CAPITULO I

TECNOLOGÍA ASP.NET

1. Tecnología ASP.NET

1.1. Introducción a la Tecnología ASP.NET

1.1.1 Que es ASP?

Las Active Server Pages son un ambiente de aplicación abierto y gratuito en el que se puede combinar código HTML, scripts y componentes ActiveX del servidor para crear soluciones dinámicas y poderosas para el web. [www.001]

1.1.2 Que es ASP.NET?

ASP.NET es un framework para aplicaciones web desarrollado y comercializado por Microsoft. ASP.NET es usado para el desarrollo de sitios web dinámicos, aplicaciones web y servicios web XML. Es la tecnología sucesora de la tecnología Active Server Pages (ASP).

ASP.NET, por sus características dadas a conocer más adelante, nos permite escribir código ASP usando cualquier lenguaje admitido por el .NET Framework.

1.1.3 Breve Historia de ASP.NET

Microsoft introdujo la tecnología llamada Active Server Pages en Diciembre de 1996; Microsoft comenzó a investigar las posibilidades para un nuevo modelo de aplicaciones web que pudiera resolver las quejas comunes sobre ASP, especialmente aquellas con respecto a la separación de la presentación y el contenido y ser capaz de escribir código "limpio". El diseño inicial fue desarrollado en el curso de dos meses por Anders y Guthrie, y Guthrie.

En prototipo inicial fue llamado "XSP".

El desarrollo inicial de XSP fue hecho usando Java, pero pronto se decidió construir una nueva plataforma sobre el Common Language Runtime(CLR), pues ofrecía un ambiente orientado a objetos, recolección de basura y otras características que fueron vistas como características deseables.

Con el cambio al Common Language Runtime, XSP fue implementado en C#, conocido internamente como "Project Cool" pero mantenido en secreto para el publico, y fue renombrado a ASP+, para este punto la nueva plataforma fue vista como el sucesor de Active Server Pages, y la intención fue proporcionar un medio fácil de migración para los desarrolladores ASP.

La primera demostración publica y la liberación de la primera beta de ASP+ (y el resto del .NET Framework) se realizó en el Microsoft's Professional Developers Conference (PDC) el 11 de Julio del 2000 en Orlando, Florida.

Durante la presentación de Bill Gates, Fujitsu demostró ASP+ usado en conjunción con COBOL, y el soporte para una variedad de otros lenguajes fue anunciada, incluyendo los nuevos lenguajes de Microsoft, Visual Basic .NET y C#, así como también el soporte por medio de herramientas de interoperabilidad para Python y Perl creadas por la empresa canadiense ActiveState.

Una vez que la marca ".NET" fue seleccionada en la segunda mitad del 2000. Se cambio el nombre de ASP+ a ASP.NET.

Después de cuatro años de desarrollo, y una serie de versiones de evaluación en los años 2000 y 2001, ASP.NET 1.0 fue liberado el 5 de Enero de 2002 como parte de la versión 1.0 del .NET Framework. Incluso antes de su liberación, docenas de libros habían sido escritos sobre ASP.NET y Microsoft lo promociono fuertemente como parte de su plataforma para servicios web.

Hasta la actualidad, ASP.NET, es una de las más fuertes herramientas usadas para el desarrollo de aplicaciones web.

1.1.4 ASP VS ASP.NET

ASP.NET es un cambio radical tanto en la forma de programación como en la de trabajo interno, es una nueva plataforma de desarrollo web. Ya no es un lenguaje interpretado como las ASP sino que es un lenguaje compilado basado en el entorno .NET. El modelo de objetos de ASP.NET es bastante distinto del modelo de ASP, es más estructurado y orientado a objetos.

ASP y ASP.NET son diferentes aunque pueden convivir juntas porque la dll que ejecuta las ASP (asp.dll) no interviene en .NET. ASP puede trabajar perfectamente en un entorno .NET, ambos no son del todo compatibles, aunque su sintaxis sea muy parecida, por ejemplo un paso directo de parámetros desde páginas asp a páginas aspx no es posible, pero si es posible tener un proyecto que este desarrollado en ASP y ASP.NET.

También ha cambiado el acceso a datos, en ASP.NET tenemos ADO.NET un nuevo modelo de objetos para acceso a datos, no disponible para asp. Por supuesto desde ASP.NET podemos utilizar el antiguo modelo de objetos de ADO (versión 2.7) pero seria desaprovechar el potencial del nuevo lenguaje. De hecho pueden utilizarse los componentes COM (Component Object Model) dentro de .NET y así poder utilizar dll que ya tenemos creadas con los nuevos desarrollos en .NET.

La arquitectura de ASP.NET ha sido rediseñada desde cero para facilitar al máximo la creación de aplicaciones Web dinámicas, y el modo en que estructuramos el código ASP.NET también promueve una mejor reutilización y compartición. Mientras que las aplicaciones tradicionales ASP utilizan la extensión .asp, las páginas ASP.NET utilizan la extensión .aspx.

El modelo de ASP.NET, con muchas características nuevas, permite a los desarrolladores escribir código más limpio y más fácil de reutilizar y compartir, incrementando el rendimiento y la escalabilidad al poder acceder a lenguajes compilados, no interpretados, a diferencia de ASP que tenía varios limitantes en este sentido.

En el modelo de desarrollo web basado en páginas activas, la programación ASP actual tiene diversas limitaciones:

- Para que todo ocurra en una página web, es habitual escribir una gran cantidad de código para resolver necesidades sencillas. ASP.NET incorpora un modelo declarativo a la programación web: los controles de servidor funcionan en una página Web simplemente declarándolos. Cuando se carga la página ASP.NET, se instancian los controles listados en la página ASP y es responsabilidad del control emitir código HTML que el navegador pueda entender.
- ASP clásico es un tanto desorganizado. En una página ASP podemos incluir casi todo: HTML plano, código script, objetos COM y texto. No hay una distinción formal entre el contenido de una página y su comportamiento. ASP.NET impone un cierto orden sobre el modelo de programación estándar ASP. En cierto modo, esta "desorganización" puede evitarse fácilmente usando el sentido común y algunas de las nuevas tecnologías. Por ejemplo, podemos escribir en nuestras páginas ASP únicamente código VBScript. Dicho código generaría un mensaje XML, que luego seria interpretado por un archivo XSLT. De esta forma conseguimos evitar el llamado "código spaguetti", aumentando la claridad del código y la velocidad de ejecución de las páginas ASP.
- Con ASP utilizamos lenguajes de scripting no tipados como VBScript o JScript. Podemos instalar otros motores de scripting que impongan verificación de tipos. ASP.NET claramente separa la porción basada en script de una página web de su contenido.
- En ASP.Net, se pueden crear rápidamente aplicaciones web, basándose en los controles incluidos en el framework o muchos gratuitos que hay

en la red, ocultando el código Ej.: Puedes crear fácilmente un grid o tabla, y ésta se auto-ordena, página, etc., obteniendo sus datos desde cualquier base de datos. Incluye una gran herramienta para la construcción de reportes, y esto incluyen medios automáticos para exportarlos a XLS o PDF, y de igual forma incluye CristalReport. Además permite separar completamente la interfaz de la lógica de negocio. Excelente para desarrollo de aplicaciones multicapas.

- Es muy sencilla la creación de páginas con AJAX, sólo incluyendo unos controles, así como descargar gratuitamente el ToolKit de ASP.Net Ajax.

1.1.5 Solución de problemas obtenidos al desarrollar aplicaciones web en ASP mediante el uso de la Tecnología ASP.NET

Mantenimiento

Las aplicaciones Cliente/Servidor en ASP son difíciles de mantener. El código ASP mezclado con la interfaz de usuario hace que muchas veces se pierda demasiado tiempo actualizando toda la aplicación, no pudiendo trabajar simplemente con el núcleo del código. ASP.Net nos permite separar la interfaz y el código.

Creación de Código

Al Usar ASP la mayoría de todo lo que funciona en una página web (clases, interfaz, controles de servidor, conexiones, etc.), debe ser creado por el desarrollador. Cada formulario que ingresa datos a una base de datos conlleva varias líneas de código, obligando al desarrollador a generar desde cero cada aplicación. El rico entorno de .NET Framework, brinda una extensa cantidad de controles predefinidos, que permiten crear aplicaciones potentes, escribiendo pocas líneas de código. [www.001]

Limitación de Lenguajes

ASP.NET incorpora soporte nativo para C#, Visual Basic y JScript. Logrando así dejar atrás las limitaciones ASP que sólo permitía código en VBScript y JScript.

1.1.6 Introducción al .net Framework

Elementos principales .NET Framework:

- CLR (Common Language Runtime)
- El conjunto de clases del .NET Framework
- ASP.NET
- Los servicios Web
- Remoting
- Windows Forms

El CLR es el motor de ejecución de las aplicaciones .NET, lo que en Java sería la máquina virtual de Java, este motor se encarga de ejecutar todo el código .NET para ello a de ser en dicho lenguaje. El CLR es el encargado de convertir este lenguaje intermedio en leguaje máquina del procesador, esto normalmente se hace en tiempo real por un compilador JIT (Just-In-Time) que lleva incorporado el CLR.

El conjunto de clases del .NET Framework es un rico conjunto de clases, interfaces, tipos que simplifican y optimizan el desarrollo de aplicaciones .NET además de proporcionar acceso a la funcionalidad del sistema.

ASP.NET es la parte del .NET Framework dedicada al desarrollo web. A través del servidor web (IIS) nuestras aplicaciones ASP.NET se ejecutarán bajo el CLR y podremos usar el conjunto de clases del .NET Framework para desarrollarlas, obteniendo así una versatilidad y una potencia nunca antes conseguida en las aplicaciones ASP.

Servicios Web nos permiten comunicarnos a través de Internet entre diferentes ordenadores, incluso entre distintos sistemas.

Remoting nos permite tener objetos en máquinas remotas e invocarlos desde otras máquinas. Y las Windows Forms, parte del .NET Framework que permite crear aplicaciones en el más clásico de los sentidos. [www.002]

1.1.7 Ventajas al desarrollar Aplicaciones Web a través de la Tecnología ASP.Net

El criterio de administración y uso actual de los sistemas ya no se ven estancados en el simple hecho de uso mono-usuario o el limitante a una sola terminal de uso.

La creciente revolución tecnológica y brote de oportunidades en todo el mundo, nos ven obligados a tener siempre a la distancia de un clic nuestras aplicaciones más importantes.

La Tecnología .Net nos brinda la oportunidad de ampliar nuestros horizontes en cuanto al diseño de aplicaciones Web mediante el uso de ASP.NET; se ejecuta a través de un servidor web IIS usando el conjunto de de clases del .NET Framework, obteniendo así una versatilidad y una potencia nunca antes conseguida en las aplicaciones ASP.

Con el uso de ASP.NET nosotros podemos tener un modelo de programación más unificado, la mejora de la seguridad y una nueva forma de escribir potentes aplicaciones Web completas.

Con un conjunto de características nuevas, ofrece código más fácil de escribir, reutilizar y compartir. ASP.NET mejora el rendimiento y escalabilidad ofreciendo acceso a lenguajes compilados; el desarrollo es más intuitivo gracias a los Formularios Web, y su base orientada a objetos facilita la reutilización. Se soportan eventos, controles y funciones de caché.

Los Controles Web, las técnicas de enlace de datos, los Formularios Web y los Servicios Web permiten sacar partido a las librerías de clases del .NET Framework y permiten exponer funciones de negocio a través del Web, ofreciendo nuevas oportunidades de desarrollo.

1.1.8 Principios de ASP.NET

Los principios de ASP.NET son:

- Facilidad de desarrollo
- Alto rendimiento y escalabilidad
- Mejorada fiabilidad
- Fácil distribución e instalación

Facilidad de Desarrollo

ASP.NET usa "Server controls", que permiten a modo de etiquetas HTML tener controles manejados por el servidor que identifican el navegador usado adaptándose para cada navegador, convirtiendo tareas tediosas como la validación de datos en fáciles y sencillas.

Existe la posibilidad de elección del lenguaje de programación que van desde un notepad hasta C#, VB.NET, pasando por la gratuita Web Matrix.

Alto Rendimiento y Escalabilidad

El código es compilado para ser ejecutado en el CLR.

Se puede optar por tenerlo en el servidor pre compilado o dejar que el servidor lo compile la primera vez que lo ejecute. El resultado es de 3 a 5 veces superior en velocidad que las antiguas páginas ASP.

