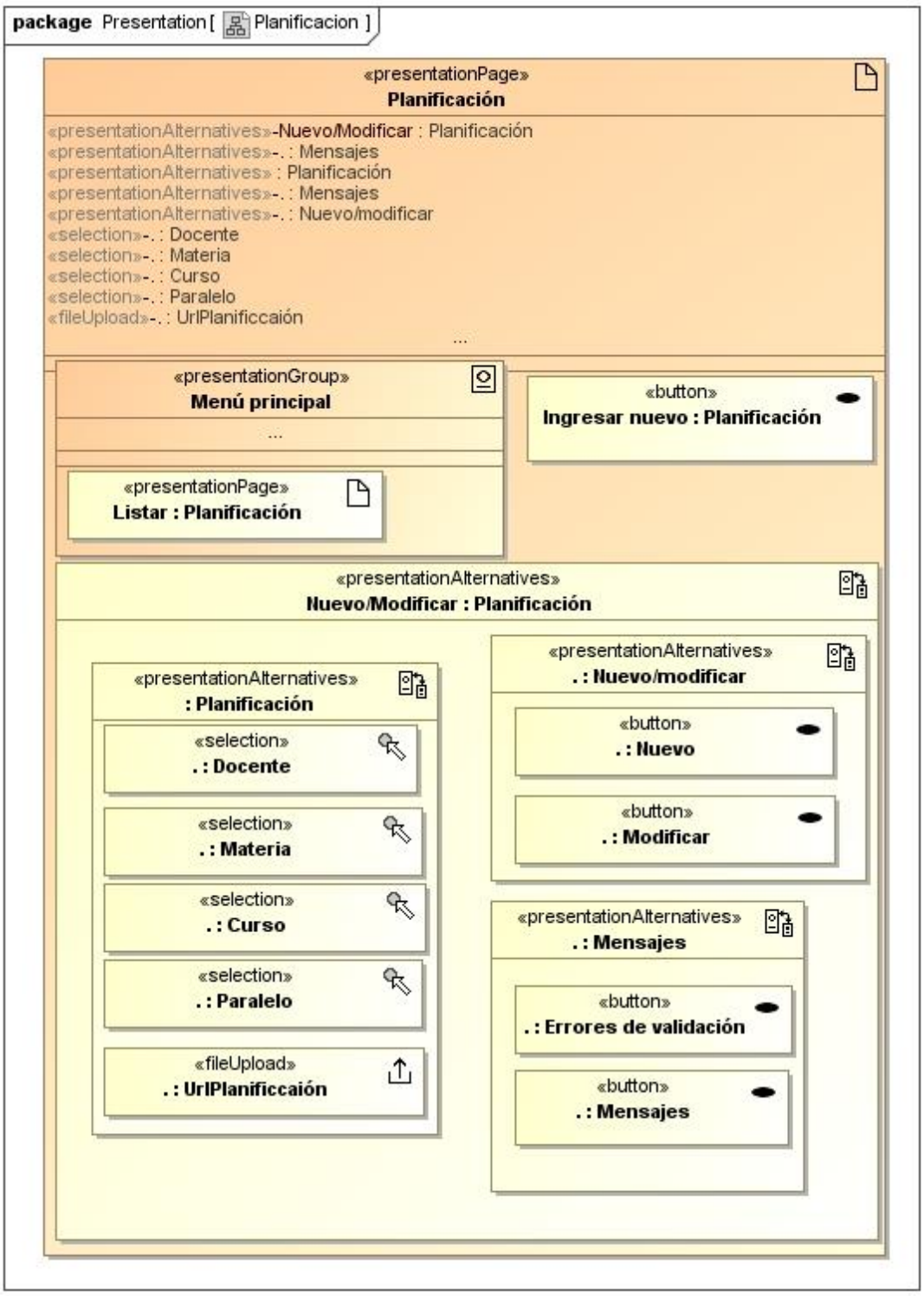

Figura 3. 46. Diagrama de presentación para la clase Planificación

d) Modelo de procesos

En este modelo se definen las acciones que realizan las clases de proceso (operacionales) especificadas en el modelo de navegación; el modelo de Proceso se divide en dos partes, la primera el Modelo de Estructura de Procesos, en el cual se incluyen las relaciones entre las clases de proceso y la segunda parte es el Modelo de Flujo de Procesos, en el que se incluyen las actividades relacionadas con cada proceso, describiendo el comportamiento interno de cada clase proceso.

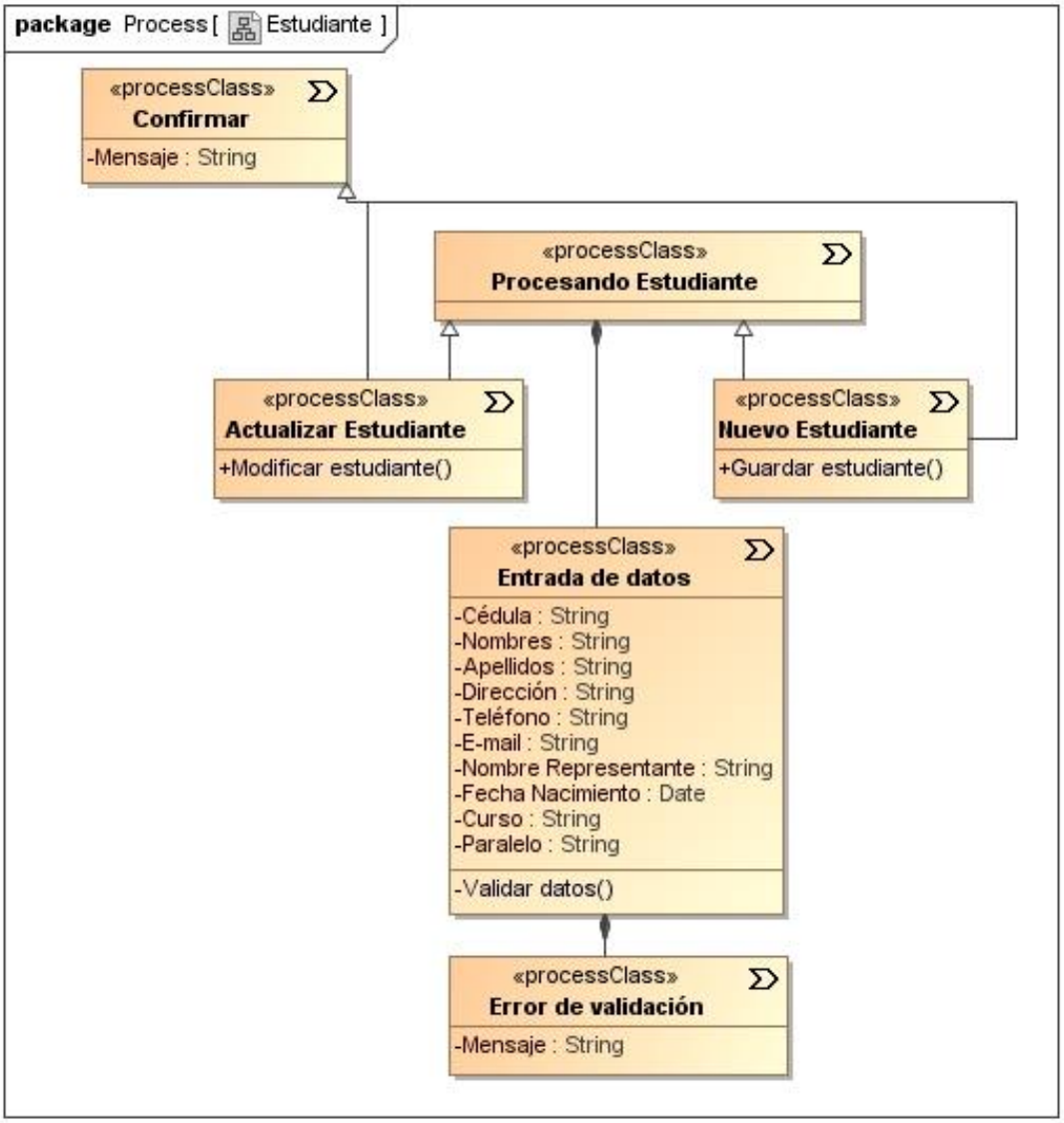

Figura 3. 47. Diagrama del Modelo de estructura de procesos para la clase Estudiante

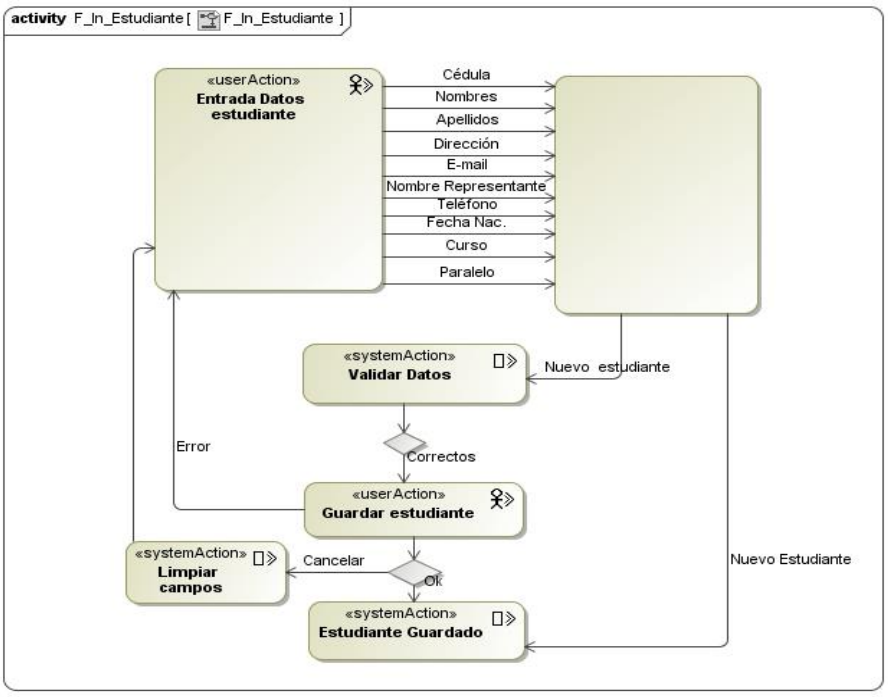

Figura 3. 48. Diagrama del Modelo de flujo de procesos para el ingreso de nuevo Estudiante

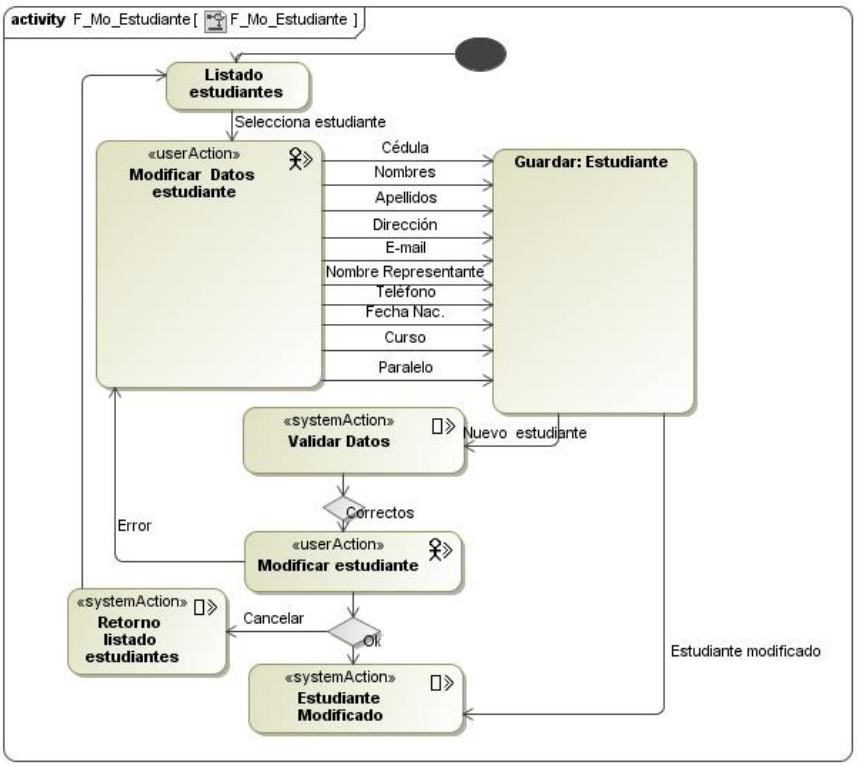

Figura 3. 49. Diagrama del Modelo de flujo de procesos para la modificación de Estudiante

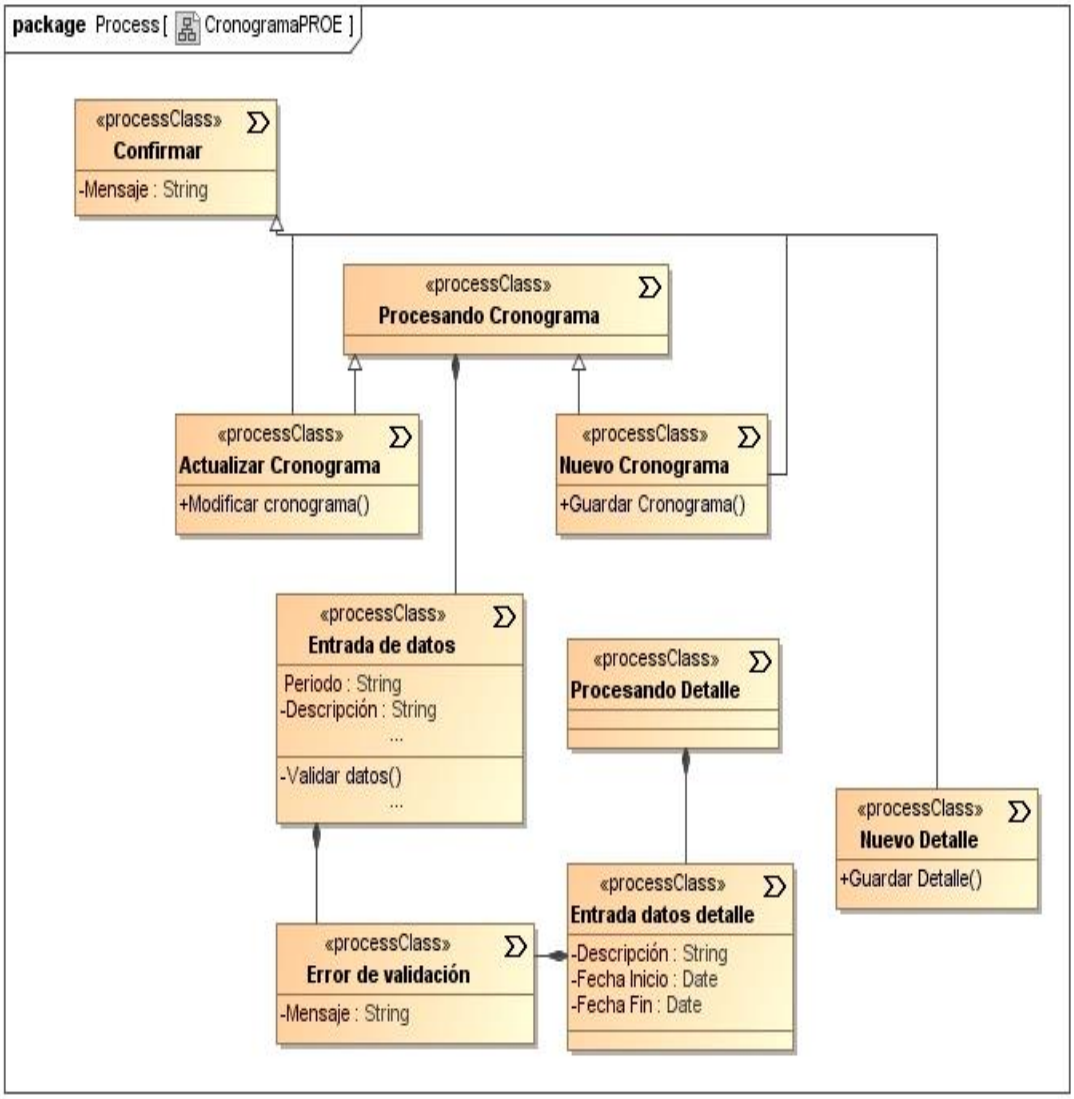

Figura 3. 50. Diagrama del Modelo de estructura de procesos para la clase Cronograma

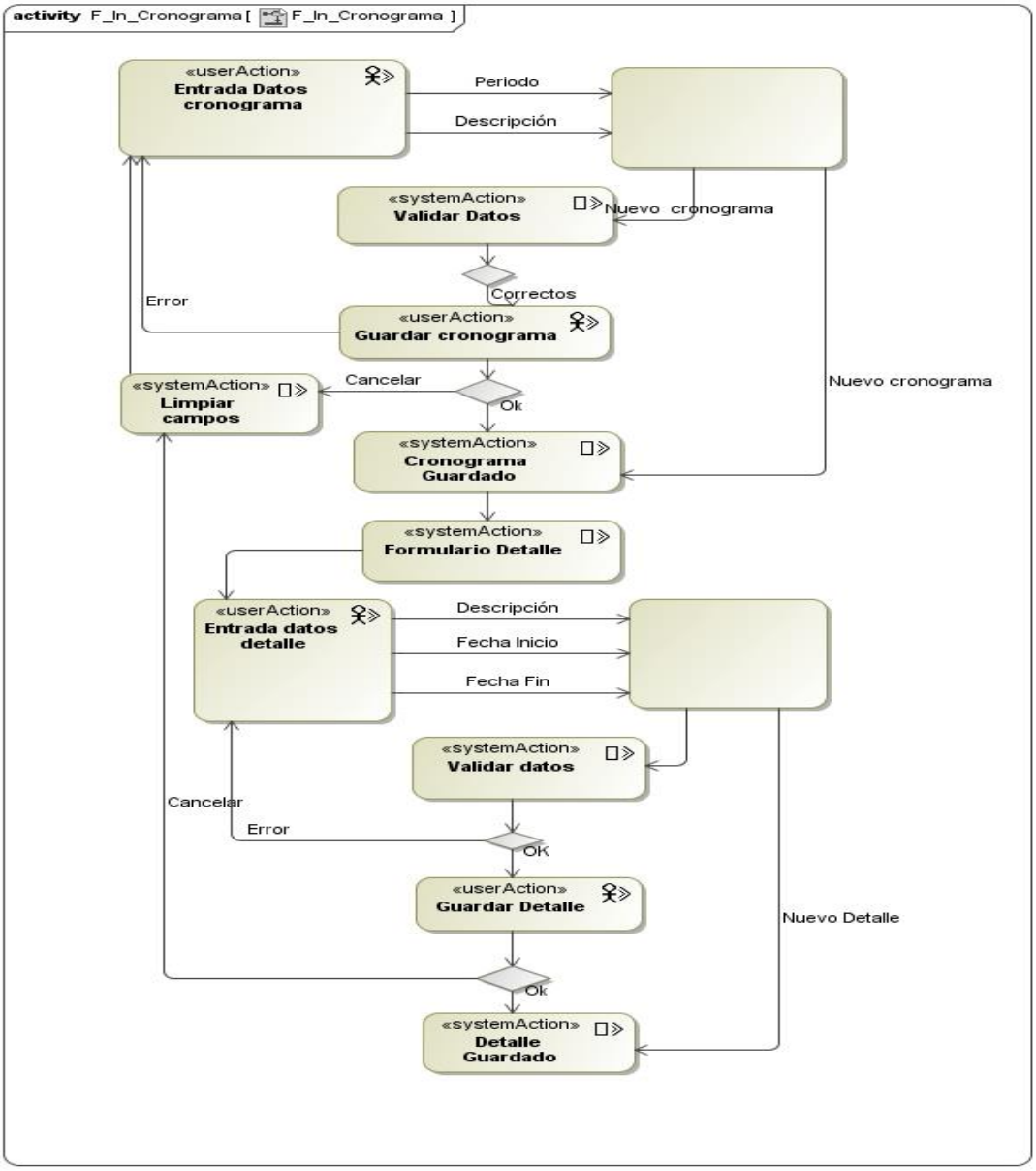

Figura 3. 51. Diagrama del Modelo de flujo de procesos para el ingreso de nuevo cronograma

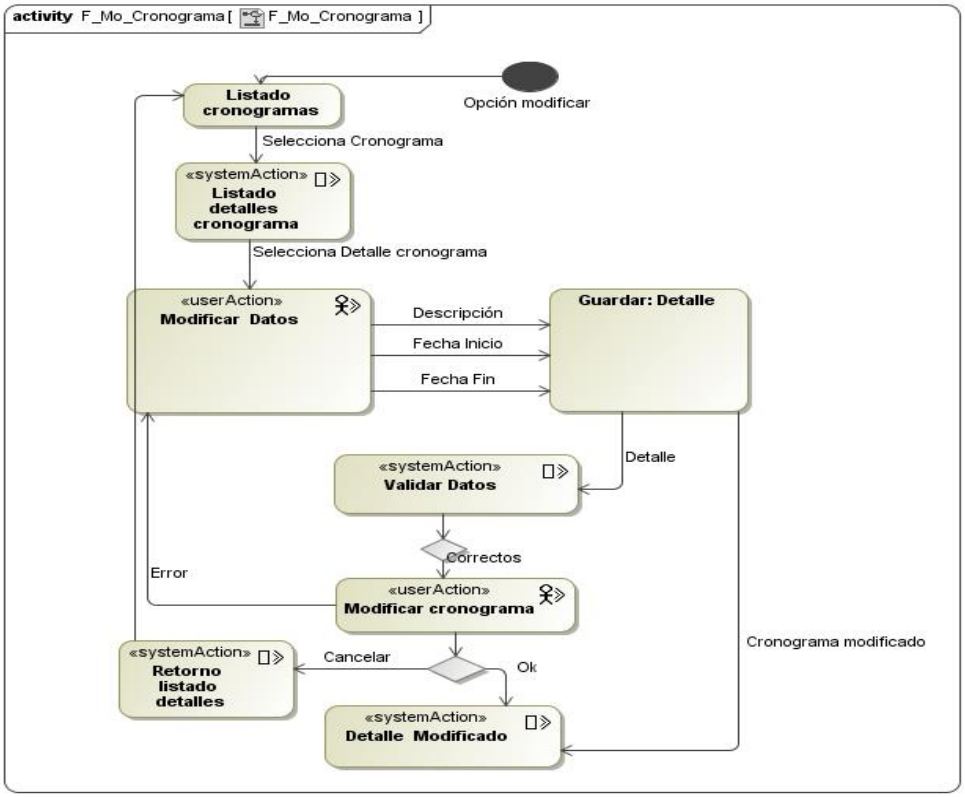

Figura 3. 52. Diagrama del Modelo de flujo para modificación

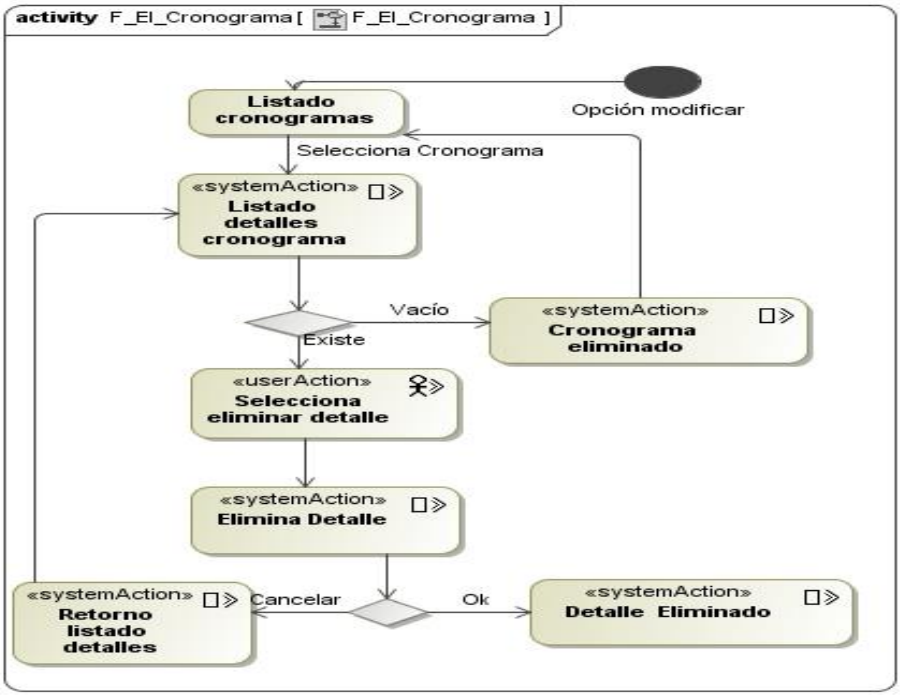

Figura 3. 53. Diagrama del Modelo de flujo de procesos para la eliminación de cronogramas

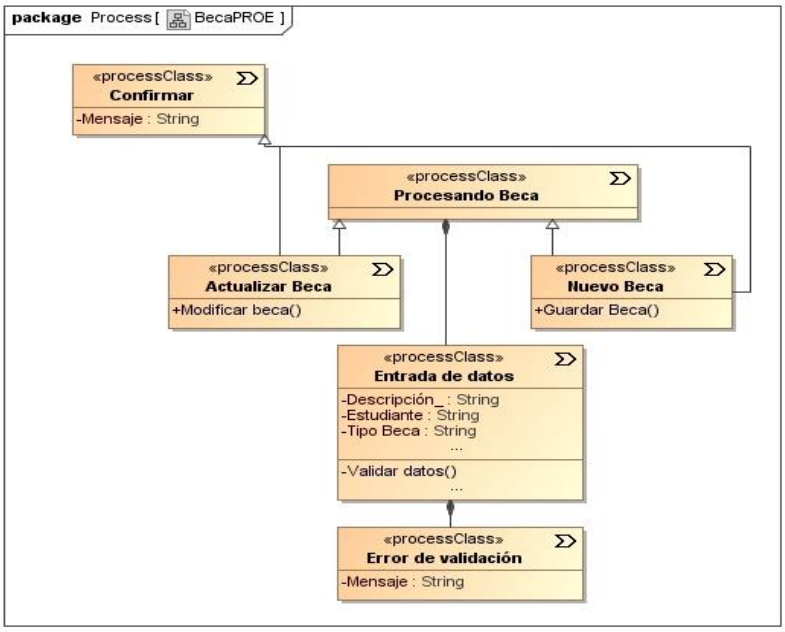

Figura 3. 54. Diagrama del Modelo de estructura de procesos para la clase Beca

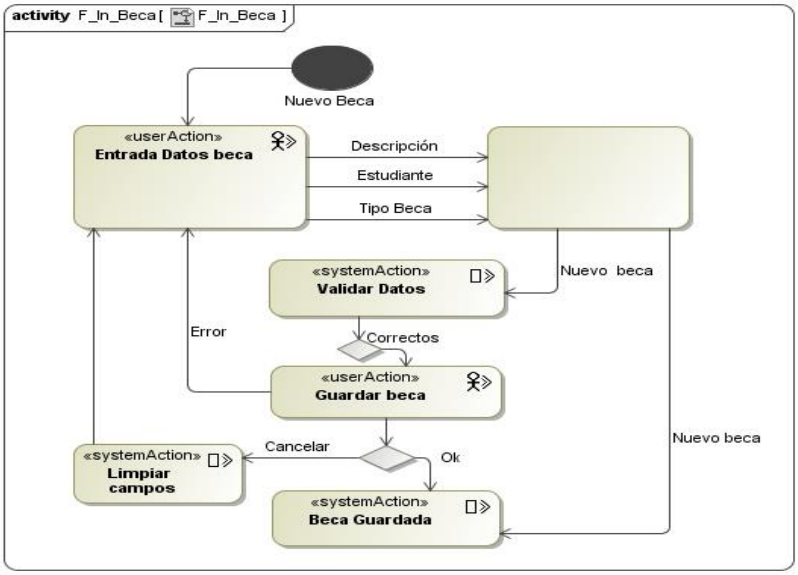

Figura 3. 55. Diagrama del Modelo de flujo de procesos para el ingreso de nueva beca

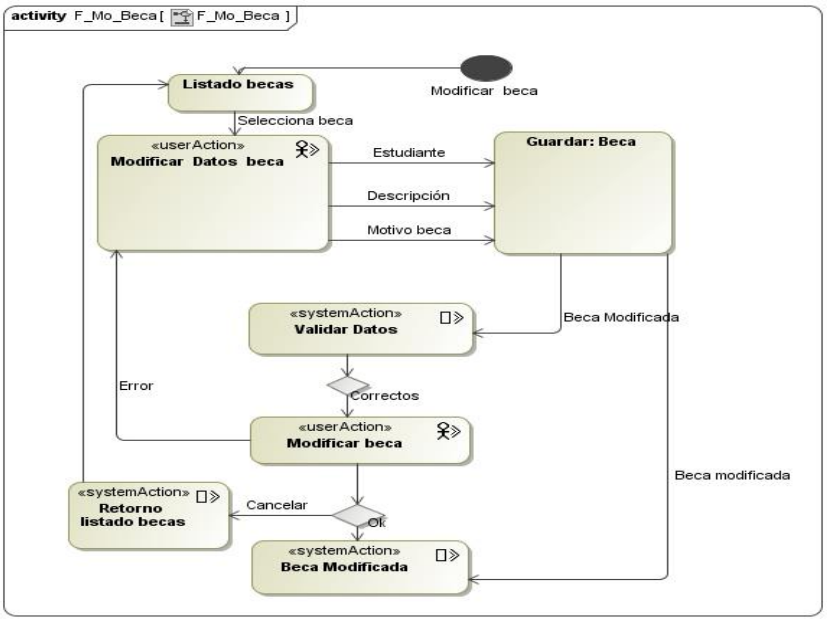

Figura 3. 56. Diagrama del Modelo de flujo de procesos para la modificación de beca

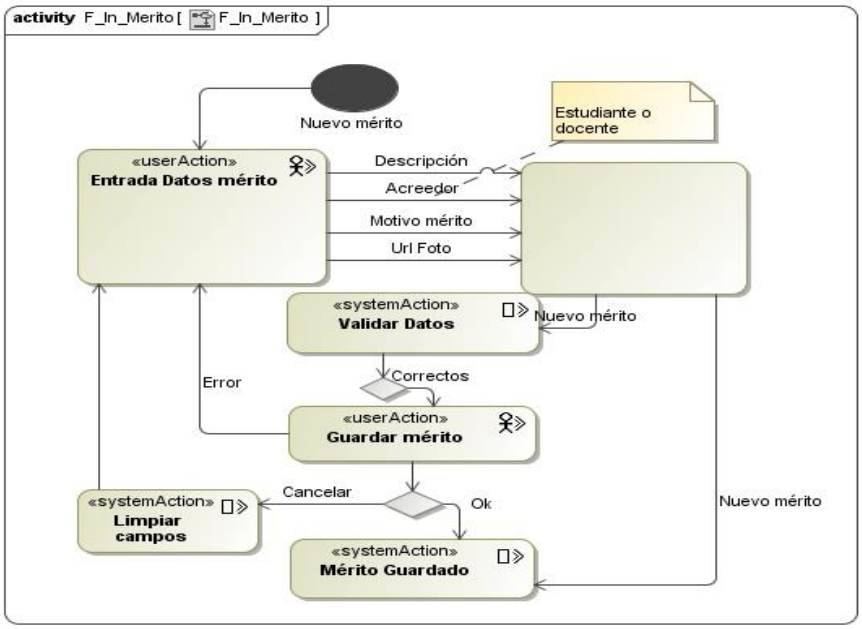

Figura 3. 57. Diagrama del Modelo de estructura de procesos para la clase mérito

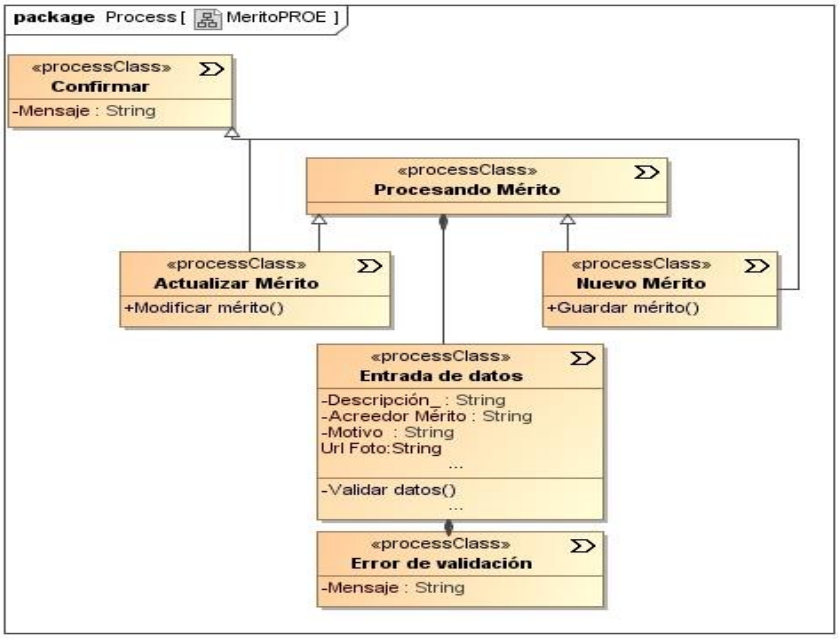

Figura 3. 58. Diagrama del Modelo de flujo de procesos para el ingreso de méritos

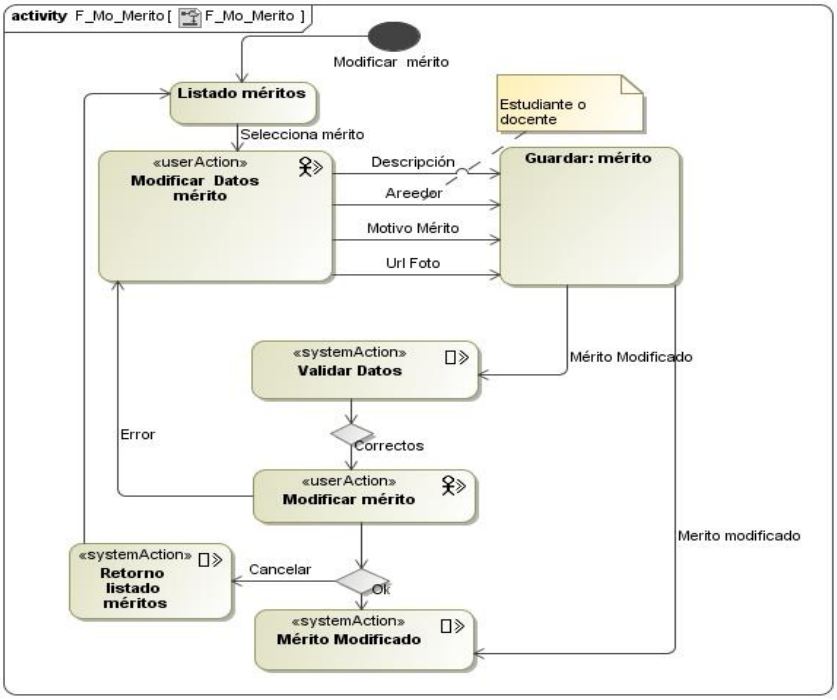

Figura 3. 59. Diagrama del Modelo de flujo de procesos para la modificación de méritos