El uso adecuado del potente caché incorporado aumenta considerablemente el rendimiento y la escalabilidad de la aplicación. Esta caché nos permitirá registrar, desde páginas completas a partes completas, pasando por conjuntos de datos extraídos de la base de datos.

Mejora de la Fiabilidad

ASP.NET es capaz de detectar perdidas de memoria, problemas con bloqueos y protección ante colapsos del sistema.

Es capaz de detectar aplicaciones web que pierden memoria, arrancando otro proceso limpio con una nueva instancia de la aplicación para cerrar la que pierde memoria liberando así la memoria perdida.

Fácil Distribución e Instalación

Las aplicaciones ASP.NET se instalan fácilmente, tan solo copiando los ficheros que la componen.

No es necesario registrar ningún componente, tan solo copiar los ficheros al web.

Se puede recompilar la aplicación o enviar nuevos ficheros sin necesidad de reiniciar la aplicación ni el servidor web. [www.003]

1.1.9 Características de ASP.NET

1.1.9.1 Eficiencia

Para asegurarse un óptimo rendimiento, el CLR compila, en algún punto, todos los códigos de aplicaciones en códigos naturales de máquina.

Esta conversión puede hacerse en el momento en que se ejecuta la aplicación (método por método), o cuando se instala la aplicación por primera vez. El proceso de compilación hará uso automáticamente de todas las características del microprocesador, disponibles en diferentes plataformas.

1.1.9.2 Soporte de Lenguajes

Con ASP.NET no estamos obligados a trabajar con VBScript o JScript.

ASP.NET soporta la programación en lenguajes potentes cómo, VisualBasic.Net (VB) y C#.

1.1.9.3 Administración del Estado

Las aplicaciones ASP.NET son alojadas en un servidor web y se tiene acceso a ellas mediante el protocolo sin estado HTTP, que no guarda ninguna información sobre conexiones anteriores. Por lo tanto, si la aplicación requiere interacción entre conexiones, tiene que implementar su propia administración del estado. ASP.NET proporciona varias maneras de administrar el estado de las aplicaciones ASP.NET.

1.1.9.4 Estado de la Aplicación

El estado de la aplicación (Application state) es una colección de variables definidas por el usuario que son compartidas por todas las invocaciones de una aplicación ASP.NET. Estos son establecidas e inicializadas cuando el evento Application_OnStart se dispara en la carga de la primera instancia de las aplicaciones y están disponible hasta que la última instancia termina. Las variables de estado de la aplicación son identificadas por nombres.

1.1.9.5 Estado de la sesión

El estado de la sesión (Session state) es una colección de variables definidas por el usuario, las cuales persisten durante la sesión de un usuario. Estas variables son únicas para diferentes instancias de una sesión de usuario, y son accedidas usando la colección Session. Las variables de sesión pueden ser preparadas para ser automáticamente destruidas después de un determinado tiempo de inactividad, incluso si la sesión no ha terminado. Del lado del cliente, una sesión de usuario es identificada por una cookie o codificando el ID de la sesión en la misma URL.
ASP.NET proporciona tres modos de persistencia para variables de sesión:

InProc
Las variables de sesión son mantenidas dentro del Proceso (informática). Sin embargo, en este modo, las variables son destruidas cuando el proceso ASP.NET es reciclado o terminado.

StateServer

En este modo, ASP.NET ejecuta un servicio de Windows separado que mantiene las variables de estado. Como esta administración de estado ocurre fuera del proceso ASP.NET, tiene un impacto negativo en el rendimiento, pero permite a múltiples instancias de ASP.NET compartir el mismo estado del servidor, permitiendo que una aplicación ASP.NET pueda tener su carga balanceada y escalada en múltiples servidores. También, como el servicio de administración del estado se ejecuta independiente de ASP.NET, las variables pueden persistir a través de las finalizaciones del proceso ASP.NET.

Estado SqlServer

En este modo, las variables de estado son almacenadas en un servidor de base de datos, accesible usando SQL. Las variables de sesión pueden persistir a través de finalizaciones de procesos también en este modo.

Estado de la Vista

El estado de la vista (View state) se refiere al mecanismo de administración de estado a nivel de página, que es utilizado por las paginas HTML generadas por las aplicaciones ASP.NET para mantener el estado de los controles de los formularios web y los widgets. El estado de los controles es codificado y enviado al servidor en cada envió del formulario en un campo oculto conocido como __VIEWSTATE. El servidor envía de regreso las variables para que cuando la página será renderizada de nuevo, los controles volverán a su último estado. Del lado del servidor, la aplicación puede cambiar el estado de la vista, si los resultados del procesamiento actualizan el estado de cualquier control. El estado de los controles individuales son decodificados en el servidor, y están disponibles para su uso en ASP.NET usando la colección ViewState. [www.001]

1.1.9.6 Contenido y Código, por separado

ASP.NET separar la interfaz de usuario con el código.

El modelo Code-behind

Microsoft recomienda que para realizar programación dinámica se use el modelo code-behind, o de respaldo, que coloca el código en un archivo

separado o en una etiqueta de script especialmente diseñada. Los nombres de los archivos code-behind están basados en el nombre del archivo ASPX tales como MiPagina.aspx.cs o MiPagina.aspx.vb (esta práctica se realiza automáticamente en Microsoft Visual Studio y otras interfaces de desarrollo). Cuando se usa este estilo de programación, el desarrollador escribe el código correspondiente a diferentes eventos, como la carga de la página, o el clic en un control, en vez de un recorrido lineal a través del documento.

El modelo code-behind de ASP.NET marca la separación del ASP clásico y alienta a los desarrolladores a construir aplicaciones con la idea de presentación y contenido separados en mente. En teoría, esto permite a un diseñador web, por ejemplo, enfocarse en la creación del diseño con menos posibilidades de alterar el código de programación mientras lo hace. Esto es similar a la separación en el Modelo Vista Controlador.

Ejemplo:

```
<%@ Page Language="C#" CodeFile="EjemploCodeBehind.aspx.cs"
Inherits="SitioWeb.EjemploCodeBehind"
AutoEventWireup="true" %>
```

La etiqueta superior es colocada al inicio del archivo ASPX.

La propiedad CodeFile de la directiva @ Page especifica que archivo (.cs o .vb) contiene el código code-behind mientras que la propiedad Inherits especifica la clase de la cual deriva la pagina.

En este ejemplo, la directiva @Page está incluida en EjemploCodeBehind.aspx y el archivo: EjemploCodeBehind.aspx.cs contendrá el código para esta página:

```
Using System;
namespace SitioWeb{
public partial class EjemploCodeBehind: System.Web.UI.Page{
    protected void Page_Load(object sender, EventArgs e) {}
}
}
```

En este caso, el método Page_Load() será llamado cada vez que la pagina ASPX sea solicitada al servidor. El programador puede implementar manejadores de eventos en varias etapas del proceso de ejecución de la página. [www.001]

1.1.9.7 Compatibilidad con Navegadores ASP.NET

ASP.NET permite crear una página web que funcionará correctamente en todos los navegadores. Esta mejora está dada especialmente por los controles de servidor incluidos en ASP.NET.

1.1.9.8 Código Compilado

ASP.NET ya no interpreta el código como la hace la versión anterior de ASP. Dentro del entorno NGWS (New Generation Windows Services) el código es compilado just-in-time, logrando un enorme aumento en el rendimiento, a través de soporte nativo y servicios de caché.

1.1.9.9 Controles de Servidor

Su librería de clases es común en toda la plataforma .NET, lo que le brinda al desarrollador una herramienta ideal para crear aplicaciones multiplataforma, con un considerable ahorro de líneas de código. Los controles de servidor están divididos en dos categorías:

- Controles Web.
- Controles HTML.

Permiten crear automáticamente controles que realicen tareas importantes en el servidor como validar la entrada de formularios, verificar las capacidades de los navegadores o implementar un sistema de banners rotativos.

1.1.9.10 Controles de Usuario

ASP.NET permite la creación de componentes reutilizables a través de la creación de Controles de Usuario (User Controls). Un control de usuario sigue la misma estructura que un formulario web, excepto que los controles derivan de la clase System.Web.UI.UserControl, y son almacenados en archivos ASCX. Como los archivos ASPX, un ASCX contiene etiquetas HTML o XHTML, además de etiquetas para definir controles web y otros controles de usuario. También pueden usar el modelo code-behind.

Los programadores pueden agregar sus propias propiedades y métodos, y manejadores de eventos. Un mecanismo de eventos en burbuja proporciona la capacidad de pasar un evento disparado por un control de usuario a la página que lo contiene.

1.1.9.11 Servicios Web

ASP.NET nos permite crear y utilizar Servicios Web. Mientras que los archivos de las aplicaciones ASP.NET tienen una extensión .aspx, los Servicios Web tienen extensión .asmx. Para crear un Servicio Web, simplemente creamos un archivo con extensión .asmx incluyendo un objeto tal y como si fuese accedido directamente por clientes locales; lo marcamos con el atributo Webmethod, indicando que deseamos que esté disponible para clientes Web, lo que expone automáticamente los métodos de la clase pública como métodos del Servicio Web; lo implantamos como parte de una aplicación Web y dejamos que ASP.NET haga el resto.

1.1.9.12 Mejora de la Seguridad

ASP.NET permite distintos tipos de identificación y autentificación de usuario:

- Windows.
- Passport.
- Cookies.

ASP.NET nos permite obtener una personalización de cuentas real y la posibilidad de que el servidor ejecute código como si el usuario estuviese presente. Podemos programáticamente verificar si el usuario dispone de un rol específico y de forma condicional permitirle realizar ciertas tareas, si tiene autorización. Además, crear sistemas de autentificación basada en formularios, en los que podamos crear nuestras propias ventanas de identificación y verificación de credenciales.

1.1.9.13 Fácil Configuración e Implantación

La configuración e implantación se simplifican con el uso de archivos de configuración directamente legibles que no necesitan ser registrados. ASP.NET utiliza una arquitectura de configuración jerárquica. La información de configuración de una aplicación ASP.NET se almacena en archivos textuales de configuración en formato XML denominados config.web, que pueden ser ubicados en los mismos directorios que los archivos de aplicación. Si la aplicación tiene gran tamaño, directorios hijos pueden heredar la configuración del directorio padre a menos que sean sobre escritos por otros archivos config.web en el propio directorio hijo. En ASP.NET, todos los archivos que un servidor Web necesita están ubicados bajo la carpeta raíz del sitio Web. Es

probable que aparezcan DLLs bajo el directorio /bin, ya que la mayoría de aplicaciones ASP.NET segregan código y datos en archivos separados.
Similar a ASP, ASP.NET soporta un archivo declarativo: global.asax para las directivas de programa a nivel de aplicación, eventos, declaraciones de objetos globalmente accesibles y el estado de la aplicación.
Para implantar un sitio Web, todo lo que necesitamos hacer es copiar de un sitio a otro la carpeta raíz la aplicación Web utilizando sencillos comandos de copia de archivos, las extensiones de servidor de FrontPage, o utilidades como FTP.
ASP.NET también puede implantar una aplicación totalmente compilada. El principal beneficio radica en que ninguna parte del código fuente es visible al administrador del sitio Web.

1.1.9.14 Soporte para la Actualización "en vivo" de las Aplicaciones

En el caso de una implantación de una sede sencilla no representa mayor problema; sin embargo, si desarrollamos una aplicación en varios niveles que utilice componentes, tendremos problemas.
En un sitio Web en producción, no es tan fácil detener el servidor Web (Microsoft Hotmail, por ejemplo, tiene cerca de 5.000 servidores). Además, resulta doloroso gestionar el proceso de registro de DLLs cuando implantamos un sitio Web. Ahora, simplemente se recompila y se copia de nuevo.

1.1.9.15 Motor de Plantillas

ASP.NET 2.0 presento el concepto de página maestra (Master Page), que permiten el desarrollo de páginas basado en plantillas web. Una aplicación web puede tener una o más paginas maestras, las cuales pueden ser anidadas. Las plantillas maestras contienen controles contenedores, llamados ContentPlaceHolders para indicar donde ira el contenido dinámico, además de HTML y JavaScript que será compartido a través de las paginas hijas.
Las paginas hijas también usan esos controles ContentPlaceHolder, que deben ser relacionados con el ContentPlaceHolder de la pagina maestra que contiene a esta página hija. El resto de la página está definido por las partes compartidas de la página maestra. Todo el lenguaje de marcado y controles de servidor en la página de contenido deben ser colocadas dentro del control ContentPlaceHolder.