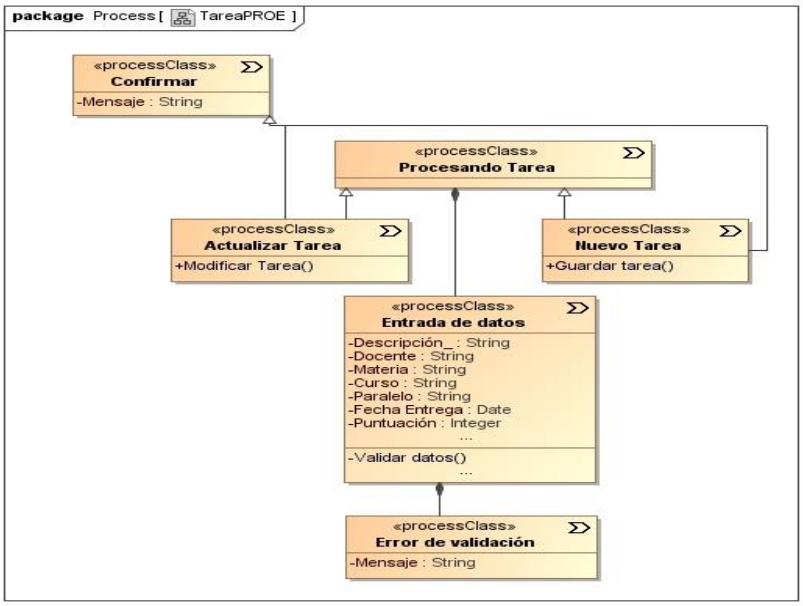

Figura 3. 60. Diagrama del Modelo de estructura de procesos para la clase tarea

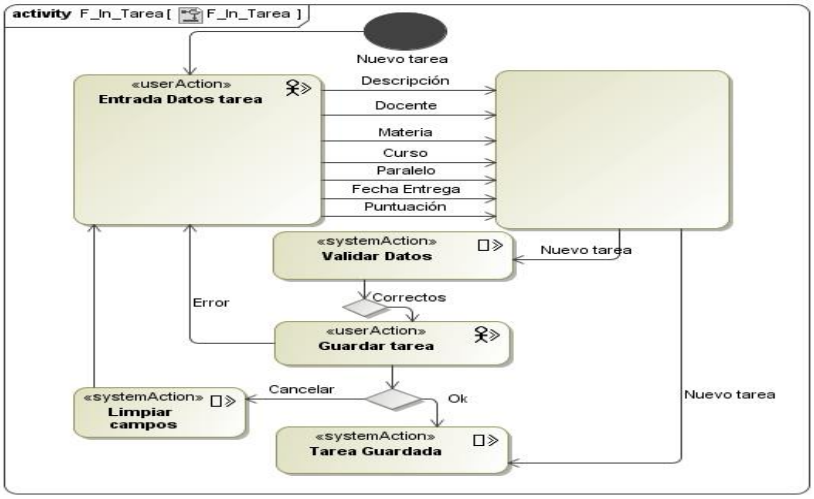

Figura 3. 61. Diagrama del Modelo de flujo para ingreso de tareas

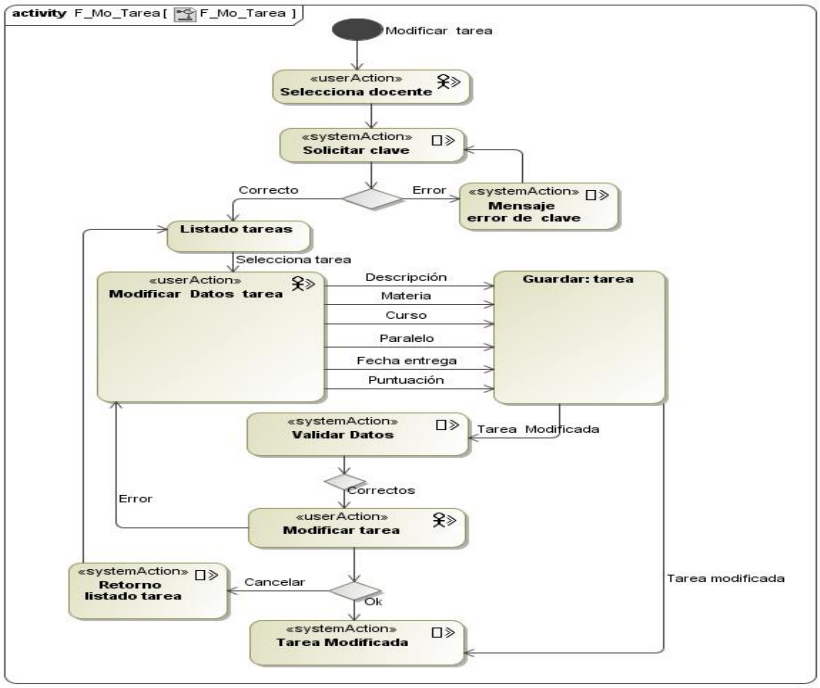

Figura 3. 62. Diagrama del Modelo de flujo de procesos para el ingreso de tareas

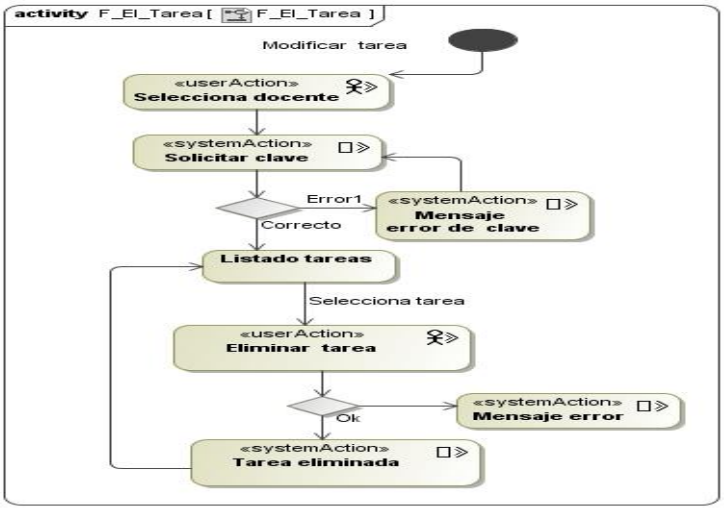

Figura 3. 63. Diagrama del Modelo de flujo de procesos para la eliminación de tareas

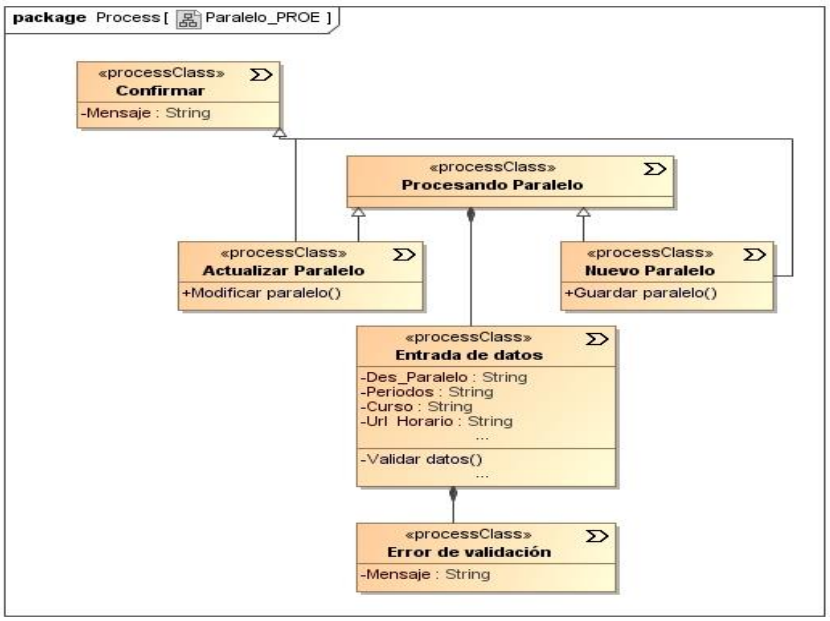

Figura 3. 64. Diagrama del Modelo de estructura de procesos para la clase paralelo

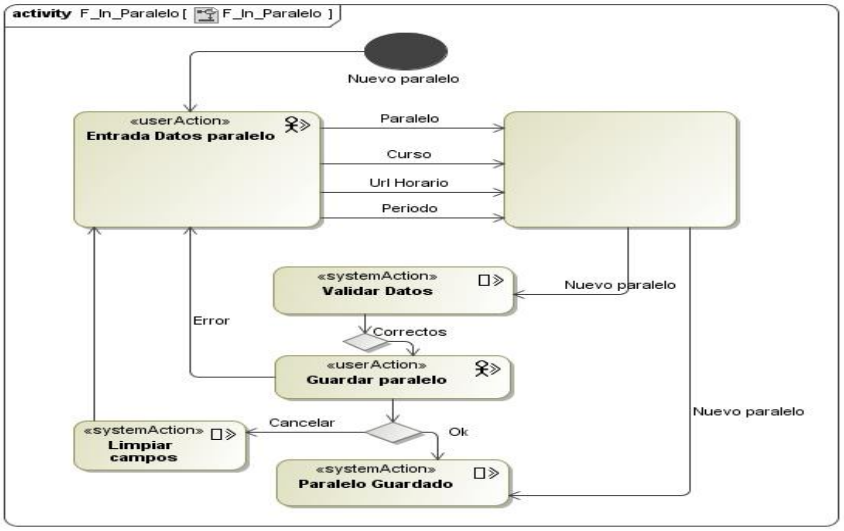

Figura 3. 65. Diagrama del Modelo de flujo de procesos para el ingreso de nuevos paralelos

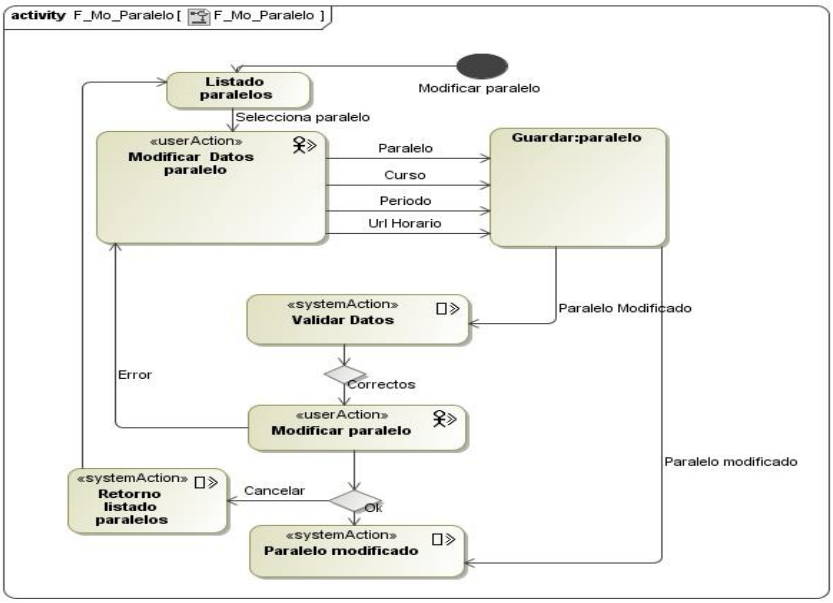

Figura 3. 66. Diagrama del Modelo de flujo de procesos para la modificación paralelos

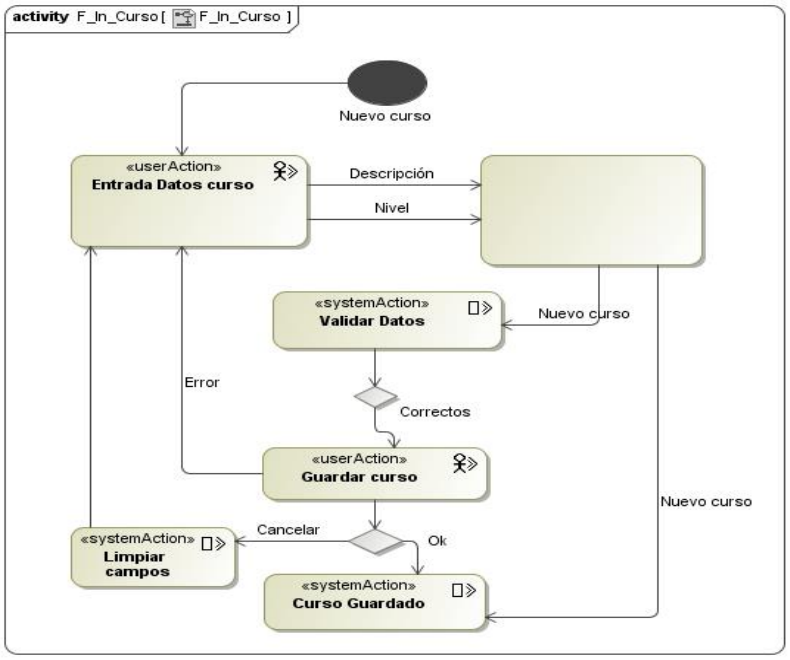

Figura 3. 67. Diagrama del Modelo de estructura de procesos para la clase curso

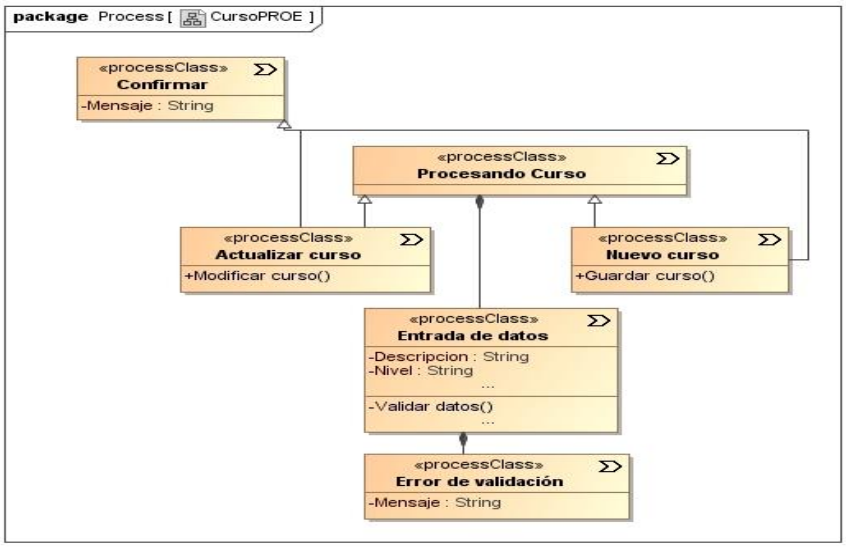

Figura 3. 68. Diagrama del Modelo de flujo de procesos para el ingreso de curso

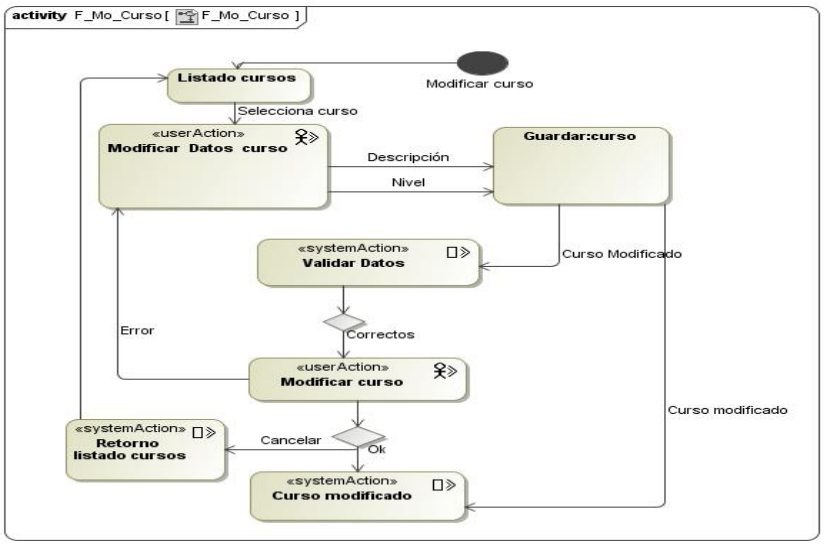

Figura 3. 69. Diagrama del Modelo de flujo de procesos para la modificación de curso

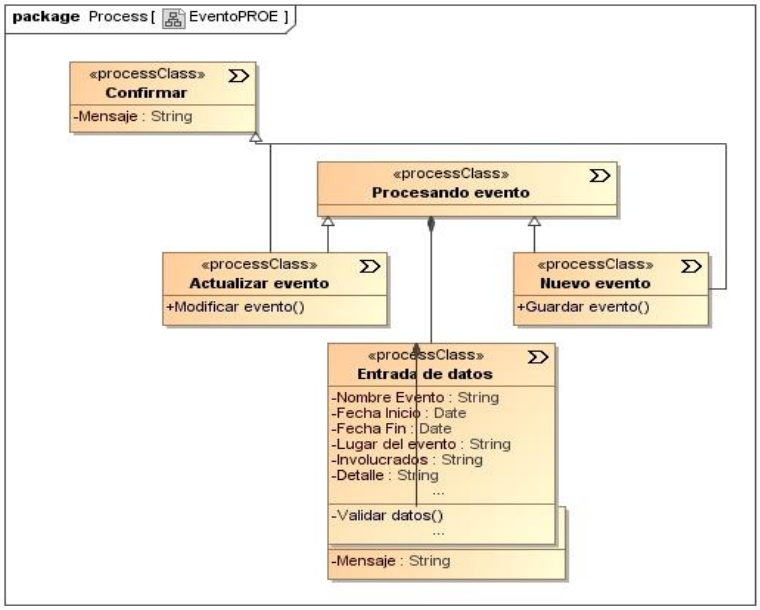

Figura 3. 70. Diagrama del Modelo de estructura de procesos para la clase evento

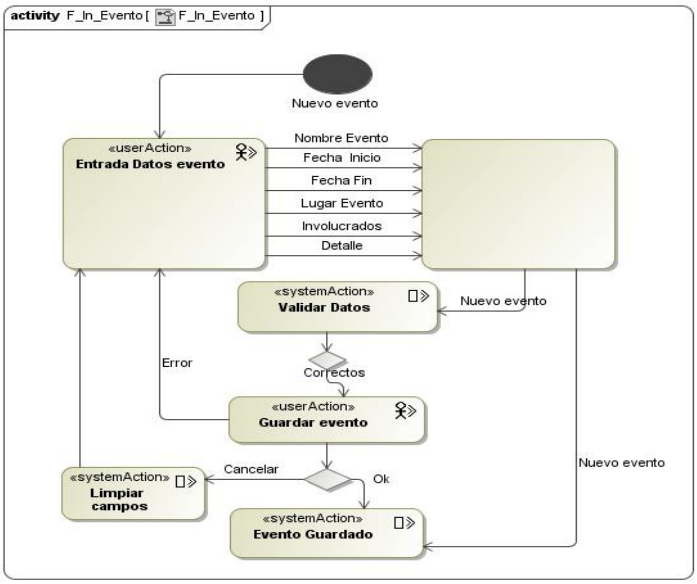

Figura 3. 71. Diagrama del Modelo de flujo de procesos para el ingreso de evento

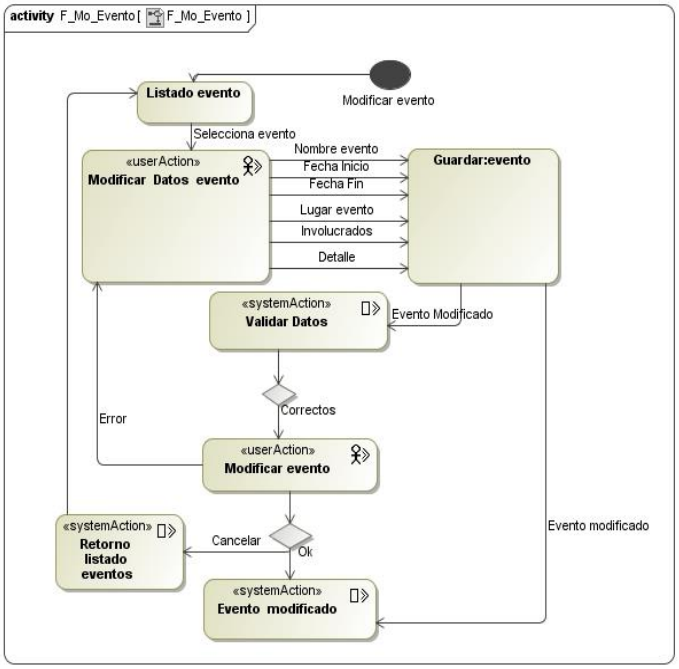

Figura 3. 72. Diagrama del Modelo de flujo de procesos para la modificación de evento

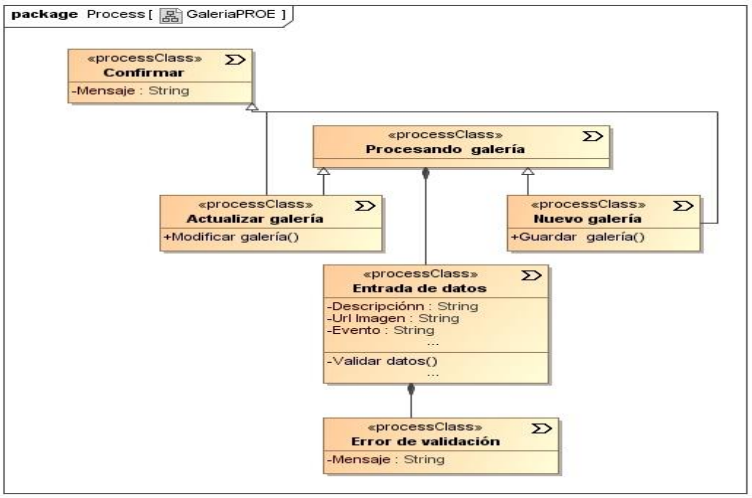

Figura 3. 73. Diagrama del Modelo de estructura de procesos para la clase galería

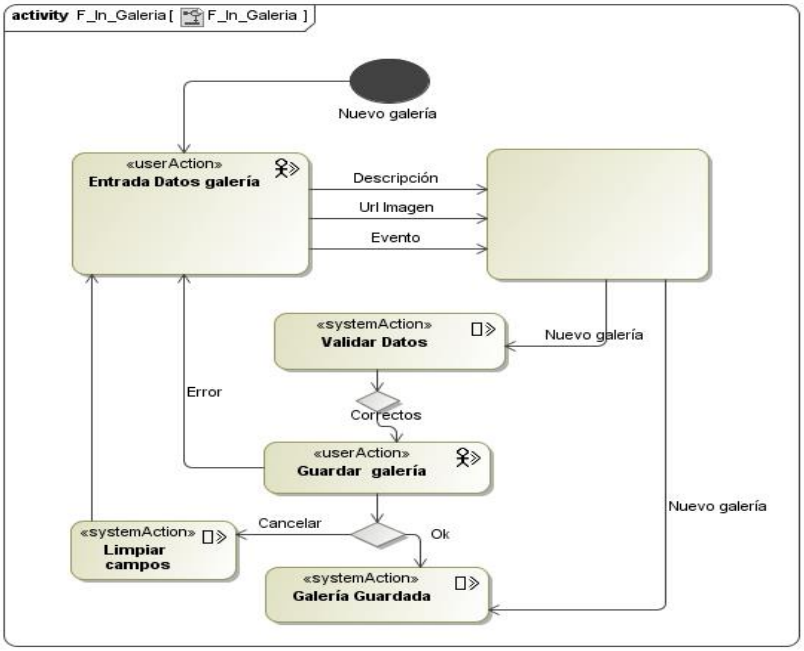

Figura 3. 74. Diagrama del Modelo de flujo de procesos para el ingreso de galería

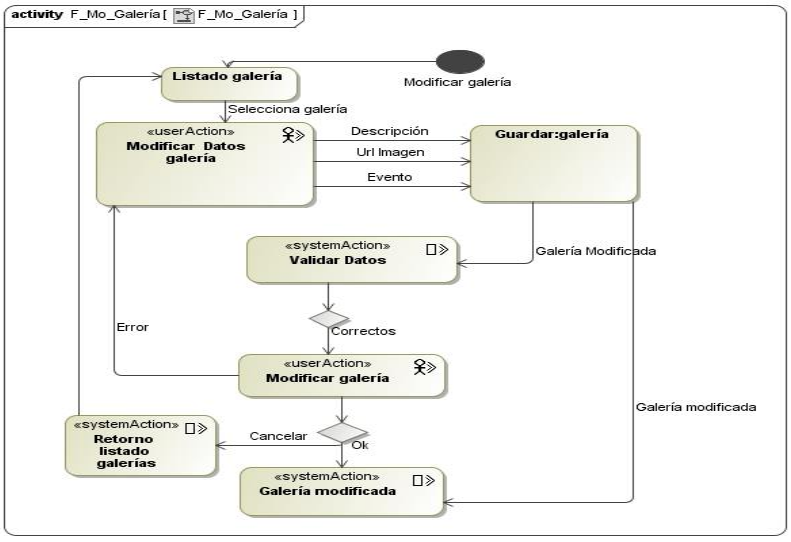

Figura 3. 75. Diagrama del Modelo de flujo de procesos para la modificación de galería

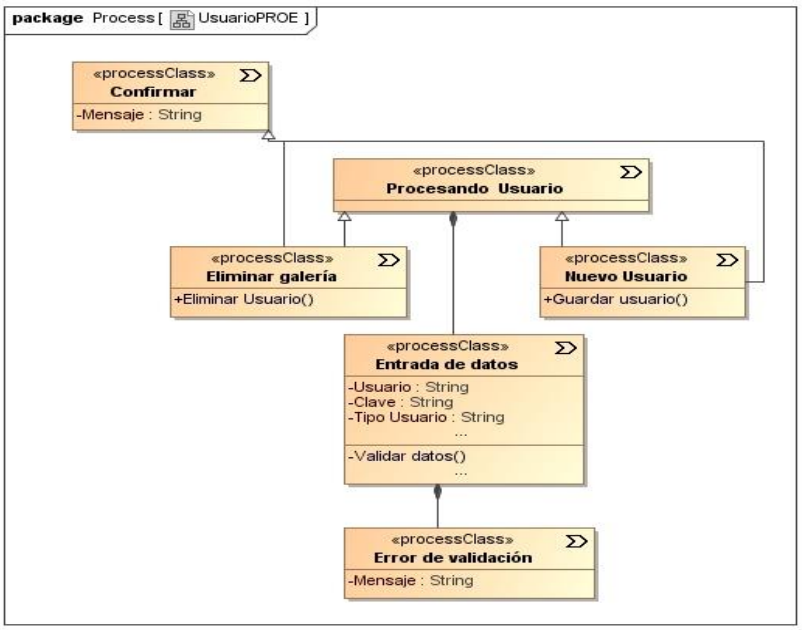

Figura 3. 76. Diagrama del Modelo de estructura de procesos para la clase usuario

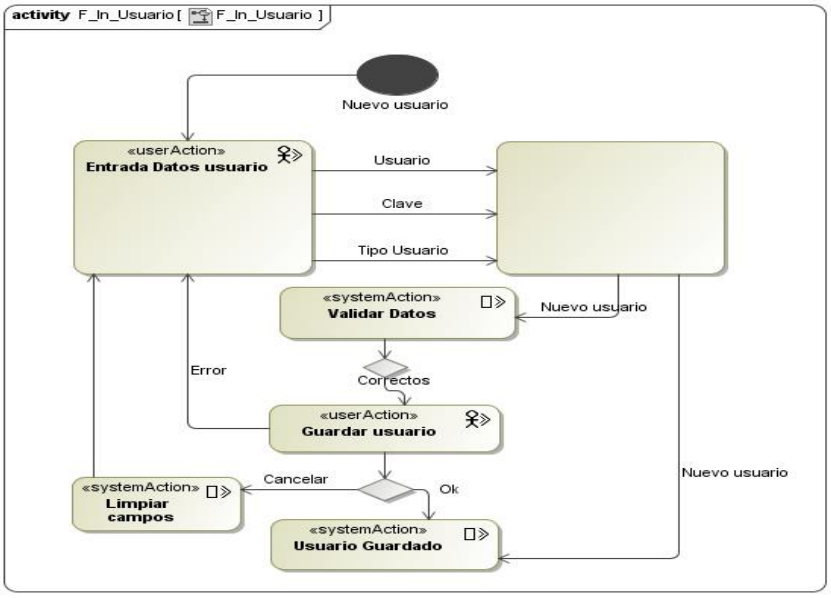

Figura 3. 77. Diagrama del Modelo de flujo de procesos para el ingreso de usuarios

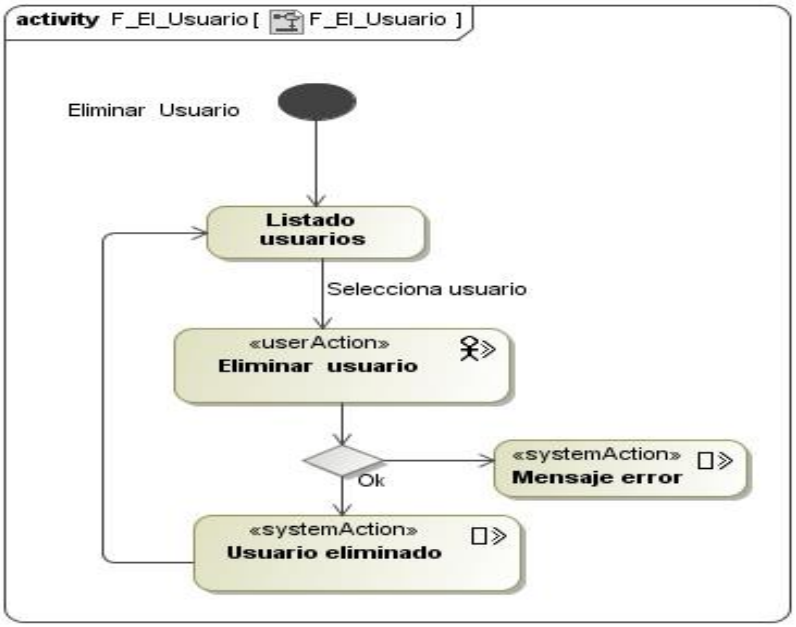

Figura 3. 78. Modelo de flujo de eliminación de usuario

Modelo físico

Este modelo queda definido por el Modelo Físico de Datos del Sistema, que es el resultado final de la evolución del Análisis, este modelo físico se implementa en un motor de base de datos SQL Server 2008 R2