Cuando una solicitud es hecha por una página de contenido, ASP.NET mezcla la salida de la página de contenido con la salida de la página maestra, y envía el resultado al usuario.

La página maestra permanece completamente accesible a la página del contenido. Esto significa que la página de contenidos puede manipular los encabezados, cambiar el titulo, configurar la cache, etc. Si la pagina maestra expone propiedades públicas o métodos, el contenido de la página puede utilizar estos también.

1.1.9.16 Estructura de Directorios

En general, la estructura de directorios de ASP.NET puede ser determinada por las preferencias del desarrollador. Aparte de unos pocos nombres de directorios reservados, el sitio puede expandirse a cualquier número de directorios. La estructura es reflejada directamente en las URLs.

Los nombres de directorios especiales se los muestra en la Tabla 1.1

App_Browsers	Contiene archivos de definición específicos para navegadores.
App_Browsers	Es un directorio para códigos.
App_Data	Directorio para las bases de datos.
App_LocalResources	Contiene archivos de recursos localizados para páginas individuales del sitio.
App_GlobalResources	Contiene archivos resx con recursos localizados disponibles para cada página del sitio.
App_Themes	Usado para temas alternativos del sitio.
App_WebReferences	Usado para archivos de descubrimiento y archivos WSDL.
Bin	Contiene código compilado (.dll).

Tabla 1.1 Estructura de Directorios

App_Browsers

Contiene archivos de definición específicos para navegadores.

App_Code

Es un directorio para códigos. El servidor ASP.NET automáticamente compilara los archivos (y subdirectorios) en esta carpeta en un ensamblado que es accesible desde cualquier página del sitio.

Es típicamente usada para código de acceso a datos, código de modelo o código de negocios. También cualquier manejador http especifico para el sitio e implementación de módulos y servicios web van este directorio.

Como alternativa a utilizar App_Code el desarrollador puede optar por proporcionar un ensamblado independiente con código pre compilado.

App_Data

Directorio para las bases de datos, tales como archivos mdb de Microsoft Access y archivos mdf de Microsoft SQL Server.

Este directorio es usualmente el único con permisos de escritura en la aplicación.

App_LocalResources

Contiene archivos de recursos localizados para páginas individuales del sitio.

App_GlobalResources

Contiene archivos resx con recursos localizados disponibles para cada página del sitio. Este es donde el desarrollador ASP.NET típicamente almacenara mensajes que serán usados en más de una página.

App_Themes

Usado para temas alternativos del sitio.

App_WebReferences

Usado para archivos de descubrimiento y archivos WSDL para referencias a servicios web para ser consumidos en el sitio.

Bin

Contiene código compilado (archivos .dll) para controles, componentes, y otro código que pueda ser referenciado por la aplicación.

Cualquier clase representada por código en la carpeta Bin es automáticamente referenciada en la aplicación. [www.001]

1.2. Web Forms ASP.NET

Web Forms en ASP.NET, es un modelo de programación escalable de Common Language Runtime que puede utilizarse en el servidor para generar páginas Web dinámicamente.

Proporciona:

- Capacidad para crear y utilizar controles de la interfaz de usuario reutilizables que puedan encapsular funcionalidades comunes y, así, reducir la cantidad de código que tiene que escribir el programador de una página.
- Capacidad para que los programadores puedan estructurar limpiamente la lógica de la página de forma ordenada (no revuelta).
- Capacidad para que las herramientas de desarrollo proporcionen un fuerte soporte de diseño WYSIWYG (Lo que ve es lo que se imprime) a las páginas (el código ASP existente es opaco para las herramientas).

1.2.1. Creación de Web Forms

Las páginas de formularios Web de ASP.NET consisten en archivos de texto con una extensión de nombre de archivo .aspx. Pueden implementarse por todo un árbol de directorio raíz virtual IIS. Cuando un explorador cliente solicita recursos .aspx, el motor en tiempo de ejecución de ASP.NET analiza y compila el archivo de destino en una clase de .NET Framework. Esta clase puede utilizarse, a continuación, para procesar de forma dinámica las solicitudes entrantes. (Debe observarse que el archivo .aspx sólo se compila la primera que se tiene acceso al mismo; la instancia de tipo compilada se vuelve a utilizar en múltiples solicitudes).

Una página de ASP.NET puede crearse tomando simplemente un archivo HTML existente y cambiando la extensión del nombre de archivo a .aspx, sin necesidad de modificar el código).

1.2.2. Uso de Bloques Representativos ASP <% %>

ASP.NET tiene compatibilidad sintáctica con páginas ASP existentes. Incluye compatibilidad para bloques de representación de código <% %> que pueden unirse con contenido HTML dentro de un archivo .aspx.

A diferencia de ASP, el código que se utiliza en los bloques <% %> anteriores realmente se compila, no se interpreta mediante un motor de secuencias de comandos. Esto produce un mejor rendimiento de la ejecución en tiempo de ejecución.

Los programadores de páginas ASP.NET pueden utilizar bloques de código <% %> para modificar dinámicamente resultados HTML más de lo que se puede actualmente con ASP.

Ejemplo:

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeFile="controldiar.aspx.cs" Inherits="ControlDoc_controldiar" %>
```

1.2.3. Listas, Datos y Enlace de Datos

ASP.NET incluye un conjunto integrado de controles de lista y cuadrícula de datos. Se pueden utilizar para proporcionar una interfaz de usuario personalizada basada en consultas a una base de datos o a otro origen de datos.

Ejemplo:

```
<asp:GridView ID="GridView1" runat="server" AllowSorting="True"
AutoGenerateColumns="False"
DataKeyNames="Codigo,Docente,Materia,Dia,H_Inicio,Num_horas_dictadas,
Semestre,Asistencia,Hora_Inicio"DataSourceID="SqlDataSource2"
CellPadding="4" ForeColor="#333333" GridLines="None">
<Columns><asp:BoundField
DataField="Docente" HeaderText="Docente" SortExpression="Docente" />
```

El control <asp:GridView ID="GridView1"/> proporciona una forma sencilla de visualizar rápidamente resultados de datos mediante una interfaz de usuario tradicional de control de cuadrícula. Como alternativa, los programadores de ASP.NET pueden utilizar otros controles con los cuales se puedan interactuar diversas presentaciones según sean necesarias.

1.2.4. Formularios Web de Código Subyacente

ASP.NET admite dos métodos para crear páginas dinámicas. El primero es el método en el que el código de página se declara físicamente en el archivo .aspx

de origen. La forma alternativa (conocida como método de código subyacente) permite que el código de página esté más claramente separado del contenido HTML en un archivo completamente independiente (archivos .aspx.cs).

Ejemplo:

```
protected void Calendar2_SelectionChanged(object sender, EventArgs e)
{
  TextBox4.Text=Calendar2.SelectedDate.ToShortDateString();
}
```

1.3. Controles de Servidor

1.3.1. Introducción a Controles de Servidor ASP.NET

Además de utilizar bloques de código <% %> para programar contenido dinámico, los programadores de páginas ASP.NET pueden utilizar controles de servidor ASP.NET para programar páginas Web. Los controles de servidor se declaran dentro de un archivo .aspx mediante etiquetas personalizadas o etiquetas HTML intrínsecas que contienen un valor de atributo runat="server". Las etiquetas HTML intrínsecas las controla uno de los controles del espacio de nombres System.Web.UI.HtmlControls. A cualquier etiqueta que no esté explícitamente asignada a uno de los controles se le asigna el tipo de System.Web.UI.HtmlControls.HtmlGenericControl.

Ejemplo:

```
<form id="form1" runat="server">
```

En tiempo de ejecución, este control de servidor genera contenido HTML automáticamente.

Debe tenerse en cuenta que estos controles de servidor mantienen automáticamente cualquier valor introducido por el cliente entre acciones de ida y vuelta al servidor. El estado del control no se almacena en el servidor, sino en un campo del formulario <input type="hidden"> que recibe acciones de ida y vuelta entre solicitudes. Hay que tener en cuenta que no se necesita ninguna secuencia de comando en el cliente.

Además de admitir controles estándar de entrada HTML, ASP.NET permite a los programadores utilizar controles personalizados enriquecidos en las páginas. [www.004]

1.3.2. Declaración de Controles de Servidor

Los controles de servidor ASP.NET se identifican en una página mediante etiquetas declarativas que contienen un atributo runat="server".

Ejemplo:

```
<asp:Button ID="Button8" runat="server" Font-Bold="True"
OnClick="Button8_Click" Text="Actualizar Informes" Width="177px" />
```

1.3.3. Manipulación de Controles de Servidor

Se puede identificar mediante programación un control de servidor individual de ASP.NET dentro de una página proporcionándolo con un atributo id. Se puede utilizar la referencia id para manipular mediante programación el modelo de objeto del control de servidor en tiempo de ejecución.

Ejemplo:

```
<asp:Label ID="Label1" runat="server" Font-Bold="True" Font-
Italic="True" ForeColor="Red" Text="Label" Visible="False"
Width="750px"></asp:Label>
```

1.3.4. Control de Eventos de Controles de Servidor

Los controles de servidor ASP.NET puede exponer un modelo de objeto con propiedades, métodos y eventos. Los programadores de ASP.NET pueden utilizar este modelo de objeto para modificar e interactuar limpiamente con la página.

En el siguiente ejemplo se muestra cómo un programador de páginas ASP.NET puede controlar el evento OnClick desde el control

Ejemplo:

```
<asp:Button ID="Button8" runat="server" Font-Bold="True"
OnClick="Button8_Click" Text="Actualizar Informes" Width="177px" />
```

1.3.5. Utilización de Controles de Servidor Personalizados

ASP.NET incluye 45 controles de servidor integrados que se pueden utilizar fuera del cuadro. Además de utilizar los controles integrados de ASP.NET, los programadores también pueden utilizar controles desarrollados por otros fabricantes.

El código que escribe el programador de una página es idéntico, sin importar si se empleo un explorador de alto o bajo nivel para tener acceso a la página

1.3.6. Controles de Validación de Formulario

El marco de trabajo de la página de formularios Web de ASP.NET proporciona un conjunto de controles de servidor de validación que proporcionan a su vez un modo sencillo a la vez que potente de comprobar errores en los formularios de entrada y, en caso necesario, mostrar mensajes al usuario.
Los controles de validación se agregan a una página ASP.NET con otros controles de servidor. Existen controles para tipos concretos de validación, como la comprobación de intervalos o la coincidencia de modelos, además de RequiredFieldValidator, que se asegura de que un usuario omita un campo de entrada.
Ejemplo:

```
<asp:requiredfieldvalidator runat=server>
```

Debe observarse que los controles de validación disponen de compatibilidad con clientes de alto y bajo nivel. Los exploradores de alto nivel realizan la validación en el cliente (mediante JavaScript y DHTML) y en el servidor. Los exploradores de bajo nivel sólo realizan la validación en el servidor. El modelo de programación de los dos escenarios es idéntico.
Debe observarse que los programadores de páginas ASP.NET pueden activar opcionalmente la propiedad Page.IsValid en tiempo de ejecución para determinar si todos los controles de validación de una página son válidos en ese momento. Esto proporciona un modo simple de determinar si se continúa con la lógica empresarial o no.

1.4. Desplazamiento de Páginas

Desplazamiento por múltiples páginas es un escenario habitual en prácticamente todas las aplicaciones Web. Por ejemplo control <asp:hyperlink runat=server> para desplazarse a otra página (pasando parámetros de cadenas de consulta personalizadas por el camino).
Ejemplo:

```
<asp:LinkButton ID="LinkButton1" runat="server" ForeColor="#FFC080"
PostBackUrl="~/index.aspx">VOLVER AL INDICE</asp:LinkButton>
```

No todos los escenarios de desplazamiento de páginas se inician a través de hipervínculos en el cliente. Las redirecciones y desplazamientos en el cliente

también puede iniciarlas desde el servidor un programador de páginas ASP.NET llamando al método Response.Redirect(url). Esto suele realizarse cuando se necesita la validación en el servidor para alguna entrada del cliente antes de que el desplazamiento sea realmente efectivo.

Ejemplo:

```
protected void Button2_Click(object sender, EventArgs e)
{
   Response.Redirect("Abogado.aspx");
}
```

1.5. Administración de Controles de Usuario

Además de los controles de servidor integrados que proporciona ASP.NET, se pueden definir fácilmente controles propios mediante las mismas técnicas de programación que ya conoce el usuario para escribir páginas de formularios Web. De hecho, con sólo unas pocas modificaciones, se puede volver a utilizar casi cualquier página de formularios Web en otra página como control de servidor.

Una página de formularios Web utilizada como un control de servidor recibe el nombre de control de usuario para abreviar. Como convención, se utiliza la extensión .ascx para indicar dichos controles. De esta forma, se evita que el archivo del control de usuario pueda ejecutarse como una página de formularios Web independiente (más adelante se verá brevemente que existen unas pocas diferencias, aunque importantes, entre un control de usuario y una página de formularios Web). Los controles de usuario se incluyen en una página de formularios mediante una directiva Register:

```
<%@ Register TagPrefix="Acme" TagName="Message"
Src="pagelet1.ascx" %>
```

TagPrefix

Determina un único espacio de nombres para el control de usuario (de forma que los controles de múltiples usuarios con el mismo nombre pueden diferenciarse entre sí).

TagName

Es el único nombre del control de usuario (se puede elegir cualquier nombre).

Src

Es la ruta de acceso virtual al control de usuario, por ejemplo, "MyPagelet.ascx" o "/MyApp/Include/MyPagelet.ascx". Después de registrar el control de usuario, se puede colocar la etiqueta de control de usuario en la página de formularios

Web del mismo modo que se haría para un control de servidor ordinario incluido el atributo runat="server". [www.004]

1.5.1. Propiedades de Control de Usuario

Cuando se trata a una página de formularios Web como a un control, los métodos y campos públicos de dicho formulario Web se ascienden a propiedades públicas (es decir, a atributos de etiqueta) y los métodos del control también.

Además de ascender los campos públicos a propiedades de control, se puede utilizar la sintaxis de propiedades. La sintaxis de la propiedad tiene la ventaja de poder ejecutar código cuando se establecen o se recuperan las propiedades.

1.5.2. Encapsulación de Eventos en un Control de Usuario

Un control de usuario puede controlar los propios eventos mediante la encapsulación de algo de lógica de la página de formularios Web contenedora.

1.5.3. Creación de Controles de Usuario Mediante Programación

Al igual que se pueden crear mediante programación controles de servidor ordinarios, también se puede con los controles de usuario. El método LoadControl de la página se utiliza para cargar el control de usuario pasando la ruta de acceso virtual al archivo de código fuente del control de usuario.

Ejemplo:

```
Dim c1 As Control = LoadControl("pagelet7.ascx")
CType(c1, (Pagelet7VB)).Category = "business"
Page.Controls.Add(c1)
```

El tipo de control de usuario viene determinado por un atributo ClassName de la directiva Control. Por ejemplo, a un control de usuario guardado con el nombre de archivo "pagelet7.ascx" se le asigna el tipo inflexible "Pagelet7CS" de la siguiente manera:

```
<%@ Control ClassName="Pagelet7CS" %>
```

Puesto que el método LoadControl devuelve un tipo de System.Web.UI.Control, debe convertirse al tipo inflexible adecuado para establecer propiedades individuales del control. Finalmente, el control de usuario se agrega a la colección ControlCollection de la página base.

El tipo inflexible de un control de usuario está disponible para la página de formularios Web contenedora sólo si se incluye una directiva Register para el

control de usuario (incluso si no hay realmente ninguna etiqueta de control de usuario declarada).

1.5.4. Estilos de Control

El Web es un entorno flexible para interfaces de usuarios con variaciones extremas en la apariencia y sensación de diferentes sitios Web. Mediante hojas de estilo en cascada (CSS) .Todos los controles de servidor HTML de ASP.NET y los controles de servidores Web se han diseñado para proporcionar compatibilidad de primera clase con los estilos de CSS.

1.5.4.1. Aplicación de Estilos a Controles HTML

Las etiquetas HTML estándar admiten CSS a través de un atributo de estilo que se puede establecer en una lista de pares atributo-valor delimitada por punto y coma.

Todos los controles de servidor HTML de ASP.NET pueden aceptar estilos exactamente del mismo modo que las etiquetas HTML estándar.

Ejemplo:

```
<div id="Layer1" style="position:absolute; left:13px; top:3px;
width:983px; height:51px; z-index:1">
```

CSS también define un atributo de clase que se puede establecer en una definición de estilo CSS incluida en una sección <style>...</style> del documento. Los atributos de clase facilitan la definición de estilos y la aplicación de los mismos a varias etiquetas sin tener que volver a definir el mismo estilo. Los estilos de controles de servidor HTML también se pueden establecer de esta forma, como se demuestra en el siguiente ejemplo.

Ejemplo:

```
<style type="text/css">
```

Cuando se analiza una página ASP.NET, la información de estilo se llena en una propiedad Style (de tipo CssStyleCollection) de la clase System.Web.UI.HtmlControls.HtmlControl. Básicamente, esta propiedad consiste en un diccionario que expone los estilos del control como colección de valores indizada por cadenas para cada clave de atributo de estilo.

Por ejemplo, se puede utilizar el siguiente código para establecer y, en consecuencia, recuperar el atributo de estilo width en un control de servidor HtmlInputText.

Ejemplo:

```
<script language="VB" runat="server" >
   Sub Page_Load(Sender As Object, E As EventArgs)
      MyText.Style("width") = "90px"
      Response.Write(MyText.Style("width"))
   End Sub
</script>
<input type="text" id="MyText" runat="server"/>
```

1.5.4.2. Aplicación de Estilos a Controles de Servidor Web

Los controles de servidor Web proporcionan un nivel adicional de compatibilidad con estilos mediante la adición de varias propiedades con establecimiento inflexible de tipos para la configuración del estilo habitual, como el color de fondo y de primer plano, el nombre y tamaño de fuente, el ancho, el alto, etc. Estas propiedades de estilo representan un subconjunto de comportamientos de estilo disponible en HTML y se representan como propiedades "planas" expuestas directamente en la clase base System.Web.UI.WebControls.WebControl. La ventaja de utilizar estas propiedades es que proporcionan comprobación de tipo en tiempo de compilación y finalización de instrucciones en herramientas de programación como Microsoft Visual Studio .NET.

El espacio de nombres System.Web.UI.WebControls incluye una clase base Style que encapsula atributos de estilo comunes (clases de estilo adicionales, como TableStyle y TableItemStyle, heredadas desde esta clase base común). Numerosos controles de servidor Web exponen propiedades de este tipo para especificar el estilo de elementos de procesamiento individuales del control. Por ejemplo, el control WebCalendar expone muchas de esas propiedades de estilo: DayStyle, WeekendDayStyle, TodayDayStyle, SelectedDayStyle, OtherMonthDayStyle y NextPrevStyle. Se pueden establecer propiedades individuales de estos estilos mediante la sintaxis de sub propiedad PropertyName-SubPropertyName.

Una sintaxis ligeramente diferente permite que se declare cada propiedad Style como un elemento secundario anidado en etiquetas de control de servidor Web.

```
<ASP:Calendar ... runat="server">
   <TitleStyle BorderColor="darkolivegreen" BorderWidth="3"
        BackColor="olivedrab" Height="50px" />
</ASP:Calendar>
```

Como sucede con los controles de servidor HTML, se pueden aplicar estilos a controles de servidor Web mediante una definición de clase CSS.
Si se establece un atributo en un control de servidor que no se corresponde con ninguna propiedad con establecimiento inflexible de tipos, el atributo y el valor se llenan en la colección Attributes del control.
De forma predeterminada, los controles de servidor procesarán estos atributos no modificados en el HTML devuelto al cliente del explorador solicitante. Esto significa que los atributos de estilo y de clase se pueden establecer directamente en controles de servidor Web en vez de utilizar las propiedades con establecimiento inflexible de tipo. Mientras que esto requiere entender algo del procesamiento real del control, también puede constituir una forma flexible de aplicar estilos. Resulta especialmente útil con los controles estándar de entrada de formulario. [www.004]

Los estilos de controles de servidor Web también se pueden establecer mediante programación con el método ApplyStyle de la clase base WebControl, como en el código que se muestra a continuación.