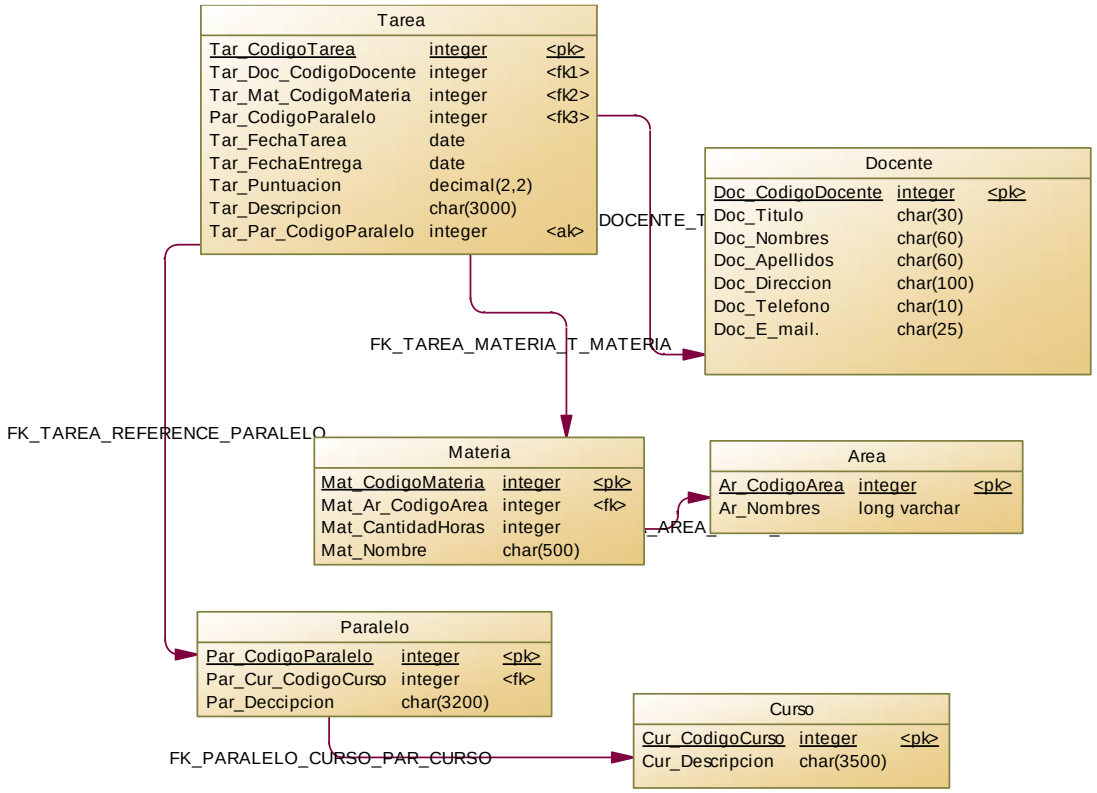

Figura 3. 79. Modelo físico de datos del sistema POWELIOX, gestión tarea

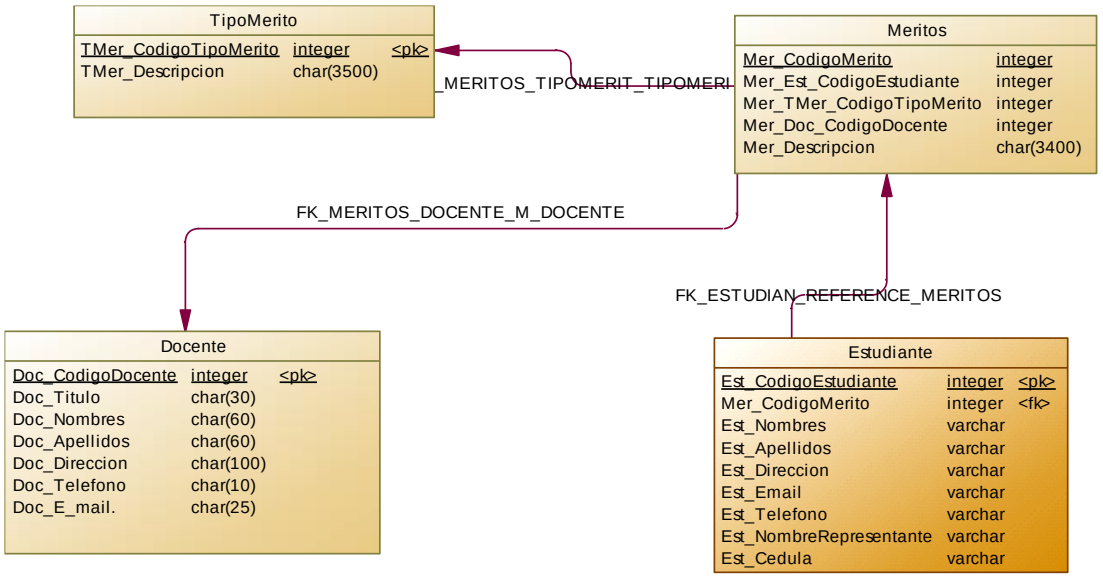

Figura 3. 80. Modelo físico de datos del sistema POWELIOX, gestión méritos

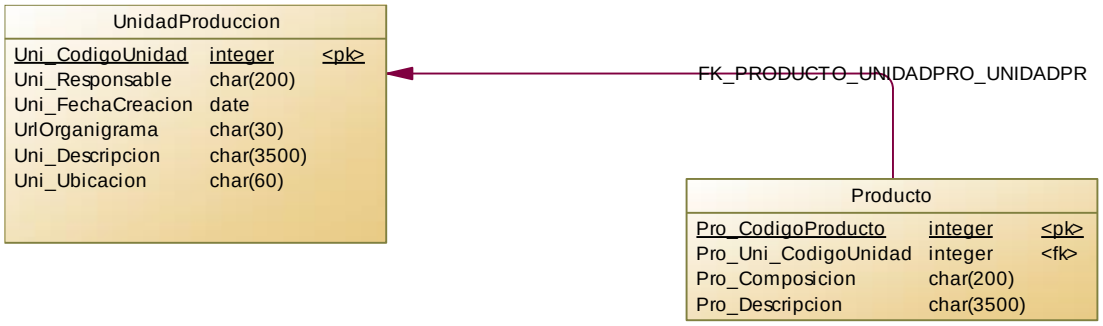

Figura 3. 81. Modelo físico de datos del sistema POWELIOX, gestión productos

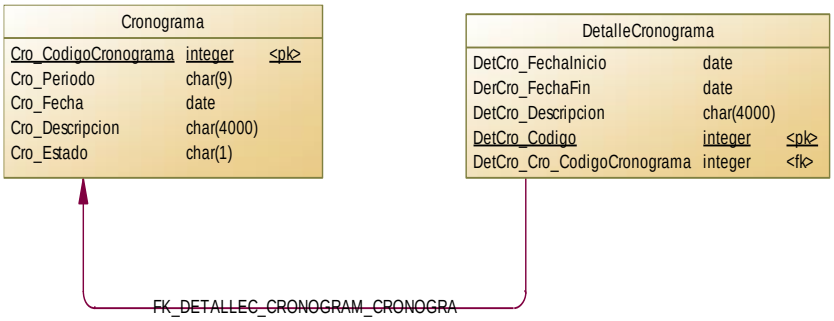

Figura 3. 82. Modelo físico de datos del sistema POWELIOX, gestión cronogramas

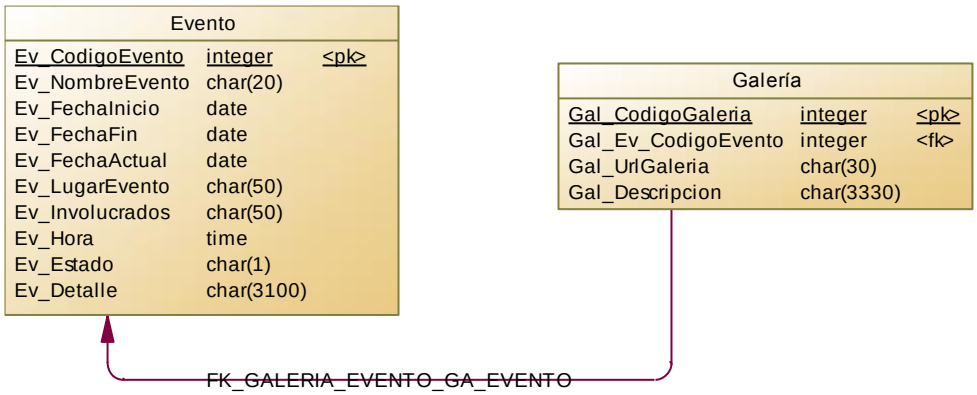

Figura 3. 83. Modelo físico de datos del sistema POWELIOX, gestión galerías

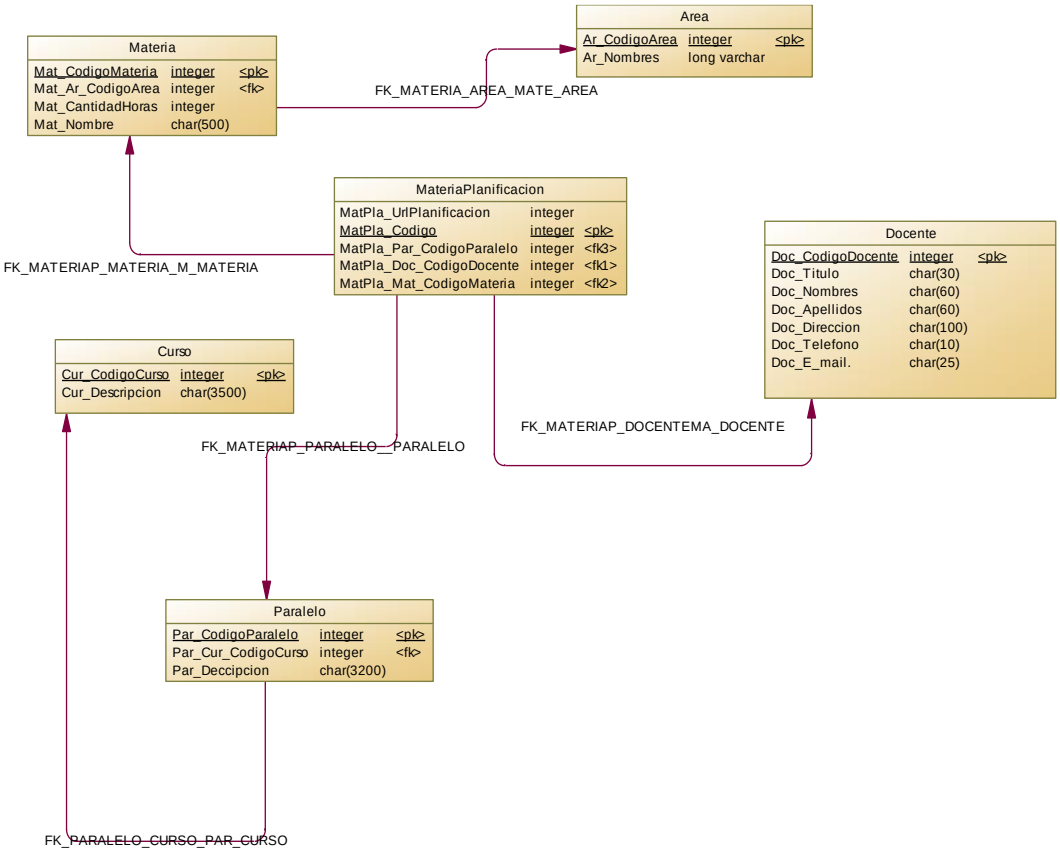

Figura 3. 84. Modelo físico de datos del sistema POWELIOX, gestión planificaciones

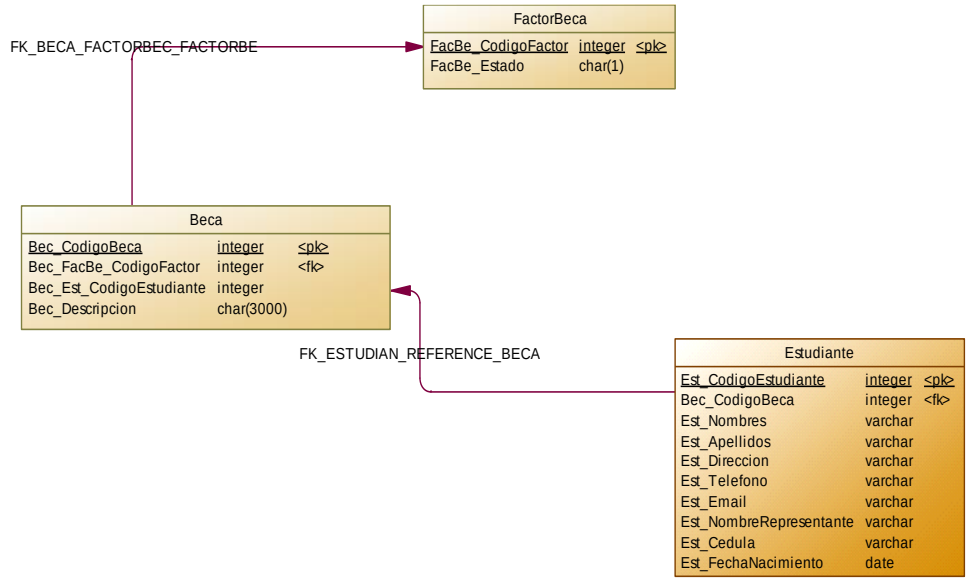

Figura 3. 85. Modelo físico de datos del sistema POWELIOX, gestión becas

RazgosGenerales		
RaGe_CodigoRazgos	integer	<pk>
RaGe_Vision	char(600)	
RaGe_Objetivo	char(600)	
RaGe_Ubicacion	char(200)	
RaGe_UrlHistoria	char(100)	
RaGe_NombreRector	char(100)	
RaGe_Telefono	varchar(9)	
RaGe_Email	varchar(50)	
RaGe_DescripcionExtensiones	varchar(500)	

Figura 3. 86. Modelo físico de datos del sistema POWELIOX, rasgos generales

3.5. Arquitectura POWELIOX

La arquitectura refleja un modelo de tres capas:

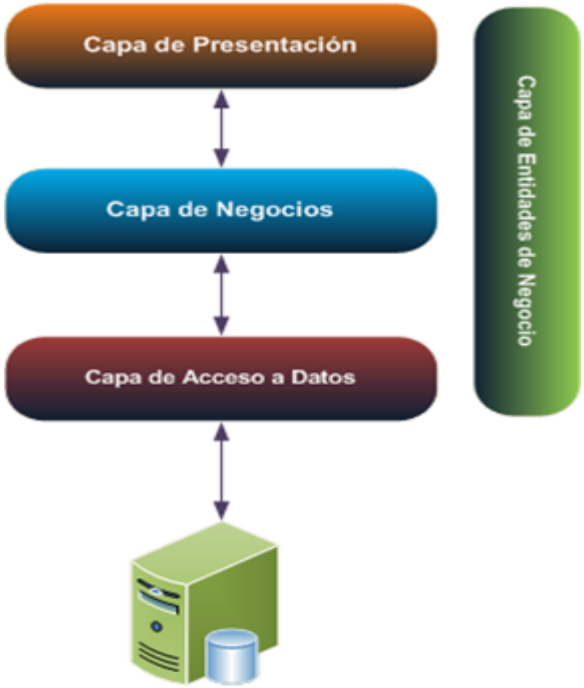

Figura 3. 87. Arquitectura en capas

Como se puede ver en la figura, en una arquitectura n-layer las capas solamente interactúan con sus capas adyacentes lo que permite abstraer funcionalidades de las capas superiores e inferiores. Por ejemplo, la capa de presentación no se da cuenta que tipo de base de datos o que repositorio de datos se utiliza porque esta solamente se comunica con la capa de negocios, y el repositorio de datos no se da cuenta en

donde se está utilizando o desplegando la información ya que este interactúa con la capa de acceso a datos. (Icomparable, 2008)

3.6. IMPLEMENTACIÓN DEL SISTEMA DE INFORMACIÓN WEB

En esta sección se revisará la forma en que se implementarán los modelos de diseño obtenidos, se revisará también en detalle a manera de ejemplo la implementación de un caso de uso.

3.6.1. Ejemplo de implementación de un caso de uso

Se muestra la implementación del caso de uso "Gestionar Paralelo", para esto mostramos el procedimiento en la figura 4.36, este procedimiento indica de manera general los pasos que se siguen en la implementación de casos de uso definidos en "Ingeniería de Requisitos", se explica a continuación.

http://localhost:4538/pg_Admisiones.aspx

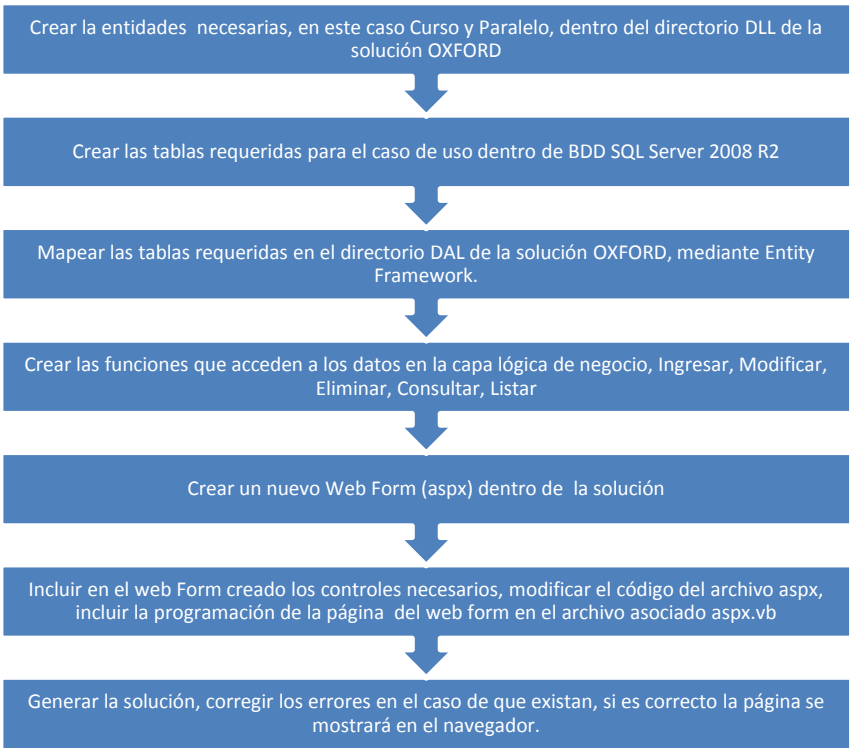

Figura 3. 88. Proceso de implementación de Gestión de Paralelo

1. Creación de las entidades necesarias

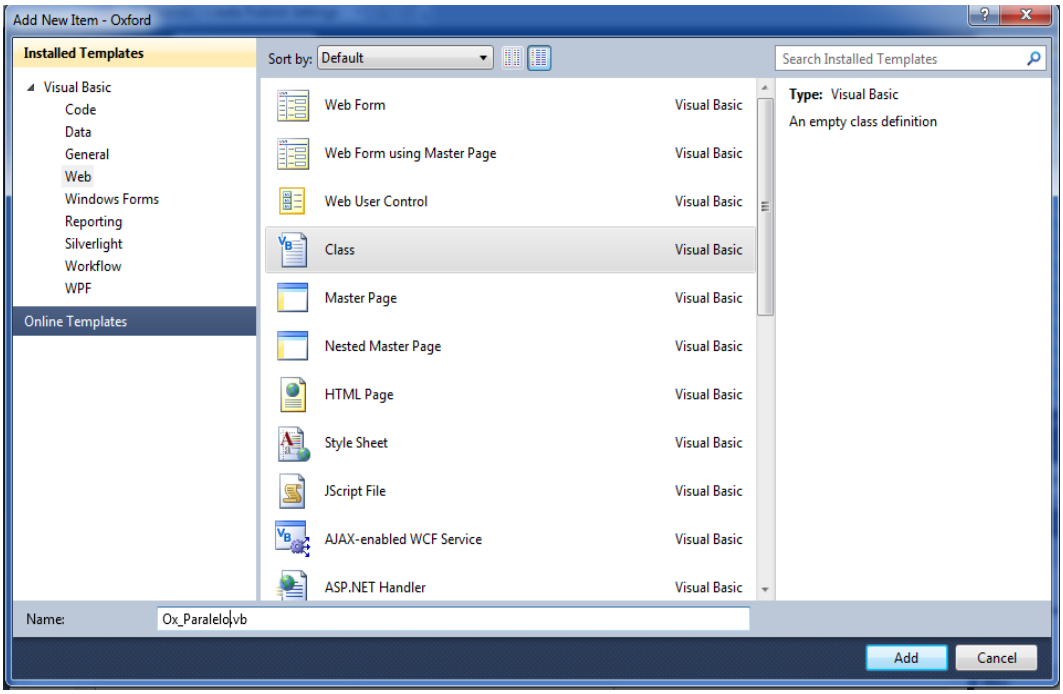

Figura 3. 89. Creación nueva Clase Paralelo.vb

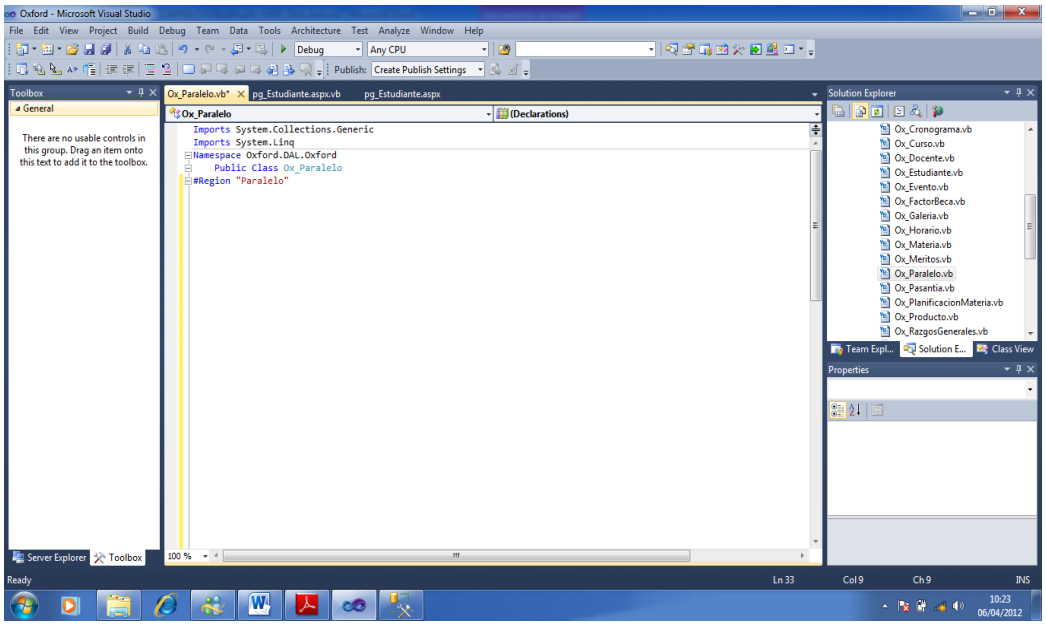

Figura 3. 90. Editor de código de la Clase Paralelo.vb

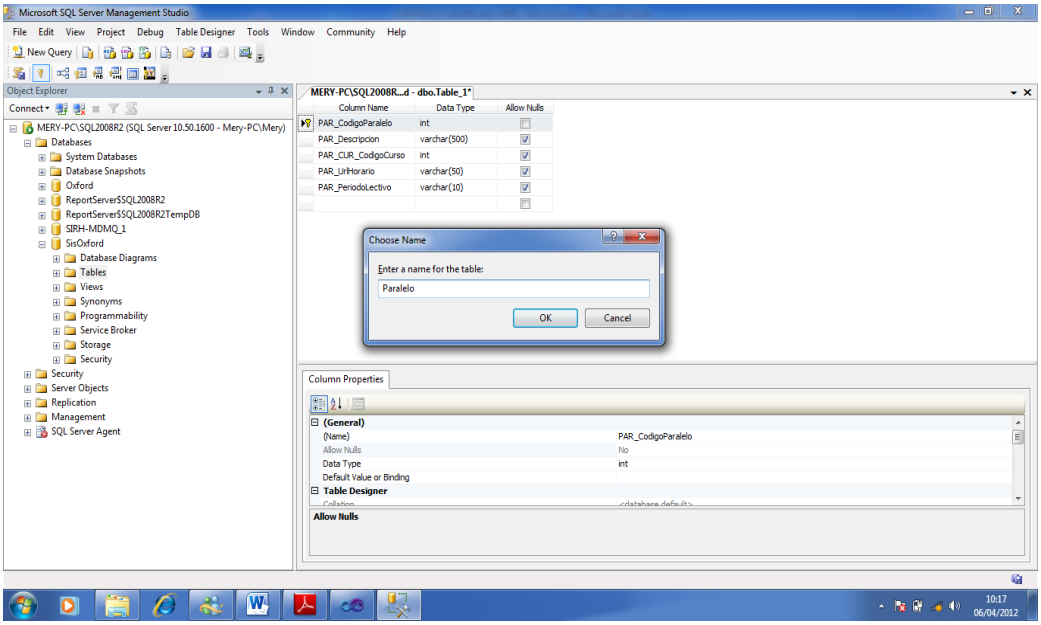

Figura 3. 91. Creación de las tablas en SQL Server 2008 R2

2. Mapear las tablas en Visual Studio mediante Entity Data Model.

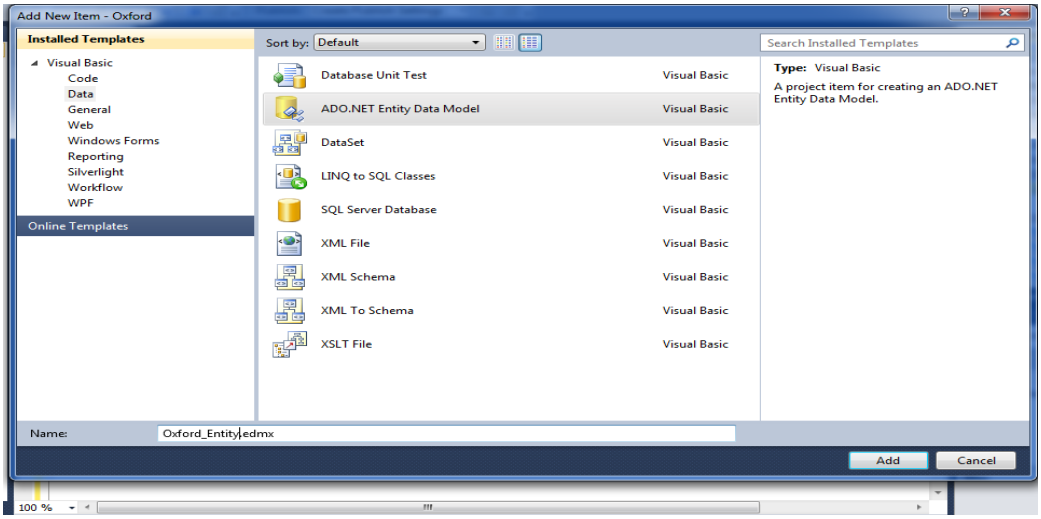

Figura 3. 92. Creación del Entity framework

3. Creación de las funciones de acceso a los datos.

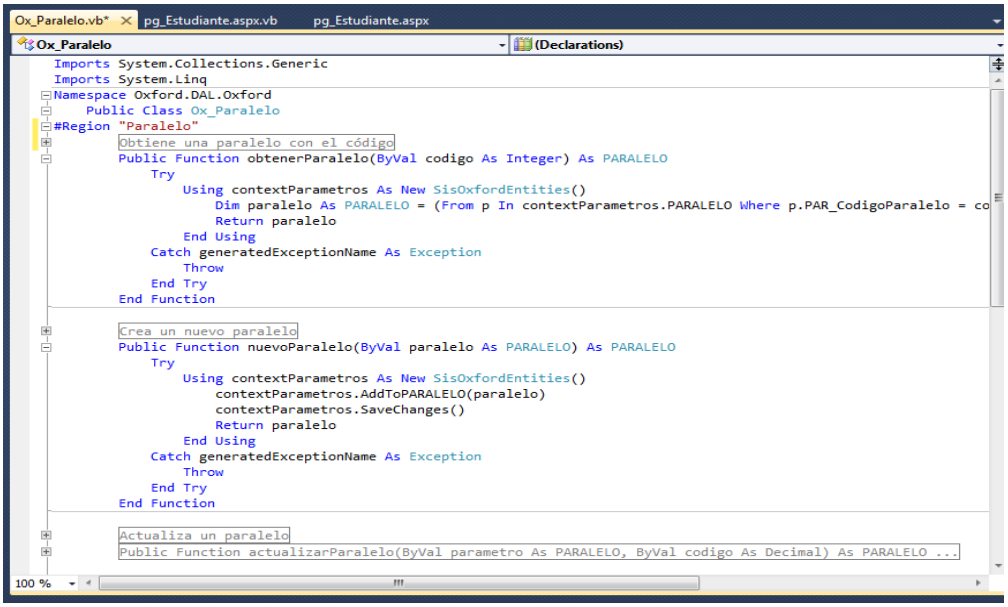

Figura 3. 93. Codificación de las funciones de acceso a los datos

4. Creación de los Web forms que implementan los casos de uso, en este caso "Gestionar paralelo".

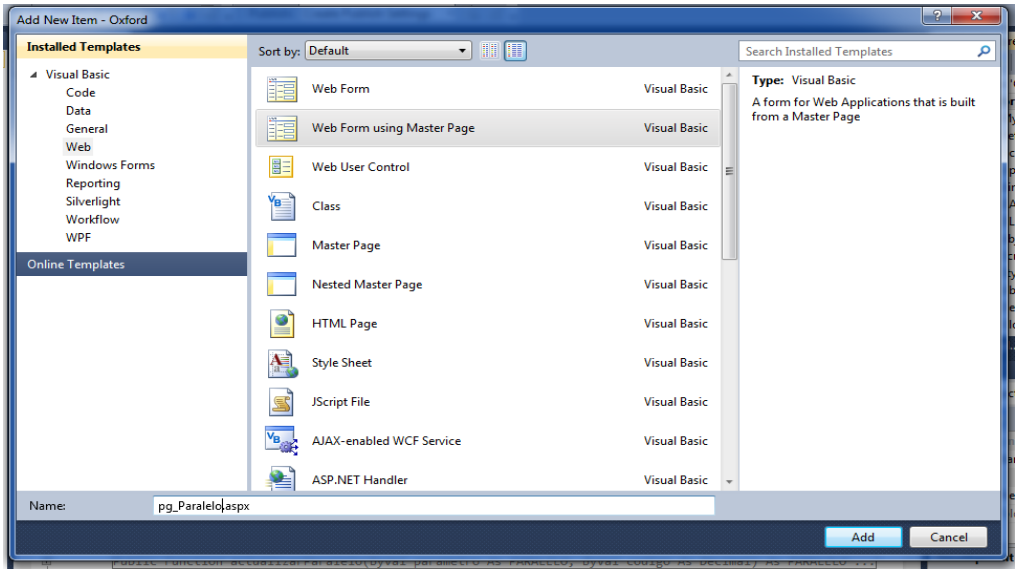

Figura 3. 94. Crear Web Form

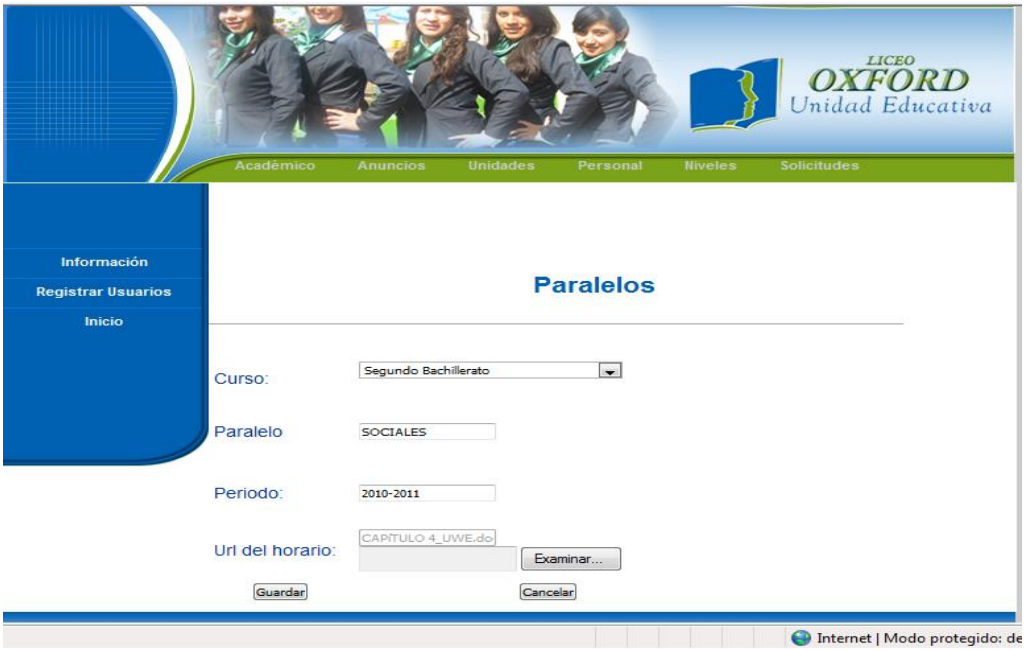

Figura 3. 95. Codificación de los métodos

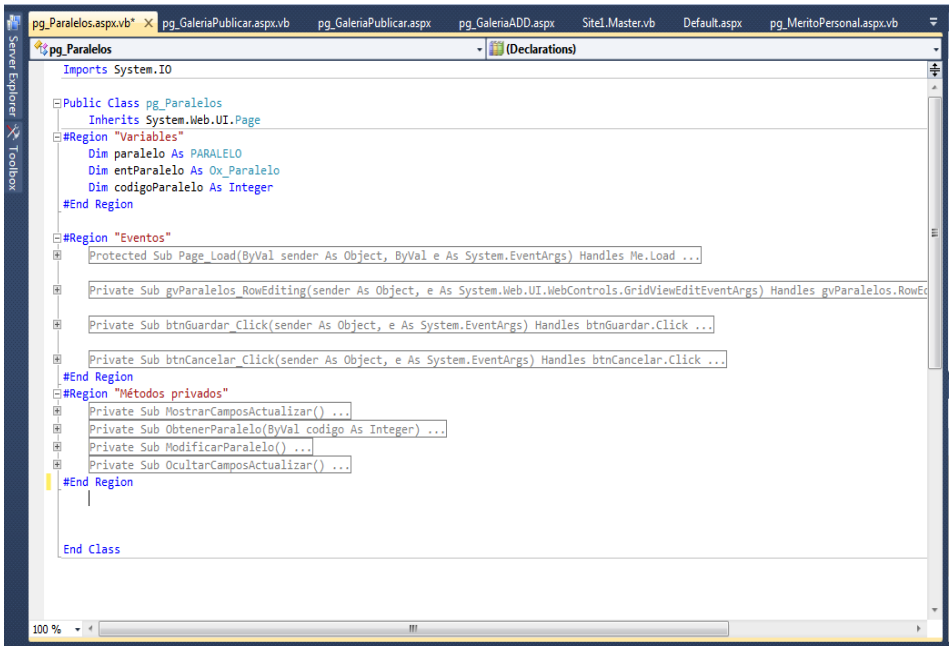

Figura 3. 96. Página mostrada en el nevegador

3.1.2. Generación de la solución.

Para generar la solución se compila toda la solución y se crea un nuevo proyecto de instalación.

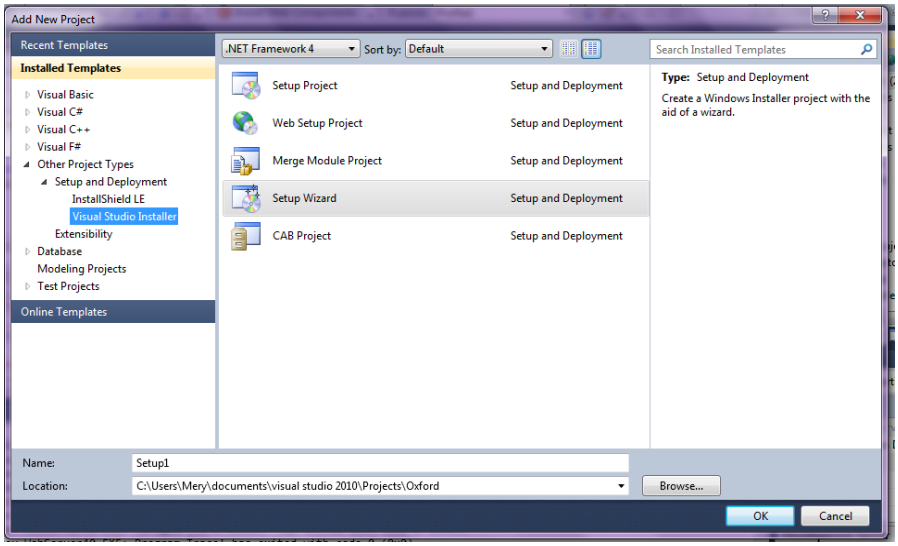

Figura 3. 97. Nuevo proyecto de instalación (SETUP WIZARD)

Una vez creado el nuevo proyecto de instalación, mediante el explorador se busca el instalador (SETUP) que se encuentra dentro de la carpeta DEBUG, la misma que está contenida en la carpeta del instalador creado.

3.1.2. Instalación del software

Se ejecuta el instalador (SETUP) y aparece una pantalla como se muestra a continuación.

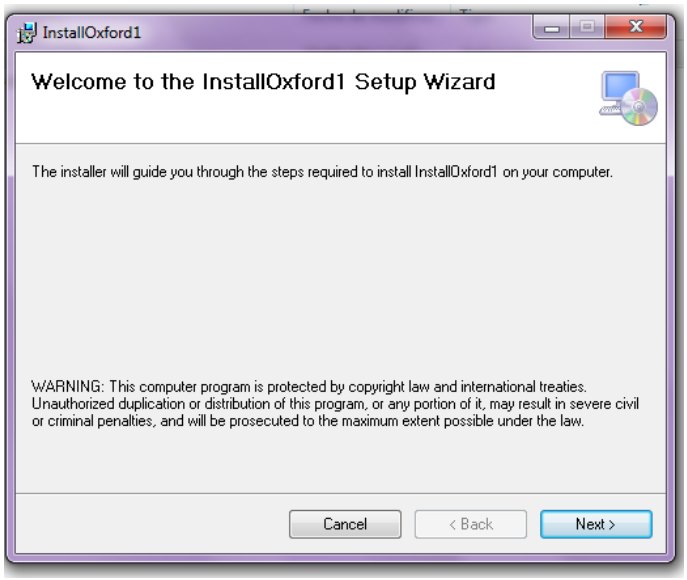

Figura 3. 98. Instalación del sistema web

Para continuar la instalación se da clic en el botón siguiente, hasta finalizar.

Es necesario tener en cuenta los siguientes aspectos:

1. Si se da un error de compatibilidad entre IIS7 y IIS6; es necesario activar esta opción en el Panel de control en Activar características de Windows, en Internet Information Service, opción "Compatibilidad de la configuración IIS6 y metabase de IIS"
2. En la configuración del IIS en el grupo de aplicaciones verificar que en el grupo DefaultAppPool, la propiedad Identidad esté en LOCALSYSTEM.