```
<script language="VB" runat="server">
   Sub Page_Load(Src As Object, E As EventArgs)
      Dim MyStyle As New Style
      MyStyle.BorderColor = Color.Black
      MyStyle.BorderStyle = BorderStyle.Dashed
      MyStyle.BorderWidth = New Unit(1)
      MyLogin.ApplyStyle (MyStyle)
      MyPassword.ApplyStyle (MyStyle)
      MySubmit.ApplyStyle (MyStyle)
   End Sub
</script>
Login: <ASP:TextBox id="MyLogin" runat="server" />/<p/>
Password: <ASP:TextBox id="MyPassword" TextMode="Password"
runat="server" />
View:  <ASP:DropDownList id="MySelect" runat="server">  ...
</ASP:DropDownList>
```

1.6. Controles de Validación

El marco de trabajo de los formularios Web incluye un conjunto de controles de servidor de validación que proporcionan un modo sencillo a la vez que potente de comprobar errores en los formularios de entrada.
Los controles de validación funcionan con un subconjunto limitado de controles de servidor HTML y Web.
En la Tabla 1.2 se detalla los principales controles.

Nombre del control	Descripción
RequiredFieldValidator	Se asegura de que el usuario no omita ninguna entrada.
CompareValidator	Compara una entrada de usuario con un valor constante o un valor de propiedad de otro control mediante un operador de comparación.
RangeValidator	Comprueba que una entrada de usuario se encuentra entre los límites superior e inferior especificados. Se pueden comprobar los intervalos entre pares, caracteres alfabéticos o fechas. Los límites se pueden expresar como constantes.
RegularExpressionValidator	Comprueba que la entrada coincide con un patrón definido por una expresión regular. Este tipo de validación permite comprobar secuencias de caracteres previsibles, como las de los números de la seguridad social, las direcciones de correo electrónico, los números de teléfono y los códigos postales.
CustomValidator	Comprueba la entrada del usuario mediante lógica de validación que codifica el usuario. Este tipo de validación permite comprobar los valores derivados en tiempo de ejecución.
ValidationSummary	Muestra los errores de validación en forma de resumen para todos los validadores de la página.

Tabla 1.2 Controles de Validación

1.6.1. Validación en el Cliente

Los controles de validación siempre realizan una comprobación de validación en el código del servidor. No obstante, si el usuario trabaja con un explorador que admite DHTML, los controles de validación también pueden realizar validaciones mediante secuencias de comandos del cliente. Con validación en el cliente, se detectan algunos errores en el cliente cuando se envía el formulario al servidor. Si resulta que alguno de los validadores es un error, se cancela el envío del formulario al servidor y se muestra la propiedad Text del validador. Esto permite al usuario corregir lo escrito antes de enviar el formulario al servidor. Los valores de campo se revalidan en cuanto el campo que contiene el error pierde el foco, proporcionando así al usuario una rica experiencia de validación interactiva.

Debe observarse que el marco de trabajo de la página de formularios Web siempre realiza la validación en el servidor, incluso cuando ya se ha realizado en el cliente. De esta forma, se evita que los usuarios puedan omitir la validación mediante la suplantación de otro usuario o de una transacción previamente aprobada.

1.6.2. Visualizar Errores de Validación

Cuando se procesa la entrada del usuario (por ejemplo, cuando se envía el formulario), el marco de trabajo de la página de formularios Web pasa la entrada del usuario al control o controles de validación asociados. Los controles de validación comprueban la entrada del usuario y establecen una propiedad para indicar si la entrada pasó la prueba de validación. Tras haber procesado todos los controles de validación, se establece la propiedad IsValid de la página; si cualquiera de los controles muestra que se produjo un error en la prueba de la validación, toda la página se establecerá como no válida.

Si un control de validación es un error, un mensaje de error puede aparecer en la página por ese control de validación o en un control ValidationSummary en cualquier otra parte de la página. Se visualiza el control ValidationSummary cuando la propiedad IsValid de la página es false. Sondea cada control de validación de la página y agrega mensajes de texto expuestos por cada uno.

1.7. Acceso a Datos en el Servidor

El acceso a datos en el servidor es exclusivo en cuanto a la ausencia básica de información de estado de las páginas Web, lo que presenta algunas dificultades al intentar realizar transacciones como insertar o actualizar registros a partir de un conjunto de datos recuperados desde una base de datos.

1.7.1. Conexiones, Comandos y Conjuntos de Datos

Common Language Runtime proporciona un conjunto completo de interfaces API de acceso a datos administrados para el desarrollo de aplicaciones con un uso intensivo de datos.

Estas API ayudan a extraer los datos y a presentarlos de forma coherente sin importar el origen real (SQL Server, OLEDB, XML, entre otros).

Existen tres objetos con los que se trabaja: conexiones, comandos y conjuntos de datos.

- Una conexión representa una conexión física a algún almacén de datos, como SQL Server o un archivo XML.
- Un comando representa una directiva para recuperar (select) desde el almacén de datos o manipular dicho almacén (insert, update, delete).
- Un conjunto de datos representa los datos reales con los que trabaja una aplicación. Hay que tener en cuenta que los conjuntos de datos se encuentran siempre desconectados de la conexión de origen y el modelo de datos correspondiente, y que pueden modificarse de forma independiente. Sin embargo, los cambios que se realizan en un conjunto de datos pueden reconciliarse fácilmente con el modelo de datos originario.

Las aplicaciones Web obtienen acceso normalmente a los orígenes de datos para el almacenamiento y la recuperación de datos dinámicos. Se puede escribir código para el acceso a los datos utilizando clases del espacio de nombres System.Data (normalmente denominado ADO.NET).

ASP.NET permite realizar el enlace de datos mediante declaración. Este proceso no requiere la existencia de código para los escenarios de datos más comunes, entre los que se incluyen:

- Seleccionar y mostrar datos.
- Ordenar, paginar y almacenar datos en memoria caché.
- Actualizar, insertar y eliminar datos.
- Filtrar datos utilizando parámetros en tiempo de ejecución.
- Crear escenarios de detalles maestros utilizando parámetros.

ASP.NET incluye dos tipos de controles de servidor que participan en el modelo de enlace de datos declarativo: controles de origen de datos y controles enlazados a datos. Estos controles administran las tareas subyacentes requeridas por el modelo Web sin estado para mostrar y actualizar datos en páginas Web ASP.NET. Por tanto, no es estrictamente necesario conocer los detalles del ciclo de vida de la solicitud de página si sólo se va a realizar el enlace de datos. [www.005]

1.7.2. Controles de Origen de Datos

Los controles de origen de datos son controles ASP.NET que administran las tareas de conexión a un origen de datos y de lectura y escritura de datos. Los controles de origen de datos no representan ninguna interfaz de usuario, sino que actúan como intermediarios entre un almacén de datos en particular (como una base de datos, un objeto comercial o un archivo XML) y los demás controles de la página Web ASP.NET. Los controles de origen de datos habilitan un amplio conjunto de funciones para recuperar y modificar datos, entre las que se incluyen la consulta, la ordenación, la paginación, el filtrado, la actualización, la eliminación y la inserción. ASP.NET incluye los controles de origen de datos mostrados en la tabla 1.3:

Control de origen de datos	Descripción
ObjectDataSource	Permite trabajar con un objeto comercial u otra clase y crear aplicaciones Web basadas en objetos de nivel medio para administrar los datos.
SqlDataSource	Permite trabajar con proveedores de datos administrados de ADO.NET, que proporcionan acceso a bases de datos de Microsoft SQL Server, OLE DB, ODBC u Oracle.
AccessDataSource	Permite trabajar con una base de datos de Microsoft Access.
XmlDataSource	Permite trabajar con un archivo XML, que es especialmente útil para controles de servidor ASP.NET jerárquicos tales como el control TreeView o Menu.
SiteMapDataSource	Se utiliza con la exploración del sitio ASP.NET.

Tabla 1.3 Controles de Origen de Datos

Los controles de origen de datos también se pueden ampliar para admitir proveedores de almacenamiento y acceso a datos adicionales.

1.7.3. Controles Enlazados a Datos

Los controles enlazados a datos representan datos como marcados al explorador que realizó la solicitud. Un control enlazado a datos se puede enlazar a un control de origen de datos y buscar datos automáticamente en el momento apropiado del ciclo de vida de la solicitud de página. Los controles enlazados a datos pueden aprovechar las ventajas de las funciones proporcionadas por un control de origen de datos entre las que se incluyen la ordenación, la paginación, el almacenamiento en caché, el filtrado, la actualización, la eliminación y la inserción. Un control enlazado a datos establece una conexión con un control de origen de datos a través de su propiedad DataSourceID.
ASP.NET incluye los controles enlazados a datos que se describen en la tabla siguiente.

Controles de lista
Representa los datos en una variedad de formato de listas. Entre los controles de lista se incluyen los controles BulletedList, CheckBoxList, DropDownList, ListBox y RadioButtonList.

AdRotator
Representa los anuncios de una página como una imagen en la que los usuarios pueden hacer clic para ir a una dirección URL asociada al anuncio.

DataList
Representa los datos en una tabla. Cada elemento se representa utilizando una plantilla de elemento definida por el usuario.

DetailsView
Muestra un registro cada vez en disposición de tabla y permite editar, eliminar e insertar registros. También se puede realizar la paginación a través de varios registros.

FormView

Es similar al control DetailsView, pero permite definir una disposición de formato libre para cada registro. El control FormView es como un control DataList para un registro único.

GridView

Muestra los datos en una tabla e incluye compatibilidad para editar, actualizar, ordenar y paginar datos sin necesidad de código.

Menu

Representa los datos en un menú dinámico jerárquico que puede incluir submenús.

Repeater

Representa los datos en una lista. Cada elemento se representa utilizando una plantilla de elemento definida por el usuario.

TreeView

Representa los datos en un árbol jerárquico de nodos que se pueden expandir. [www.006]

1.7.4. Obtener Acceso a Datos Basados en SQL

Una aplicación suele necesitar realizar una o más consultas a la base de datos de SQL en lo que respecta a seleccionar, insertar, actualizar o eliminar.

Con el fin de proporcionar a la página acceso a las clases que se necesitarán para realizar un acceso a datos de SQL, hay que importar los espacios de nombres System.Data y System.Data.SqlClient a la página en cuestión.

```
<%@ Import Namespace="System.Data" %>
<%@ Import Namespace="System.Data.SqlClient" %>
```

Para realizar una consulta de selección en una base de datos de SQL, es necesario crear SqlConnection para la base de datos pasando la cadena de conexión y, a continuación, hay que construir un objeto SqlDataSource con la instrucción de consulta. Para llenar un objeto DataSet con los resultados de la consulta, hay que llamar al método Fill del comando.

La ventaja de utilizar un conjunto de datos consiste en que proporciona una vista desconectada de la base de datos. Se puede operar con un conjunto de

datos de la aplicación y, posteriormente, cotejar los cambios con la base de datos real.

En el caso de aplicaciones cuya ejecución es de larga duración, éste suele ser el mejor enfoque. En el caso de aplicaciones Web, se suelen realizar operaciones breves con cada solicitud (normalmente, sólo para visualizar los datos).

A menudo, no se necesita mantener un objeto DataSet en una serie de varias solicitudes. Para situaciones como éstas, se puede utilizar SqlDataSource.

Al ejecutar comandos que no necesitan devolución de datos, como los de insertar, actualizar y eliminar, también se utiliza SqlCommand. [www.007]

1.7.5. Selección Parametrizada de Datos

Se puede realizar una selección Parametrizada mediante el objeto SqlDataSource. En el siguiente ejemplo se muestra cómo se pueden modificar los datos seleccionados mediante el valor expuesto desde un control HtmlControl seleccionado.

SqlDataSource mantiene una colección Parameters que puede utilizarse para reemplazar identificadores variables (indicado por un símbolo "@" delante del nombre) con valores. Se agrega un nuevo objeto SqlParameter a la colección que especifica el nombre, el tipo y el tamaño del parámetro y, a continuación, se establece la propiedad Value en el valor de la selección.

Ejemplo:

```
myCommand.SelectCommand.Parameters.Add(New SqlParameter("@State",
SqlDbType.NVarChar, 2))
myCommand.SelectCommand.Parameters("@State").Value =
MySelect.Value
```

1.7.6. Inserción de Datos en una Base de Datos de SQL

Para insertar una fila en una base de datos, se puede agregar a la página un formulario de entrada sencillo y ejecutar un comando de inserción en el controlador de eventos de envío de formularios. Hay que tener en cuenta que también hay que realizar una comprobación para asegurarse de que los valores requeridos no son nulos antes de intentar insertarlos en la base de datos. Esto evita una infracción accidental de las restricciones de campo de la base de datos. El comando de inserción también se ejecuta dentro de un bloque try/catch, por si se da el caso de que ya exista la clave primaria de la fila insertada.

Ejemplo:

```
try{
SqlDataSource3.InsertCommand = "insert into CARGA_HORARIA values(" +
DropDownList1.SelectedValue + ",'" + DropDownList2.SelectedValue + "'," +
+ TextBox1.Text + ")";
    SqlDataSource3.Insert();
     TextBox1.Text = "";
     Label1.Visible = false;
     GridView1.DataBind();
 }
catch (Exception) {mensajes(1);}
```

1.7.7. Actualización de Datos en una Base de Datos de SQL

Actualizar una base de datos puede resultar a menudo complicado en aplicaciones Web. El control GridView proporciona soporte integrado para este escenario que simplifica las actualizaciones. Para permitir que se editen filas, GridView admite una propiedad EditItemIndex de tipo entero que indica qué fila de la cuadrícula se podrá editar. Cuando se establece esta propiedad, GridView procesa la fila en el índice como cuadros de entrada de texto en vez de simples etiquetas. Un valor de -1 (el predeterminado) indica que no se puede editar ninguna fila. La página puede incluir el control GridView en un formulario del servidor y obtener acceso a los datos editados a través del modelo de objeto de GridView.

Para saber qué fila se puede editar, se necesita una forma de aceptar alguna entrada del usuario relacionada con la fila que se desearía editar. GridView puede incluir una EditCommandColumn que procesa vínculos para desencadenar tres eventos especiales: EditCommand, UpdateCommand, y CancelCommand. EditCommandColumn se agrega de forma declarativa a la colección Columns de GridView, tal y como se muestra en el siguiente ejemplo.

```
<ASP:GridView id="MyGridView" runat="server"
<Columns> <asp:EditCommandColumn EditText="Edit"
 CancelText="Cancel" UpdateText="Update" /> </Columns>
 </ASP:GridView>
```

En la misma etiqueta GridView, se conectan controladores de eventos con cada uno de los comandos desencadenados desde EditCommandColumn.

El argumento GridViewCommandEventArgs de estos controladores proporciona acceso directo al índice seleccionado por el cliente que se utiliza para establecer el EditItemIndex de GridView. Hay que tener en cuenta que se necesita volver a enlazar el control GridView para que se produzca el cambio, tal y como se muestra en el siguiente ejemplo.

```
Public Sub MyGridView_Edit(sender As Object, E As
GridViewCommandEventArgs)
    MyGridView.EditItemIndex = E.Item.ItemIndex
    BindGrid()
End Sub
```

Cuando se está editando una fila de GridView, EditCommandColumn procesa los vínculos Update y Cancel. Si el cliente selecciona Cancel, EditItemIndex simplemente se vuelve a establecer en -1. No obstante, si el cliente selecciona Update, es necesario ejecutar el comando de actualizaciÃ³n en la base de datos. Para realizar una consulta de actualización se necesita conocer la clave principal de la base de datos para la fila que se desea actualizar. Para admitirlo, GridView expone una propiedad DataKeyField que puede establecerse en el nombre del campo de la clave principal. En el controlador de eventos conectado a UpdateCommand, se puede recuperar el nombre de la clave desde la colección DataKeys de GridView. Para indizar en esta colección, se utiliza el ItemIndex del evento, tal y como se muestra en el siguiente ejemplo.

```
myCommand.Parameters("@Id").Value =
MyGridView.DataKeys(CType(E.Item.ItemIndex, Integer))
```

Al final del controlador de eventos de actualización, EditItemIndex se vuelve a establecer en -1.

Los controles BoundColumn no son los únicos que se pueden establecer en la colección Columns de GridView. También se puede especificar

TemplateColumn, que proporciona un control total sobre el contenido de la columna. La plantilla sólo tiene contenido arbitrario; se puede procesar cualquier cosa que se desee, incluyendo controles de servidor, dentro de las columnas de GridView.

Para la Actualización de Datos mediante código podemos utilizar el método Update() y su declaración de Sentencia UpdateCommand.

Ejemplo:

```
try{
        SqlDataSource3.UpdateCommand = "update CARGA_HORARIA set
        CARGAH=" + TextBox1.Text + " where nro_p=" +
        DropDownList1.SelectedValue + " and nombred='" +
        DropDownList2.SelectedValue + "'"; SqlDataSource3.Update();
        TextBox1.Text = "";
        Label1.Visible = false;GridView1.DataBind();
}
catch (Exception) {mensajes(2); }
```

Realizar eliminaciones en una base de datos es muy similar a una actualización o a un comando de inserción, pero se necesita una forma de determinar la fila concreta de la cuadrícula que se va a eliminar. Otro control que puede agregarse a la colección Columns de GridView es el control ButtonColumn, que simplemente procesa un control de botón. ButtonColumn admite una propiedad CommandName que puede establecerse en Delete. En GridView, se conecta un controlador de eventos a DeleteCommand, donde se realiza la operación de eliminación. De nuevo, se utiliza la colección DataKeys para determinar la fila seleccionada por el cliente.

Para la Eliminación de Datos mediante código podemos utilizar el método Delete() y su declaración de Sentencia DeleteCommand. [www.008]

Ejemplo:

```
try{
SqlDataSource3.DeleteCommand = "delete from CARGA_HORARIA where
nro_p=" + DropDownList1.SelectedValue + " and nombred='" +
DropDownList2.SelectedValue + "'"; SqlDataSource3.Delete();
TextBox1.Text = ""; Label1.Visible = false;  GridView1.DataBind(); }
catch (Exception) {mensajes(5);}
```

Un requisito común de cualquier cuadrícula radica en la capacidad de ordenar los datos que contiene. Mientras que el control GridView no ordena explícitamente los datos, proporciona una forma de llamar a un controlador de eventos cuando el usuario hace clic en un encabezado de columna, que se puede utilizar para ordenar los datos.

Cuando la propiedad AllowSorting de GridView se establece en true, procesa hipervínculos para los encabezados de columna que desencadenan un comando Sort de nuevo en la cuadrícula.

La propiedad OnSortCommand de GridView se establece en el controlador al que se desea llamar cuando el usuario hace clic en un vínculo de columna. El nombre de la columna se pasa como una propiedad SortExpression en el argumento GridViewSortCommandEventArgs, que se puede utilizar para establecer la propiedad Sort del enlace DataView en la cuadrícula.
En el siguiente ejemplo se muestra este proceso.

```
<script>
   Protected Sub MyGridView_Sort(Src As Object, E As
GridViewSortCommandEventArgs)
      ...
      DataView Source = ds.Tables("Authors").DefaultView
      Source.Sort = E.SortExpression
      MyGridView.DataBind()
   End Sub
</script>
<form runat="server">
   <ASP:GridView id="MyGridView" OnSortCommand="MyGridView_Sort"
AllowSorting="true" runat="server" />
</form>
```

1.7.8. Relaciones Detalles Maestros

Una interfaz Web muy común es aquella en la que una fila de datos, que se puede seleccionar, desplaza al cliente hacia una página "details" que muestra información detallada acerca de la fila seleccionada.
Para conseguirlo mediante GridView, se puede agregar una columna HyperLinkColumn a la colección Columns, que especifica la página de detalles a la que se desplazará el cliente al hacer clic en el vínculo. La sintaxis de cadena de formato se utiliza para sustituir un valor del campo en este vínculo que se pasa como un argumento querystring.
En el siguiente ejemplo se muestra este proceso.

```
   <ASP:GridView id="MyGridView" runat="server">
    <Columns>
      <asp:HyperLinkColumn DataNavigateUrlField="au_id"
       DataNavigateUrlFormatString="GridView13_details.aspx?id={0}"
       Text="Get Details" /> </Columns> </ASP:GridView>
```

En la página de detalles, se recupera el argumento querystring y se realiza una selección combinada para obtener detalles a partir de la base de datos.

1.7.9. Redacción y Utilización de Procedimientos Almacenados

En general, realizar consultas ad hoc se produce a costa del rendimiento. La utilización de procedimientos almacenados puede reducir el coste de realizar importantes operaciones en la base de datos de una aplicación. Resulta fácil crear un procedimiento almacenado e incluso puede hacerse mediante una instrucción de SQL.

En el siguiente ejemplo de código se crea un procedimiento almacenado que simplemente devuelve una tabla.

```
CREATE Procedure GetAuthors AS
   SELECT * FROM Authors
   return
GO
```

También se pueden crear procedimientos almacenados que aceptan parámetros. Por ejemplo:

```
CREATE Procedure LoadPersonalizationSettings (@UserId varchar(50)) AS
   SELECT * FROM Personalization WHERE UserID=@UserId
   return
GO
```

El uso de un procedimiento almacenado de una página ASP.NET es una extensión del objeto SqlCommand.

CommandText sólo es el nombre del procedimiento almacenado en vez del texto de consulta ad hoc. Para indicar a SqlCommand que CommandText es un procedimiento almacenado, se establece la propiedad CommandType. myCommand.SelectCommand.CommandType = CommandType.StoredProcedure. [www.005]

1.7.10. Acceso a Datos Basados en XML

DataSet admite un método ReadXml que toma un objeto FileStream como parámetro. El archivo que se lee en este caso debe contener un esquema y los datos que se desean leer. DataSet espera que los datos estén en el formulario tal y como se muestra en el siguiente ejemplo.

Ejemplo:

```
<DocumentElement>
   <TableName>
      <ColumnName1>column value</ColumnName1>
      <ColumnName2>column value</ColumnName2>
      <ColumnName3>column value</ColumnName3>
      <ColumnName4>column value</ColumnName4>
   </TableName>
   <TableName>
      <ColumnName1>column value</ColumnName1>
      <ColumnName2>column value</ColumnName2>
      <ColumnName3>column value</ColumnName3>
      <ColumnName4>column value</ColumnName4>
   </TableName>
</DocumentElement>
```

También es posible leer los datos y el esquema por separado mediante los métodos ReadXmlData y ReadXmlSchema de DataSet.
Al igual que DataSet admite métodos de lectura de datos XML, también admite la escritura de los datos. [www.009]

1.8. Aplicaciones ASP.NET

1.8.1. Introducción a las Aplicaciones ASP.NET

Es como el conjunto de todos los archivos, páginas, controladores, módulos y código ejecutable que se pueden invocar o ejecutar dentro del ámbito de un determinado directorio virtual (además de sus subdirectorios) en un servidor de aplicaciones Web.

Cada aplicación ASP.NET Framework de un servidor Web se ejecuta dentro de un dominio único de aplicaciones ejecutables de .NET Framework, lo que garantiza el aislamiento de clases (no se producen conflictos de nombres o versiones), el uso seguro de recursos (se impide el acceso a determinados equipos o recursos de red) y el aislamiento de variables estáticas.

ASP.NET mantiene una agrupación de instancias HttpApplication durante el período de duración de una aplicación Web. ASP.NET asigna automáticamente

una de estas instancias para procesar cada solicitud HTTP entrante recibida por la aplicación. La instancia HttpApplication asignada en particular es responsable del proceso de la solicitud a lo largo de todo su período de duración y sólo se puede volver a utilizar después de que la solicitud se haya completado. Esto significa que el código de usuario incluido en la instancia HttpApplication no necesita ser reentrante.

1.8.2. Duración de una Aplicación ASP.NET

Una aplicación ASP.NET Framework se crea la primera vez que se realiza una solicitud al servidor; antes de ello, no se ejecuta ningún código ASP.NET. Cuando se realiza la primera solicitud, se crea una agrupación de instancias HttpApplication y se provoca el evento Application_Start. Las instancias HttpApplication procesan esta solicitud y las siguientes hasta que la última instancia termina y se provoca el evento Application_End.

Observe que los métodos Init y Dispose de HttpApplication se invocan por cada instancia y, de este modo, pueden utilizarse varias veces entre Application_Start y Application_End. Sólo se comparten estos eventos entre todas las instancias de HttpApplication en una aplicación ASP.NET. [www.010]

1.8.3. Estados en Aplicaciones ASP.NET

1.8.3.1. Introducción al Estado de Aplicaciones ASP.NET

El estado de aplicación es un repositorio de datos que está disponible para todas las clases de una aplicación ASP.NET.
El estado se almacena en la memoria del servidor y ofrece más rapidez que el almacenamiento y la recuperación de información de una base de datos. A diferencia del estado de sesión, que es específico de las sesiones de un solo usuario, el estado de aplicación se aplica a todos los usuarios y sesiones. El estado de aplicación es un lugar útil para almacenar pequeñas cantidades de datos utilizados a menudo que no cambian de un usuario a otro.
El estado de aplicación se almacena en la clase HttpApplicationState, de la cual se crea una nueva instancia la primera vez que un usuario tiene acceso a un recurso de dirección URL en una aplicación. La clase HttpApplicationState se expone a través de la propiedad Application.

El estado de aplicación almacena los datos como tipos de datos Object. Por consiguiente, es necesario convertir de nuevo los datos al tipo adecuado al recuperarlos.
El estado de aplicación se almacena en la memoria del servidor, por lo que una cantidad grande de datos puede llenarla rápidamente. Si se reinicia la aplicación, los datos de estado de aplicación se pierden. [www.011]

1.8.3.2. Opciones de Administración de Estado en el Cliente

Las opciones en el cliente para almacenar la información de página no utilizan recursos de servidor. Estas opciones normalmente ofrecen poca seguridad pero proporcionan un rendimiento rápido del servidor porque la demanda de recursos del servidor es pequeña. Sin embargo, puesto que se debe enviar información al cliente para su almacenamiento, existe un límite práctico para la información que puede almacenarse de esta forma. [www.004]
A continuación se muestran las opciones de administración de estado en el cliente que admite ASP.NET:

- Estado de vista
- Estado de control
- Campos ocultos
- Cookies
- Cadenas de consulta

1.8.3.2.1. Estado de la Vista

Las páginas de formularios Web Forms proporcionan la propiedad ViewState como una estructura integrada para conservar automáticamente valores entre varias peticiones de una misma página. El estado de vista se mantiene en la página en forma de campo oculto.
Por ejemplo, si la aplicación mantiene información específica del usuario (es decir, información que se utiliza en la página pero que no forma parte necesariamente de ningún control), se puede almacenar en el estado de vista.

Ventajas de utilizar el estado de vista

- No se requieren recursos del servidor: el estado de vista está incluido en una estructura dentro del código de la página.
- Implementación sencilla: el estado de vista no requiere que se utilice una programación personalizada

- Características de seguridad mejorada: los valores en el estado de vista están fragmentados, comprimidos y codificados para implementaciones de Unicode, por lo que son más seguros que los campos ocultos.

Desventajas de utilizar el estado de vista

- Consideraciones sobre el rendimiento: dado que el estado de vista se almacena en la propia página, el almacenamiento de valores de gran tamaño puede hacer que la página se muestre y se envíe de forma más lenta para los usuarios. Esto es especialmente relevante para los dispositivos móviles, donde el ancho de banda constituye a menudo una limitación.
- Limitaciones de dispositivos: los dispositivos móviles pueden carecer de una capacidad de memoria suficiente para almacenar una gran cantidad de datos del estado de vista.
- Posibles riesgos de seguridad: el estado de vista se almacena en uno o varios campos ocultos de la página. Aunque el estado de vista almacena los datos fragmentados, se puede manipular. Se puede ver la información de un campo oculto si se obtiene acceso al código fuente de la página directamente.

1.8.3.2.2. Estado de la Control

El marco de trabajo de la página ASP.NET proporciona la propiedad ControlState como medio para almacenar los datos de control personalizados entre los recorridos del servidor.

Ej.:

Si ha escrito un control personalizado que tiene varias fichas en las que se muestra distinta información, para que ese control funcione tal y como se espera, dicho control debe saber qué ficha se selecciona entre los recorridos de ida y vuelta.

Ventajas de utilizar el estado de control

- No se requieren recursos del servidor: de forma predeterminada, el estado de control se almacena en campos ocultos de la página.
- Confiabilidad: como el estado de control no se puede desactivar, como ocurre con el estado de vista, es un método más confiable para administrar el estado de los controles.

- Versatilidad: se pueden escribir adaptadores personalizados que controlen el modo y el lugar en el que se almacenan los datos del estado de control.

Desventajas de utilizar el estado de control

- Requiere programación: aunque el marco de trabajo de la página ASP.NET proporciona una base para el estado de control, este estado es un mecanismo personalizado de persistencia de estados. Para utilizar todas las funciones del estado de control, debe escribir el código que guarde y cargue el estado de control.

1.8.3.2.3. Campos Ocultos

Puede almacenar información específica de una página en un campo oculto de la página como forma de conservar el estado de la misma.
Si utiliza campos ocultos, es mejor almacenar sólo cantidades pequeñas de datos que cambian con frecuencia en el cliente.

Ventajas de utilizar campos ocultos

- No se requieren recursos del servidor: el campo oculto se almacena y se lee desde la página.
- Compatibilidad generalizada: la mayoría de los exploradores y dispositivos de cliente admiten formularios con campos ocultos.
- Implementación sencilla: los campos ocultos son controles HTML estándar que no requieren una lógica de programación compleja.

Desventajas de utilizar campos ocultos

- El campo oculto se puede manipular: se puede ver la información de un campo oculto si se obtiene acceso al código fuente de la página directamente.
- Puede cifrar y descifrar el contenido de un campo oculto de forma manual, pero esto requiere una codificación y una sobrecarga adicionales.
- Arquitectura de almacenamiento sencilla: el campo oculto no admite los tipos de datos enriquecidos. Los campos ocultos sólo ofrecen un campo de valor de cadena en el que se introduce la información. Para almacenar múltiples valores, debe implementar cadenas delimitadas y que el código analice estas cadenas.

- Consideraciones sobre el rendimiento: puesto que los campos ocultos se almacenan en la propia página, el almacenamiento de valores de gran tamaño puede hacer que la página se muestre y se envíe de forma más lenta a los usuarios.
- Limitaciones de almacenamiento: si la cantidad de datos de un campo oculto llega a ser muy grande, algunos servidores proxy y algunos firewall impedirán el acceso a la página que incluye estos campos ocultos.

1.8.3.2.4. Cookies

Las cookies son útiles para almacenar pequeñas cantidades de información que cambia con frecuencia en el cliente. La información se envía al servidor con la petición.

Ventajas de utilizar cookies

- Reglas de caducidad que se pueden configurar: la cookie puede caducar cuando se cierre la sesión del explorador o puede existir indefinidamente en el equipo cliente, en función de las reglas de caducidad del cliente.
- No se requieren recursos del servidor: la cookie se almacena en el cliente y el servidor la lee después de que se produzca un envío.
- Simplicidad: una cookie es una estructura ligera y basada en texto que simplemente utiliza pares clave-valor.
- Persistencia de los datos: aunque la duración de la cookie en un equipo cliente está sujeta a los procesos de caducidad de cookies y a la intervención del usuario, las cookies generalmente son el formulario más duradero para la persistencia de los datos en el cliente.

Desventajas de utilizar cookies

- Limitaciones de tamaño: la mayoría de los exploradores establecen un límite de 4096 bytes para el tamaño de una cookie.
- Rechazo configurado por el usuario: algunos usuarios desactivan la capacidad de recibir cookies del explorador o dispositivo de cliente, limitando así esta funcionalidad.
- Posibles riesgos de seguridad: las cookies se pueden manipular. Los usuarios pueden manipular las cookies de su equipo, lo que puede comprometer la seguridad o producir un error en la aplicación que

depende de la cookie. Aunque las cookies sólo son accesibles por el dominio que las envía al cliente, los piratas informáticos han hallado el modo de tener acceso a las cookies de otros dominios de un equipo de usuario. Puede cifrar y descifrar las cookies manualmente, pero esto requiere una codificación adicional y puede afectar al rendimiento de la aplicación debido al tiempo que llevan las operaciones de cifrado y descifrado.

1.8.3.2.5. Cadenas de Consulta

Una cadena de consulta es información que se anexa al final de la dirección URL de una página.

Puede utilizar una cadena de consulta para enviar datos de vuelta a la página o a otra página a través de la dirección URL. Las cadenas de consulta proporcionan una manera sencilla pero limitada de mantener cierta información de estado.

Por ejemplo, las cadenas de consulta constituyen un medio sencillo para transferir información de una página a otra, como transferir un código de producto de una página a otra en la que se procesará.

Ventajas de utilizar cadenas de consulta

- No se requieren recursos del servidor: la cadena de consulta está incluida en la solicitud HTTP de una dirección URL determinada.
- Compatibilidad generalizada: casi todos exploradores y dispositivos de cliente admiten el uso de cadenas de consulta para transferir valores.
- Implementación sencilla ASP.NET es totalmente compatible con el método de cadenas de consulta, incluidos los métodos para leer cadenas de consulta mediante la propiedad Params del objeto HttpRequest.

Desventajas de utilizar cadenas de consulta

- Posibles riesgos de seguridad: el usuario puede ver directamente la información de una cadena de consulta en la interfaz de usuario del explorador. Un usuario puede marcar la dirección URL o enviarla a otros usuarios, transfiriendo así la información de la cadena de consulta junto con la dirección URL.
- Capacidad limitada Algunos exploradores y dispositivos de cliente: establecen un límite de 2083 caracteres para la longitud de las direcciones URL. [www.012]

1.8.3.3. Opciones de Administración de Estado en el Servidor

Las opciones para almacenar la información de la página en el servidor suelen proporcionar un mayor nivel de seguridad que las opciones del cliente, pero pueden hacer un uso mayor de los recursos del servidor Web, lo que puede afectar a la escalabilidad cuando el tamaño de la información almacenada sea grande.

A continuación se muestran las opciones de administración de estado en el servidor que admite ASP.NET:

- Estado de aplicación
- Estado de sesión
- Propiedades de perfiles
- Compatibilidad con bases de datos

1.8.3.3.1. Estado de Aplicación

ASP.NET proporciona el estado de aplicación mediante la clase HttpApplicationStatefrlrfSystemWebHttpApplicationStateClassTopic como un método para almacenar información global específica de la aplicación de forma que sea visible para toda la aplicación.

Se pueden almacenar los valores específicos de la aplicación en el estado de aplicación, que administra el servidor.

Los datos que se comparten entre varias sesiones y que no experimentan cambios con frecuencia son el tipo ideal de datos que deberían insertarse en las variables del estado de aplicación.

Ventajas de utilizar el estado de aplicación

- Implementación sencilla: el estado de aplicación es fácil de usar, es coherente con otras clases de .NET Framework y los programadores de ASP lo conocen bien.
- Ámbito de la aplicación: dado que todas las páginas de una aplicación pueden tener acceso al estado de aplicación, si se almacena información en el estado de aplicación puede ocurrir que sólo exista una única copia de la información.

Desventajas de utilizar el estado de aplicación

- Ámbito de la aplicación: el ámbito del estado de aplicación también puede ser una desventaja. Las variables que se almacenan en el estado

de la aplicación son globales sólo para el proceso en el que se está ejecutando la aplicación y cada proceso de aplicación puede tener valores diferentes. Por lo tanto, no puede basarse en el estado de aplicación para almacenar valores únicos o para actualizar contadores globales en configuraciones de matrices de procesos Web y baterías de servidores Web.

- Duración limitada de los datos. como los datos globales almacenados en el estado de aplicación son volátiles, se perderán si se destruye el proceso de servidor Web que los contiene, por ejemplo si el servidor se queda bloqueado, se actualiza o se apaga.
- Requisitos de recursos: el estado de aplicación necesita memoria del servidor, lo que puede afectar al rendimiento del mismo así como a la escalabilidad de la aplicación.

Mediante un cuidadoso diseño y una buena implementación del estado de la aplicación puede mejorar el rendimiento de la aplicación Web. Por ejemplo, si se utiliza el estado de aplicación para guardar conjuntos de datos de uso común y relativamente estáticos, se puede aumentar el rendimiento del sitio mediante la reducción del número global de solicitudes a la base de datos.

Sin embargo, existe una contrapartida relativa al rendimiento. Las variables de estado de aplicación que contienen grandes bloques de información reducen el rendimiento del servidor Web a medida que aumenta la carga del mismo. La memoria que ocupa una variable almacenada en el estado de la aplicación no se libera hasta que el valor es eliminado o reemplazado. Por tanto, es mejor utilizar variables de estado de aplicación sólo con conjuntos de datos pequeños y cuya modificación sea poco frecuente.

1.8.3.3.2. Estado de Sesión

ASP.NET proporciona el estado de sesión, que está disponible mediante la clase HttpSessionState como método para almacenar información específica de la sesión de forma que sea visible sólo dentro de ésta. El estado de sesión de ASP.NET identifica las solicitudes recibidas desde el mismo explorador durante un período limitado de tiempo como una sesión y proporciona la capacidad de conservar los valores de las variables durante la duración de esa sesión.
Se pueden almacenar los valores y objetos específicos de la sesión en el estado de sesión, que el servidor administra, y que está disponible para el explorador

o el dispositivo cliente. Lo ideal es almacenar en las variables de estado de la sesión datos de duración corta, datos confidenciales para una sesión individual específica. [www.013]

Ventajas de utilizar el estado de sesión

- Implementación sencilla: el estado de sesión es fácil de usar, es coherente con otras clases de .NET Framework y los programadores de ASP lo conocen bien.
- Eventos específicos de la sesión: se pueden desencadenar eventos de administración de sesiones y utilizarlos en la aplicación.