Finalmente queda instalado el software, para verificar esto se lo puede revisar en la carpeta INETPUB subcarpeta WWWROOT, donde debe haberse creado ya una nueva carpeta con el nombre del instalador, la misma que contiene todos los archivos del sistema.

3.7. Pruebas

Un aspecto muy importante en el control de calidad del desarrollo de software son las pruebas y, dentro de estas, las pruebas funcionales, en las cuales se hace una verificación dinámica del comportamiento de un sistema, basada en la observación de un conjunto seleccionado de ejecuciones controladas o casos de prueba.

3.7.1. Casos de prueba

Para definir los resultados obtenidos serán representados por la siguiente notación:

- SATISFACTORIO: cuando el objeto sometido a prueba responde según el resultado esperado de la prueba.
- FALLIDO: cuando el sistema se cae, presenta errores o realiza mal la funcionalidad a evaluar en la prueba.

En el caso de definir la gravedad del error:

- Alta si el error no puede pasarse por alto.
- Media si es manejable por el usuario aún con el error hasta que sea cambiado.
- Baja si el error no causa dificultad en otras pantallas.

Para la identificación del caso de prueba se lo nombra con el nombre que aparece en el menú seguido por la clase a verificar y numerada; estos deben estar separados por puntos.

Los casos de prueba se ejecutarán por cada pantalla y se muestran a continuación:

Tabla 3. 41.

Caso de prueba de las pantallas de ingreso y modificación de MATERIA

Caso de Prueba **1**	
Código de Identificación:	**Académico.Materia_01**
Nombre del Proyecto:	**POWELIOX_Académico_Materia**
Descripción (Alcance y Objetivos):	**Verificar la funcionalidad tanto de ingreso como modificación de los datos de una materia.**
Requisitos asociados	**REQ20, REQ21, REQ22** **CU18, CU19**
Variables de Entrada (Inputs):	**Nombre** **Cantidad_horas**
Flujo normal del evento	**1.- Ingresar nombre de la materia** **2.- Ingresar la cantidad de horas que corresponden a la materia** **3.- Elegir el área de la lista** **4.- Guardar**
Resultado esperado:	**Mensaje de guardado con éxito** **Mensaje de modificado exitoso**
Flujo alterno	**1.-Si el usuario ingresa una cantidad de caracteres mayor al que soporta el nombre.** **2.-Si ingresa una cantidad de horas mayor a las definidas.** **3.-Si presiona en el botón Cancelar**
Resultado alternativo esperado:	**1.-Mensajes que indican que es necesario llenar todos los campos.** **2.-Mensaje de dimensión máxima de cantidad de caracteres o número máximo de horas** **3.-Cancelar en la ventana de ingreso limpia los campos** **4.-Cancelar en la ventana de modificación, retorna al listado**
Evaluación de prueba	
Ejecutado por:	**Administrador del sistema**
Lugar de ejecución	**Liceo Oxford**
Resultados obtenidos	**Satisfactorio**
Observaciones:	
Gravedad del error:	
Notas del programador	
Estado:	**Resuelto:**
Acciones de corrección:	
Corregido por:	

Tabla 3. 42.

Caso de prueba de las pantallas de ingreso y modificación de MÉRITO

Caso de Prueba **2**	
Código de Identificación:	**Académico.Merito_01**
Nombre del Proyecto:	**POWELIOX_Académico_Merito**
Descripción (Alcance y Objetivos):	**Verificar la funcionalidad tanto de ingreso como modificación de los datos de un mérito.**
Requisitos asociados	**REQ16, REQ17** **CU14, CU15**
Variables de Entrada (Inputs):	**Descripción** **Acreedor del mérito**
Flujo normal del evento	**1.- Ingresar la descripción del mérito** **2.- Seleccionar el acreedor del mérito(docente o estudiante)** **3.- Seleccionar el motivo del mérito** **4.- Seleccionar una foto** **5.- Guardar**
Resultado esperado:	**1.-Mensaje de guardado con éxito** **2.-Mensaje de modificación exitosa**
Flujo alterno	**1.-Si el usuario ingresa una cantidad de caracteres mayor al que soporta la descripción.** **2.-Si no elige un acreedor del mérito** **3.-Si no selecciona una foto** **4.-Si presiona el botón cancelar**
Resultado alternativo esperado:	**1.-Mensajes que indican que es necesario llenar los campos requeridos, descripción del mérito, acreedor, foto** **2.-Cancelar en la ventana de ingreso limpia los campos** **3.-Cancelar en la ventana de modificación retorna al listado.**
Evaluación de prueba	
Ejecutado por:	**Administrador del sistema**
Lugar de ejecución	**Liceo Oxford**
Resultados obtenidos	**Fallido**
Observaciones:	**Guarda un mérito sin indicar el acreedor, y cuando se intenta modificar se cae.**
Gravedad del error:	**Alta**
Notas del programador	
Estado:	**Resuelto:**
Acciones de corrección:	**Validar este campo para que no permita registrar un mérito sin haber elegido un acreedor.**
Corregido por:	**Desarrollador**

Tabla 3. 43.

Caso de prueba de las pantallas de ingreso y modificación de BECA

Caso de Prueba **3**	
Código de Identificación:	**Académico.Beca_01**
Nombre del Proyecto:	**POWELIOX_Académico_Beca**
Descripción (Alcance y Objetivos):	**Verificar la funcionalidad tanto de ingreso como modificación de los datos de una beca y el motivo de la beca.**
Requisitos asociados	**REQ11, REQ12** **CU10, CU11**

Continúa →

Variables de Entrada (Inputs):	**Descripción** **Factor beca**
Flujo normal del evento	**1.- Ingresar la descripción de la beca** **2.- Seleccionar el estudiante** **3.- Seleccionar el tipo de beca**
Resultado esperado:	**1.-Mensaje de guardado con éxito** **2.-Mensaje de modificación exitosa**
Flujo alterno	**1.-Si el usuario ingresa una cantidad de caracteres mayor al que soporta la descripción.** **2.-Si no existen tipo de beca previamente ingresados** **3.-No existen becas registradas** **4.-Si presiona en el botón Cancelar**
Resultado alternativo esperado:	**1.-Mensajes que indican que es necesario llenar los campos requeridos: descripción.** **2.-Pantalla de ingreso de tipos de beca.** **3.-Mensaje que no existen becas registradas.** **4.-Cancelar en la ventana de ingreso limpia los campos** **5.-Cancelar en la ventana de modificación retorna al listado.**
Evaluación de prueba	
Ejecutado por:	**Administrador del sistema**
Lugar de ejecución	**Liceo Oxford**
Resultados obtenidos	**Satisfactorio**
Observaciones:	
Gravedad del error:	
Notas del programador	
Estado:	**Resuelto:**
Acciones de corrección:	
Corregido por:	

Tabla 3. 44.

Caso de prueba de las pantallas de ingreso y modificación de AREA

Caso de Prueba **4**	
Código de Identificación:	**Académico.Área_01**
Nombre del Proyecto:	**POWELIOX_Académico_Área**
Descripción (Alcance y Objetivos):	**Verificar la funcionalidad tanto de ingreso como modificación de los datos de una área.**
Requisitos asociados	**REQ17, REQ18, REQ19** **CU16, CU17**
Variables de Entrada (Inputs):	**Nombre**
Flujo normal del evento	**1.- Ingresar el nombre del área** **2.- Guardar**
Resultado esperado:	• **Mensaje de guardado con éxito** • **Mensaje de modificación exitosa**
Flujo alterno	• **Si el usuario ingresa una cantidad de caracteres mayor al que soporta el nombre.** • **Si presiona sobre el botón Cancelar**
Resultado alternativo esperado:	• **Mensajes que indican que es necesario llenar los campos nombre.** • **Cancelar en la ventana de ingreso limpia los campos.**
Evaluación de prueba	
Ejecutado por:	**Administrador del sistema**
Lugar de ejecución	**Liceo Oxford**
Resultados obtenidos	**Satisfactorio**
Observaciones:	

Continúa →

Gravedad del error:	
Notas del programador	
Estado:	**Resuelto:**
Acciones de corrección:	
Corregido por:	

Tabla 3. 45.

Caso de prueba de las pantallas de ingreso y modificación de EVENTO

Caso de Prueba **5**	
Código de Identificación:	**Anuncios.Evento_01**
Nombre del Proyecto:	**POWELIOX_Anuncios_Evento**
Descripción (Alcance y Objetivos):	**Verificar la funcionalidad tanto de ingreso como modificación de los datos de un evento**
Requisitos asociados	**REQ3, REQ4, REQ5** **CU05, CU06**
Variables de Entrada (Inputs):	**Nombre del evento** **Fecha de inicio** **Fecha de finalización** **Involucrados** **Detalle** **Lugar**
Flujo normal del evento	**1.- Ingresar el nombre del evento** **2.- Ingresa la fecha de inicio** **3.- Ingresa la fecha de finalización** **4.- Ingresa el lugar donde es el evento** **5.- Ingresa los involucrados** **6.- Ingresa un detalle del evento.** **7.- Guardar**
Resultado esperado:	**Mensaje de guardado con éxito** **Mensaje de modificación exitosa**
Flujo alterno	• **Si el usuario ingresa una cantidad de caracteres mayor al que soporta el nombre.** • **Si no escribe en los campos requeridos: nombre, fecha inicio, fecha fin, involucrados, lugar, detalle.** • **Si las fecha son incoherentes fecha fin menor a la de inicio.** • **Que no existan eventos previamente registrados para la modificación.** • **Si presiona sobre el botón Cancelar.**
Resultado alternativo esperado:	• **Mensajes que indican que es necesario llenar los campos nombre del evento, fecha inicio, fecha fin, lugar, involucrados, detalle.** • **Cancelar en la ventana de ingreso limpia los campos.** • **Cancelar en la ventana de modificación retorna al listado de actividades que corresponden al cronograma.**
Evaluación de prueba	
Ejecutado por:	**Administrador del sistema**
Lugar de ejecución	**Liceo Oxford**
Resultados obtenidos	**Satisfactorio**
Observaciones:	
Gravedad del error:	
Notas del programador	
Estado:	**Resuelto:**
Acciones de corrección:	

Tabla 3. 46.

Caso de prueba de las pantallas de ingreso y modificación de CRONOGRAMA

Caso de Prueba **6**	
Código de Identificación:	**Anuncios.Cronograma_01**
Nombre del Proyecto:	**POWELIOX_Anuncios_Cronograma**
Descripción (Alcance y Objetivos):	**Verificar la funcionalidad tanto de ingreso como modificación de los datos de un cronograma.**
Requisitos asociados	**REQ6, REQ7, REQ8, REQ9, REQ10** **CU07, CU08, CU09**
Variables de Entrada (Inputs):	**Periodo** **Descripción** **Fecha inicio** **Fecha fin** **Descripción_detalle**
Flujo normal del evento	**1.- Ingresar el periodo lectivo** **2.- Ingresar la descripción del cronograma** **3.- Guardar** **Mensaje de Ingreso de las actividades que corresponden al cronograma.** **4.- Ingresa la descripción de la actividad** **5.- Ingresa la fecha inicial** **6.- Ingresa la fecha final** **7.- Guardar** **Para eliminar:** **1.- Selecciona la opción eliminar del listado de actividades**
Resultado esperado:	• **Mensaje de guardado del cronograma** • **Mensaje de guardado de las actividades del cronograma.** • **Mensaje de modificación exitosa** • **En el caso de eliminación, si son eliminadas todas las actividades del cronograma, entonces también queda eliminado el cronograma. Si desea eliminar en forma directa el cronograma no es posible debido a las actividades que este contiene.**
Flujo alterno	• **Si no ha llenado los campos requeridos.** • **Si las fecha no son coherentes** • **Si hay errores en la validación de los datos** • **Para la modificación** • **Si no existen cronogramas previamente registrados**
Resultado alternativo esperado:	• **Mensajes que indican que es necesario llenar los campos periodo, descripción, descripción del detalle, fecha inicio, fecha fin.** • **Mensaje de error de validación de los datos** • **Mensaje de error en las fechas**
Evaluación de prueba	
Ejecutado por:	**Administrador del sistema**
Lugar de ejecución	**Liceo Oxford**
Resultados obtenidos	**Satisfactorio**
Observaciones:	**El sistema permite ingresar una descripción de cronograma sin actividades.**
Gravedad del error:	**Leve**
Notas del programador	
Estado:	**Resuelto:**
Acciones de corrección:	
Corregido por:	

Tabla 3. 47.

Caso de prueba de las pantallas de ingreso y modificación de GALERÍA

Caso de Prueba **7**	
Código de Identificación:	**Anuncios.Galería_01**
Nombre del Proyecto:	**POWELIOX_Anuncios_Galería**
Descripción (Alcance y Objetivos):	**Verificar la funcionalidad tanto de ingreso como modificación de los datos de una galería.**
Requisitos asociados	**REQ12, REQ13** **CU13, CU14**
Variables de Entrada (Inputs):	**Descripción** **Url Imágenes**
Flujo normal del evento	**1.- Ingresar la descripción** **2.- Selecciona la ubicación del archivo de imagen** **3.- Guardar**
Resultado esperado:	• **Mensaje de guardado de la galería** • **Mensaje de modificación exitosa**
Flujo alterno	• **Si no ha llenado los campos requeridos.** • **Si hay errores en la validación de los datos** • **Para la modificación** • **Si no existen galerías previamente registrados**
Resultado alternativo esperado:	• **Mensajes que indican que es necesario llenar los campos descripción y url de imágenes.** • **Mensaje de error de validación de los datos**
Evaluación de prueba	
Ejecutado por:	**Administrador del sistema**
Lugar de ejecución	**Liceo Oxford**
Resultados obtenidos	**Satisfactorio**
Observaciones:	
Gravedad del error:	
Notas del programador	
Estado:	**Resuelto:**
Acciones de corrección:	
Corregido por:	

Tabla 3. 48.

Caso de prueba de las pantallas de ingreso y modificación de UNIDAD DE PRODUCCIÓN

Caso de Prueba **8**	
Código de Identificación:	**Unidades.Producción_01**
Nombre del Proyecto:	**POWELIOX_Unidades_Producción**
Descripción (Alcance y Objetivos):	**Verificar la funcionalidad tanto de ingreso como modificación de los datos de una unidad de producción.**
Requisitos asociados	**REQ30, REQ31** **CU25, CU26**
Variables de Entrada (Inputs):	**Responsable** **Descripción** **Fecha creación** **Ubicación**

Continúa →

Flujo normal del evento	**1.- Ingresar el nombre del responsable.** **2.- Ingresar la descripción de la unidad.** **3.- Selecciona el archivo del organigrama** **4.- Ingresa la ubicación de la unidad.** **3.- Guardar**
Resultado esperado:	• **Mensaje de guardado de la unidad de producción** • **Mensaje de modificación exitosa**
Flujo alterno	• **Si no ha llenado los campos requeridos** • **Si hay errores en la validación de los datos** • **Para la modificación** • **Si no existen unidades previamente registradas**
Resultado alternativo esperado:	• **Mensajes que indican que es necesario llenar los campos responsable, descripción, fecha de creación, url del organigrama, ubicación.** • **Mensaje de error de validación de los datos**
Evaluación de prueba	
Ejecutado por:	**Administrador del sistema**
Lugar de ejecución	**Liceo Oxford**
Resultados obtenidos	**Satisfactorio**
Observaciones:	**El sistema no valida la fecha de creación que sea menor o igual a la fecha actual**
Gravedad del error:	**Leve**
Notas del programador	
Estado:	**Resuelto:**
Acciones de corrección:	**Validada la fecha de creación para que no sea mayor a la fecha actual.**
Corregido por:	**Desarrollador**

Caso de Prueba **9**	
Código de Identificación:	**Unidades.Producción_Producto_01**
Nombre del Proyecto:	**POWELIOX_Unidades_Producción_Producto**
Descripción (Alcance y Objetivos):	**Verificar la funcionalidad tanto de ingreso como modificación de los datos de un producto.**
Requisitos asociados	**REQ32, REQ33** **CU27, CU28 -**
Variables de Entrada (Inputs):	**Composición** **Nombre del producto**
Flujo normal del evento	**1.- Ingresar la composición del producto** **2.- Ingresar el nombre del producto** **3.- Selecciona la unidad a la que corresponde** **4.- Guardar**
Resultado esperado:	• **Mensaje de guardado del producto.** • **Mensaje de modificación exitosa.**
Flujo alterno	• **Si no ha llenado los campos requeridos.** • **Si hay errores en la validación de los datos.** • **Si no existen unidades previamente registradas.** **Para la modificación** **Que no existan productos registrados.**
Resultado alternativo esperado:	• **Mensajes que indican que es necesario llenar los campos nombre del producto, composición del producto.** • **Mensaje de error de validación de los datos** • **Mensaje de inexistencia de unidades registradas.** • **Mensaje de listado vacío de productos.**
Evaluación de prueba	

Continúa →

Ejecutado por:	**Administrador del sistema**
Lugar de ejecución	**Liceo Oxford**
Resultados obtenidos	**Satisfactorio**
Observaciones:	
Gravedad del error:	
Notas del programador	
Estado:	**Resuelto:**
Acciones de corrección:	
Corregido por:	

Tabla 3. 49.

Caso de prueba de las pantallas de ingreso y modificación de PRODUCTO

Tabla 3. 50.

Caso de prueba de las pantallas de ingreso y modificación de DOCENTE

Caso de Prueba **10**	
Código de Identificación:	**Personal.Docente_01**
Nombre del Proyecto:	**POWELIOX_Personal_Docente**
Descripción (Alcance y Objetivos):	**Verificar la funcionalidad tanto de ingreso como modificación de los datos de un docente.**
Requisitos asociados	**REQ34, REQ35** **CU03, CU04**
Variables de Entrada (Inputs):	**Nombres** **Apellidos** **Dirección** **Teléfono** **Título**
Flujo normal del evento	**1. Ingresar los nombres** **2. Ingresar los apellidos** **3. Ingresar la dirección** **4. Ingresar el e-mail** **5. Ingresar el título** **6. Ingresar número telefónico.** **7. Seleccionar el área a la que pertenece.** **8. Guardar**
Resultado esperado:	• **Mensaje de guardado del docente.** • **Mensaje de modificación exitosa.**
Flujo alterno	• **Si no ha llenado los campos requeridos.** • **Si hay errores en la validación de los datos.** • **Si no existen áreas previamente registradas.** **Para la modificación** **Que no existan docentes registrados.**
Resultado alternativo esperado:	• **Mensajes que indican que es necesario llenar los campos nombres, apellidos, dirección, teléfono, título.** • **Mensaje de error de validación de los datos** • **Mensaje de inexistencia de áreas registradas.** • **Mensaje de listado vacío de docentes.**
Evaluación de prueba	
Ejecutado por:	**Administrador del sistema**
Lugar de ejecución	**Liceo Oxford**
Resultados obtenidos	**Fallido**

Continúa →

Continúa →

Observaciones:	**En los campos nombres, apellidos, dirección, e-mail no está validado la cantidad máxima de caracteres.**
Gravedad del error:	**Media**
Notas del programador	
Estado:	**Resuelto:**
Acciones de corrección:	**Validados los campos con el número máximo de caracteres permitidos.**
Corregido por:	**Desarrollador.**

Tabla 3. 51.

Caso de prueba de las pantallas de ingreso y modificación de ESTUDIANTE

Caso de Prueba **11**	
Código de Identificación:	**Personal_Estudiante_01**
Nombre del Proyecto:	**POWELIOX_Personal_Estudiante**
Descripción (Alcance y Objetivos):	**Verificar la funcionalidad tanto de ingreso como modificación de los datos de un estudiante.**
Requisitos asociados	**REQ1, REQ2** **CU01, CU02**
Variables de Entrada (Inputs):	**Nombres** **Apellidos** **Dirección** **Teléfono** **Nombre representante** **Fecha de nacimiento**
Flujo normal del evento	**1. Ingresar los nombres** **2. Ingresar los apellidos** **3. Ingresar la dirección** **4. Ingresar el e-mail** **5. Ingresar el nombre des representante** **6. Ingresar número telefónico.** **7. Ingresar la fecha de nacimiento** **8. Selecciona el curso** **9. Selecciona el paralelo** **10. Guardar**
Resultado esperado:	• **Mensaje de guardado del estudiante.** • **Mensaje de modificación exitosa.**
Flujo alterno	• **Si no ha llenado los campos requeridos.** • **Si hay errores en la validación de los datos.** • **Si no existen áreas previamente registradas.** **Para la modificación** **Que no existan estudiantes registrados.**
Resultado alternativo esperado:	• **Mensajes que indican que es necesario llenar los** • **campos nombres, apellidos, dirección, teléfono, nombre del representante, fecha de nacimiento.** • **Mensaje de error de validación de los datos** • **Mensaje de inexistencia de cursos y paralelos registrados.** • **Mensaje de listado vacío de estudiantes.**
Evaluación de prueba	
Ejecutado por:	**Administrador del sistema**
Lugar de ejecución	**Liceo Oxford**
Resultados obtenidos	**Fallido**
Observaciones:	**En los campos nombres, apellidos, dirección, e-mail, nombre del**

Continúa →

	representante, fecha de nacimiento no están validados la cantidad máxima de caracteres.
Gravedad del error:	**Media**
Notas del programador	
Estado:	**Resuelto:**
Acciones de corrección:	**Validados los campos con el número máximo de caracteres permitidos.**
Corregido por:	**Desarrollador.**

Tabla 3. 52.

Caso de prueba de las pantallas de ingreso y modificación de PARALELO

Caso de Prueba **12**	
Código de Identificación:	**Niveles_Paralelo_01**
Nombre del Proyecto:	**POWELIOX_Niveles_Paralelo**
Descripción (Alcance y Objetivos):	**Verificar la funcionalidad tanto de ingreso como modificación de los datos de un paralelo.**
Requisitos asociados	**REQ28, REQ29** **CU32, CU33**
Variables de Entrada (Inputs):	**Paralelo** **Período**
Flujo normal del evento	**1. Ingresar el paralelo** **2. Ingresar el periodo** **3. Seleccionar el curso** **4. Seleccionar la ubicación del horario** **5. Guardar**
Resultado esperado:	• **Mensaje de guardado del estudiante.** • **Mensaje de modificación exitosa.**
Flujo alterno	• **Si no ha llenado los campos requeridos.** • **Si hay errores en la validación de los datos.** • **Si no existen cursos previamente registrados.** **Para la modificación** **Que no existan paralelos registrados.**
Resultado alternativo esperado:	• **Mensajes que indican que es necesario llenar los campos paralelo, periodo, url del horario.** • **Mensaje de error de validación de los datos** • **Mensaje de inexistencia de cursos registrados.** • **Mensaje de listado vacío de paralelos.**
Evaluación de prueba	
Ejecutado por:	**Administrador del sistema**
Lugar de ejecución	**Liceo Oxford**
Resultados obtenidos	**Satisfactorio**
Observaciones:	
Gravedad del error:	
Notas del programador	
Estado:	**Resuelto:**
Acciones de corrección:	
Corregido por:	

Tabla 3. 53.

Caso de prueba de las pantallas de ingreso y modificación de PASANTÍA

Caso de Prueba **13**	
Código de Identificación:	**Información.Pasantía_01**
Nombre del Proyecto:	**POWELIOX_Información_Pasantía**
Descripción (Alcance y Objetivos):	**Verificar la funcionalidad tanto de ingreso o modificación de los datos básicos de pasantías.**
Requisitos asociados	**REQ26, REQ27** **CU23, CU24**
Variables de Entrada (Inputs):	**Áreas** **Descripción** **Fecha de inicio** **Nivel**
Flujo normal del evento	**1. Ingresar las áreas a aplicar** **2. Ingresar la descripción de las actividades a realizar.** **3. Ingresar la fecha de inicio** **4. Seleccionar el nivel (Inicial, básico, bachillerato)** **5. Guardar**
Resultado esperado:	• **Mensaje de guardado de los datos de la pasantía.** • **Mensaje de modificación exitosa.**
Flujo alterno	• **Si hay errores en la validación de los datos.** • **Si hay campos sin llenar y son requeridos.**
Resultado alternativo esperado:	• **Mensaje de error de validación de los datos** • **Mensajes de campos vacíos que son necesarios llenar: áreas, descripción, fecha de inicio.** • **Mensaje de listado vacío de pasantías.**
Evaluación de prueba	
Ejecutado por:	**Administrador del sistema**
Lugar de ejecución	**Liceo Oxford**
Resultados obtenidos	**Satisfactorio**
Observaciones:	
Gravedad del error:	
Notas del programador	
Estado:	**Resuelto:**
Acciones de corrección:	
Corregido por:	

Tabla 3. 54.

Caso de prueba de las pantallas de ingreso y modificación de USUARIO

Caso de Prueba **14**	
Código de Identificación:	**Registrar Usuarios_01**
Nombre del Proyecto:	**POWELIOX_Registrar usuarios**
Descripción (Alcance y Objetivos):	**Verificar la funcionalidad tanto de ingreso de datos de nuevos usuarios o su eliminación.**
Requisitos asociados	**REQ40, REQ41** **CU34, CU35**
Variables de Entrada (Inputs):	**Usuario** **Clave**
Flujo normal del evento	**1. Ingresar el nombre de usario** **2. Ingresar la clave** **3. Guardar** **En el caso de eliminar** **Elegir un usuario de la lista desplegada**
Resultado esperado:	• **Mensaje de guardado de los datos de usuario registrado.** • **Mensaje de eliminación**
Flujo alterno	• **Si no ha llenado los campos requeridos.** • **Si hay errores en la validación de los datos.**
Resultado alternativo esperado:	• **Mensaje de error de validación de los datos** • **Mensajes d campos requeridos.** • **Mensaje de listado vacío de usuarios**
Evaluación de prueba	
Ejecutado por:	**Administrador del sistema**
Lugar de ejecución	**Liceo Oxford**
Resultados obtenidos	**Satisfactorio**
Observaciones:	
Gravedad del error:	
Notas del programador	
Estado:	**Resuelto:**
Acciones de corrección:	
Corregido por:	

Tabla 3. 55.

Caso de prueba de las pantallas de ingreso y modificación de TAREA

Caso de Prueba **15**	
Código de Identificación:	**Docente.Tarea_01**
Nombre del Proyecto:	**POWELIOX_Docente_Tarea**
Descripción (Alcance y Objetivos):	**Verificar la funcionalidad tanto de ingreso, modificación y eliminación de los datos básicos de una tarea**
Requisitos asociados	**REQ23, REQ24, REQ25** **CU20, CU21, CU22**
Variables de Entrada (Inputs):	**Descripción** **Fecha de entrega** **Puntuación**
Flujo normal del evento	**1. Ingresar la descripción de la tarea** **2. Selecciona el docente** **3. Selecciona el curso** **4. Selecciona el paralelo** **5. Selecciona la materia** **6. Ingresar la fecha de entrega** **7. Ingresa el puntaje para la tarea**

Continúa →

	8. Guardar **Para eliminación** **Elige una tarea del listado de tarea y selecciona la opción eliminar.**
Resultado esperado:	• **Mensaje de guardado de los datos de la tarea.** • **Mensaje de eliminación exitosa.**
Flujo alterno	• **Si no ha llenado los campos requeridos.** • **Si hay errores en la validación de los datos.**
Resultado alternativo esperado:	• **Mensaje de error de validación de los datos** • **Mensaje de listado vacío de tareas.**
Evaluación de prueba	
Ejecutado por:	**Docente de la institución.**
Lugar de ejecución	**Liceo Oxford**
Resultados obtenidos	**Satisfactorio**
Observaciones:	**El campo puntaje admite cualquier valor, cuando el máximo puntaje que puede recibir es de 10 puntos.** **Y fecha de entrega no puede ser la misma que la fecha actual.**
Gravedad del error:	**Media**
Notas del programador	
Estado:	**Resuelto:**
Acciones de corrección:	**Validar el campo puntaje para que admita hasta 10 como máximo valor.** **Validar el campo fecha para que no admita una fecha menor o igual a la actual.**
Corregido Por:	**Desarrollador**

CAPÍTULO IV

4. CONCLUSIONES Y RECOMENDACIONES

4.1. Conclusiones

a) Después de haber investigado y analizado la evolución de las aplicaciones web dinámicas, se puede concluir que una de las técnicas con gran efectividad es AJAX.

b) Una vez investigado sobre la técnica AJAX, se concluye que es una técnica muy eficiente debido a que evita que el usuario permanezca mucho tiempo frente a una pantalla en blanco en espera del retorno de una página con la información solicitada, esto hace que la experiencia del usuario sea satisfactoria.

c) Mediante el desarrollo de la aplicación se puede concluir también que las tecnologías embebidas dentro de esta técnica son de gran utilidad y permiten que las aplicaciones sean eficientes, debido a que ayudan en la interacción del usuario con la página web haciendo que los pedido se ejecuten en forma asincrónica.

d) Mediante la aplicación realizada se puede indicar que la funcionalidad es similar a una aplicación de escritorio por su efectividad en la ejecución.

e) AJAX funciona en cualquier navegador moderno, debido a que es compatible con cualquier tipo de servidor estándar y lenguaje de programación web.

f) Con AJAX se puede manejar la aplicación como que fuese una aplicación de escritorio casi sin darse cuenta que está dentro de un navegador web.

g) La metodología utilizada para el desarrollo es de gran utilidad debido a su proceso ordenado y sistemático, características especiales que ayudan a entender y solucionar de mejor manera las propuestas en desarrollo de sistemas.

h) UWE es una metodología que se enfoca mayormente en la fase de Análisis y Diseño puesto que es la base para un desarrollo efectivo.

i) Desarrollar sistemas mediante la utilización de métodos y metodologías es de gran importancia ya que estos generan procesos ordenados y entendibles, los mismos que permiten que el software no quede atado a su desarrollador.

4.2. Recomendaciones

- Aplicar esta técnica en el desarrollo de aplicaciones web dinámicas debido a su funcionalidad y productividad en la interacción del usuario y la aplicación.
- Para que se puedan ejecutar aplicaciones con AJAX es necesario que el usuario tenga activado JavaScript en el navegador.
- Aplicar esta técnica en el desarrollo de aplicaciones que requieran de gran dinámica en su interacción.
- Utilizar métodos y metodologías hacen que el desarrollo sea un trabajo menos complejo y más entendible.
- Para utilizar la metodología es necesario contar con una buena herramienta para la creación de los modelos y evitar así pérdidas de tiempo.

LINKGRAFÌA

- Blogs. (24 de 3 de 2010). Obtenido de http://primergrupoisc.blogspot.com/2010/03/tecnologias-agrupadas-bajo-el-concepto.html
- Chile, U. d. (2012). Obtenido de http://www.dcc.uchile.cl/~rbaeza/inf/xml.html)
  chileunder. (12 de 09 de 2011). Obtenido de http://www.chileunder.com/programacion-193/lenguajes-de-programacion-web-ventajas-y-desventajas-3142.html
- DEVX. (14 de 04 de 2006). Obtenido de http://www.devx.com/codemag/Article/31195
- Eguiluz, J. (2010). LibrosWeb. Obtenido de http://librosweb.es/libro/ajax/capitulo_1.html
  ElCodigo. (2 de 1 de 2011). Obtenido de http://www.elcodigo.net/tutoriales/jsavanzado/jsavanzado7.htl)
- Ferri., I. C. (8 de 5 de 2010). C O M D I T E C H. Obtenido de http://www.comditech.com/index.php?option=com_content&view=article&id=28%3Aestaticasvsdinamicas&catid=31%3Agenerales&Itemid=46
- García, F. (13 de 1 de 2009). USAL. Obtenido de http://zarza.usal.es/~fgarcia/doc/tuto2/CGI.htm
- García, F. (23 de 09 de 2010). USAL. Obtenido de http://zarza.usal.es/~fgarcia/doc/tuto2/VI_3.htm
- García, F. (1 de 10 de 2010). USAL. Obtenido de http://zarza.usal.es/~fgarcia/doc/tuto2/VI_3.htm
- García, F. (1 de 10 de 2010). USAL. Obtenido de http://zarsa.usal.es/fgarcia/doc/tuto2/VI.3.htm
- Giardina, F. (7 de 02 de 2011). MaestrosDelWeb. Obtenido de http://www.maestrosdelweb.com/editorial/tutorial-asp-net-utilizacion-de-ajax/
- Gonzaga, A. (16 de 5 de 2007). Obtenido de

http://www.slideshare.net/rocker652/modelo-conceptual

HTMLQUICK. (2009). Obtenido de http://www.htmlquick.com/es/tutorials/css.html

- Icomparable. (21 de 10 de 2008). Obtenido de http://icomparable.blogspot.com/2008/10/arquitectura-n-tier-o-arquitectura-n.html
- Libros Web. (2010). Obtenido de http://www.librosweb.es/ajax/capitulo1.html)
- LibrosWeb. (12 de 12 de 2010). Obtenido de http://www.librosweb.es/ajax/capitulo1.html)
- Microsoft. (2011). Obtenido de http://www.microsoft.com/latam/msdn/articulos/2000/03/art03/#top
- Monografías. (2012). Obtenido de http://www.monografias.com/trabajos7/xml/xml.shtml).
- Piñol., C. M. (2012). Obtenido de

http://www.cibernetia.com/manuales/introduccion_aplicaciones_web/3_historia_aplicaciones_web.php

- Quin, L. (20 de 04 de 2013). W3. Obtenido de http://www.w3.org/XML/ Ramon. (2011). Obtenido de http://www.ramon.org/xml/articulos/intro_xml-html.htm
- Riko, G. (01 de 08 de 2010). Scribd. Obtenido de http://es.scribd.com/doc/44936310/Estudio-de-UWE-Metodologia-de-Desarrollo-Web
- Turmero, P. (20 de 3 de 2012). Obtenido de http://www.monografias.com/trabajos10/diusuar/diusuar.shtm
- Universidad Politécnica de Madrid, España. (2005). Universidad Politécnica de Madrid, España. Obtenido de http://www-app.etsit.upm.es/~alvaro/manual/manual.html
- Valle, J. (08 de 08 de 2003). Obtenido de http://www.webestilo.com/php/php12c.phtml

w3c. (2010). Pergaminoirtual. Obtenido de

http://www.pergaminovirtual.com/definicion/DOM.html

- W3Schools. (2010). Obtenido de http://www.w3schools.com/js/default.asp
- W3Schools. (2012). Obtenido de http://www.w3schools.com/html/html_intro.asp
- WXVC. (2011). Obtenido de http://www.xsvc.com.ve/tutoriales/ManualCssHojasDeEstilos.pdf

CERTIFICACIÓN

Se Certifica que el presente trabajo fue desarrollado por la señorita Mery Susana Zambonino Bautista, bajo nuestra supervisión.

Ing. Xavier Montaluisa
DIRECTOR DE PROYECTO

Ing. Marcelo Álvarez
CODIRECTOR DE PROYECTO

Ing. Lucas Garcés
DIRECTOR DE CARRERA

Dr. Rodrigo Vaca
SECRETARIO ACADÉMICO