- Persistencia de los datos: los datos guardados en variables de estado de sesión se conservan cuando se reinician los Servicios de Internet Information Server (IIS) y los procesos de trabajo sin que se pierdan datos de sesión, ya que éstos se almacenan en otro espacio de proceso.
- Escalabilidad de la plataforma: el estado de sesión se puede utilizar en configuraciones multi-sistema y multiproceso, optimizando así los escenarios de escalabilidad.
- Compatibilidad sin cookies: el estado de sesión funciona con los exploradores que no admiten las cookies HTTP, aunque se suele utilizar en combinación con éstas para proporcionar servicios de identificación de usuario a las aplicaciones Web. Si se utiliza el estado de sesión sin cookies, es necesario que el identificador de la sesión se sitúe en la cadena de consulta, que está sujeta a los problemas de seguridad que se tratan en la sección de las cadenas de consulta en este mismo tema.
- Extensibilidad: puede personalizar y ampliar el estado de sesión escribiendo su propio proveedor de estados de sesión. Los datos del estado de sesión se pueden almacenar a continuación en un formato de datos personalizado en una variedad de mecanismos de almacenamiento de datos, como una base de datos, un archivo XML o incluso un servicio Web.

Desventajas de utilizar el estado de sesión

Consideraciones sobre el rendimiento: las variables de estado de sesión permanecen en la memoria hasta que se eliminan o se reemplazan y pueden, por tanto, perjudicar al rendimiento del servidor. Las variables de estado de sesión que contienen bloques de información, como conjuntos de datos, pueden afectar negativamente al rendimiento del servidor Web a medida que aumenta la carga.

1.8.3.3.3. Propiedades de Perfiles

Permite almacenar datos específicos del usuario. Es similar al estado de sesión, salvo porque a diferencia de éste, el perfil de datos no se pierde cuando caduca la sesión de un usuario. La característica propiedades de perfiles utiliza un perfil ASP.NET, que se guarda en un formato persistente y que se asocia con un usuario específico. El perfil ASP.NET permite administrar con facilidad la información sobre el usuario sin que sea necesario crear y mantener una base de datos propia. El perfil hace que la información del usuario esté disponible mediante una API con establecimiento inflexible de tipos a la que puede obtener acceso desde cualquier punto de la aplicación.
Puede almacenar objetos de cualquier tipo en el perfil.
La característica perfil ASP.NET proporciona un sistema de almacenamiento genérico que permite definir y mantener casi todos los tipos de datos mientras éstos siguen estando disponibles de un modo seguro para los tipos.

Ventajas de utilizar propiedades de perfil

- Persistencia de los datos: Los datos situados en las propiedades de perfiles se mantienen cuando se reinician los Servicios de Internet Information Services (IIS) y el proceso de trabajo sin que se pierdan datos, porque éstos están almacenados en un mecanismo externo.
- Las propiedades de perfiles pueden preservarse en varios procesos, como en una batería de servidores Web o una matriz de procesos Web.
- Escalabilidad de la plataforma: las propiedades de perfiles se pueden utilizar en configuraciones multi-sistema y multiproceso, optimizando así los escenarios de escalabilidad.
- Extensibilidad: para utilizar las propiedades de perfiles, debe configurar un proveedor de perfiles. ASP.NET incluye una clase SqlProfileProvider que permite almacenar los datos del perfil en una base de datos de SQL, aunque también puede crear una clase propia de proveedor de perfiles que almacene los datos del perfil en un formato personalizado y disponga de un mecanismo de almacenamiento personalizado, como un archivo XML o incluso un servicio Web.

Desventajas de usar las propiedades de perfiles

- Consideraciones de rendimiento Las propiedades de perfiles normalmente son más lentas que el estado de sesión porque en lugar de

almacenar los datos en la memoria, los datos se conservan en un almacén de datos.

- Requisitos de configuración adicionales A diferencia del estado de sesión, la característica Propiedades de perfiles requiere un volumen de configuración considerable. Para utilizar las propiedades de perfiles, no sólo debe configurar un proveedor de perfiles, sino que además debe configurar previamente todas las propiedades de perfiles que desee almacenar.
- Mantenimiento de los datos: las propiedades de perfiles requieren cierto mantenimiento. Dado que los datos de los perfiles se guardan en un almacén que no es volátil, debe comprobar que la aplicación llama a los mecanismos de limpieza adecuados, mecanismos que proporciona el proveedor de perfiles, cuando los datos quedan obsoletos. [www.013]

1.8.3.3.4. Compatibilidad con Bases de Datos

En algunos casos, puede preferir utilizar la compatibilidad con las bases de datos para guardar el estado en el sitio Web. Generalmente, la compatibilidad con bases de datos se utiliza en combinación con las cookies o el estado de sesión

Por ejemplo, es muy frecuente que un sitio Web de comercio electrónico conserve la información de estado en una base de datos relacional por las razones mostradas en la tabla 1.4:

Seguridad	Registro de cuenta y contraseña.
Personalización	Distinción de cliente por el cookie.
Coherencia	Se guardan todos los registros transaccionales.
Extracción de Datos	Registro de visitas y transacciones.

Tabla 1.4 Razones para Registro de Información en una Base de Datos

Las características típicas de un sitio Web con base de datos combinado con cookies:

- Seguridad El usuario escribe un nombre de cuenta y una contraseña en una página de inicio de sesión del sitio. La infraestructura del sitio hace una consulta en la base de datos con los valores de inicio de sesión para determinar si el usuario dispone de los permisos necesarios para utilizar el sitio. Si la base de datos valida la información del usuario, el sitio Web distribuirá una cookie válida que contenga un identificador

único para ese usuario en el equipo cliente. El sitio concede el acceso al usuario.

- Personalización Si dispone de información de seguridad, el sitio puede distinguir a cada usuario leyendo la cookie en el equipo cliente. Normalmente, los sitios tienen información en la base de datos que describe las preferencias de cada usuario (identificado por un identificador único). Esta relación se conoce como personalización. El sitio puede investigar las preferencias del usuario utilizando el identificador único que contiene la cookie y, a continuación, poner el contenido y la información a disposición del usuario según sus deseos específicos e ir reaccionando a las preferencias del usuario.
- Consistencia Si ha creado un sitio Web de comercio electrónico, es posible que desee mantener los registros transaccionales de las compras de productos y servicios realizados en el sitio. Puede guardar esta información en la base de datos de un modo confiable y hacer referencia a ella mediante el identificador exclusivo del usuario. Puede utilizarla para determinar si una transacción de compra se ha completado y, en el caso de que se haya producido un error, para determinar el curso de las acciones que deben seguirse. También se puede utilizar para informar al usuario sobre el estado de un pedido realizado en el sitio.
- Extracción de Datos La información sobre el uso del sitio, los visitantes o las transacciones de un producto se pueden almacenar de forma segura en la base de datos. Por ejemplo, el departamento de desarrollo de la empresa puede utilizar los datos recogidos del sitio para determinar la línea de productos o la política de distribución para el próximo año. El departamento de marketing puede examinar información demográfica sobre los usuarios del sitio. Los departamentos de ingeniería y soporte pueden analizar las transacciones y observar las áreas donde se podría mejorar el proceso de compra. La mayoría de las bases de datos relacionales a nivel empresarial, como, por ejemplo, Microsoft SQL Server, contienen un amplio conjunto de herramientas para la mayor parte de proyectos de extracción de datos.

Si se diseña el sitio Web para que consulte repetidamente a la base de datos mediante el identificador exclusivo durante cada una de las fases generales de la situación descrita, el sitio mantiene el estado. De esta manera, el usuario percibe que el sitio le recuerda y que reacciona de manera personalizada.

Ventajas de utilizar una base de datos para mantener el estado

- Seguridad: el acceso a las bases de datos requiere una autenticación y autorización rigurosas.
- Capacidad de almacenamiento: se puede almacenar en la base de datos tanta información como se desee.
- Persistencia de los datos: la información de la base de datos se puede almacenar durante el tiempo deseado, y no está sujeta a la disponibilidad del servidor Web.
- Solidez e integridad de los datos: las bases de datos suelen incluir diversas funciones para el mantenimiento de la corrección de los datos, entre ellas desencadenadores e integridad referencial, transacciones, etc.
- Si se guarda la información de transacciones en una base de datos (en lugar de en el estado de sesión), es más rápido recuperarse de los errores.
- Accesibilidad: un gran número de herramientas de procesamiento de la información pueden tener acceso a los datos almacenados en la base de datos.
- Compatibilidad generalizada: existe una gran variedad de herramientas de bases de datos y configuraciones disponibles. [www.014]

Desventajas de utilizar una base de datos para mantener el estado:

- Complejidad: si se utiliza una base de datos para la administración de estado, las configuraciones del hardware y el software serán más complejas.
- Consideraciones sobre el rendimiento: una construcción deficiente del modelo de datos relacionales puede generar problemas de escalabilidad. También, el uso de demasiadas consultas a la base de datos puede afectar negativamente al rendimiento del servidor. [www.014]

1.8.4. Uso de Caché en ASP.NET

El almacenamiento en caché es una técnica utilizada en informática para aumentar el rendimiento.

Consiste en mantener datos de acceso frecuente o costoso en una memoria más rápida. En el contexto de una aplicación Web, el almacenamiento en caché se utiliza para retener páginas o datos durante las solicitudes HTTP y re utilizarlos sin tener que crearlos de nuevo.
ASP.NET dispone de tres clases de caché que se pueden utilizar en las aplicaciones Web. Ver la Tabla 1.5

Caché de resultados	Almacena en caché la respuesta dinámica generada por una solicitud.
Caché de fragmentos	Almacena en caché partes de una respuesta generada por una solicitud.
Caché de datos	Almacena en caché objetos arbitrarios mediante programación. Para ello, ASP.NET proporciona un motor de caché muy completo que permite a los programadores retener fácilmente datos entre solicitudes.

Tabla 1.5 Clases de Caché

1.8.4.1. Caché de Resultados

La caché de resultados es útil cuando el contenido de toda una página se puede almacenar en memoria caché.
En un sitio con muchos accesos, mantener en caché las páginas con accesos más frecuentes durante incluso un minuto cada vez puede producir incrementos significativos del rendimiento. Mientras una página permanece en la caché de resultados, las solicitudes subsiguientes de esa página se atienden desde la página de resultados, sin ejecutar el código que la creó.

1.8.4.2. Caché de Fragmentos

Se puede utilizar la caché de fragmentos para mantener en caché partes del resultado de una página.

A veces, no resulta práctico mantener en caché una página entera (quizá es mejor crear o personalizar partes de la página para cada solicitud). En este caso, suele ser preferible identificar aquellos objetos o datos cuya construcción sea más costosa como candidatos al almacenamiento en caché. Tras identificar

a esos elementos, se pueden crear una sola vez y almacenarse en caché durante algún tiempo.

1.8.4.3. Caché de Datos

Para algunos elementos, los datos podrían actualizarse a intervalos regulares o ser válidos durante un cierto período de tiempo. En ese caso, los elementos en caché pueden estar sometidos a unas reglas de caducidad que determinen el tiempo que pueden estar almacenados en la caché.
El código que obtiene acceso al elemento en caché simplemente comprueba la ausencia del elemento y vuelve a crearlo si es necesario.

La caché de ASP.NET admite dependencias de clave y de archivos, lo que permite a los programadores hacer que un elemento en caché dependa de un archivo externo o de otro elemento almacenado en caché. Esta técnica se puede utilizar para invalidar elementos cuando su origen de datos subyacente cambia. [www.015]

1.8.5. Configuraciones para Aplicaciones ASP.NET

Una de las ventajas que presta un Servidor de aplicaciones Web es la existencia de un sistema de configuración rico y flexible que permita a los programadores asociar valores de configuración con una aplicación instalable, sin la necesidad de incluir los valores dentro del código. Mediante esto, los administradores de una aplicación tengan la facilidad de personalizar estos valores después de la distribución.
El sistema de configuración de ASP.NET está diseñado para satisfacer las necesidades de ambos colectivos, proporcionando una infraestructura de configuración jerárquica que permite definir datos de configuración extensibles y utilizarlos a lo largo de una aplicación, un sitio o un equipo.

Características

ASP.NET, nos brinda la oportunidad de realizar diversas configuraciones para nuestras aplicaciones de una manera más sencilla y eficaz:

- ASP.NET permite almacenar valores de configuración junto con contenido estático, páginas dinámicas y objetos empresariales dentro de una única jerarquía de directorios de la aplicación. Anteriormente los

usuarios o administradores necesitaban copiar un único árbol de directorios para instalar una aplicación ASP.NET Framework en un equipo, en la actualidad a través de un instalador propio para la aplicación, esta tarea se realiza automáticamente.

- Los datos de configuración se almacenan en archivos de texto sin formato donde los usuarios pueden leer y escribir perfectamente. Los administradores y los programadores pueden utilizar cualquier editor de textos estándar, analizador XML o lenguaje de secuencias de comandos para interpretar y actualizar los valores de configuración.
- ASP.NET proporciona una infraestructura de configuración extensible que permite a los administradores de las aplicaciones, almacenar sus propios valores de configuración, definir el formato de persistencia de esos valores, participar de forma inteligente en su procesamiento y controlar el modelo de objetos resultante a través del cual esos valores se exponen en última instancia.
- El sistema detecta automáticamente los cambios en los archivos de configuración de ASP.NET y los aplica sin intervención del usuario sin verse en la necesidad de reiniciar el Servidor Web o el equipo para que surtan efecto.

1.8.6. Formato del archivo de configuración web.config

Los archivos de configuración de ASP.NET son archivos de texto basados en XML (cada uno con el nombre web.config).

Un archivo web.config aplica valores de configuración al directorio en el que se encuentra ubicado y a todos sus subdirectorios virtuales. Los valores de los subdirectorios pueden reemplazar o modificar opcionalmente los valores especificados en sus directorios principales.

Las aplicaciones web de ASP.NET, al trabajar con un Framework, pasan a trabajar directamente con el archivo machine.config, ubicado en el directorio raíz:\WINDOWS\Microsoft.NET\Framework\version\CONFIG\machine.config

El archivo machine.config, nos proporciona valores de configuración predeterminados para todo el equipo.

ASP.NET configura IIS para impedir el acceso directo desde un explorador a los archivos web.config, y así garantizar que sus valores no se hagan públicos (los intentos de acceso hacen que ASP.NET devuelva el código 403: Acceso prohibido).

Durante la ejecución, ASP.NET utiliza estos archivos de configuración web.config para calcular jerárquicamente una colección exclusiva de valores para cada solicitud de destino URL entrante (estos valores se calculan sólo una vez y después se guardan en caché a lo largo de las solicitudes subsiguientes. ASP.NET vigila automáticamente si se producen cambios e invalida la caché si cualquiera de los archivos de configuración cambia).

1.8.7. Secciones y Controladores de Secciones de Configuración

Un archivo web.config es un archivo de texto XML que puede contener elementos de documentos XML estándar, incluidos comentarios, texto, etiquetas bien formadas, etc.
El archivo puede ser ANSI, UTF-8 o Unicode; el sistema detecta automáticamente la codificación.
El elemento raíz de un archivo web.config es siempre una etiqueta <configuration>.
Los valores de ASP.NET y del usuario final se encapsulan entonces dentro de la etiqueta, como se indica a continuación:

```
<configuration>
   <!- Configuraciones dadas a la Aplicación -->
</configuration>
```

1.8.8. Elementos de la Etiqueta <configuration>

1.8.8.1. Controladores de Secciones de Configuración

Los controladores de secciones de configuración se declaran dentro de un archivo web.config mediante directivas de etiquetas de sección anidadas dentro de una etiqueta <configSections>. Las etiquetas de sección pueden calificarse con más detalle mediante etiquetas de grupo que permiten organizarlas en grupos. [www.016]

1.8.8.2. Grupos de Secciones de Configuración

La configuración de ASP.NET permite el agrupamiento jerárquico de secciones para una mejor organización.
Cada etiqueta de sección identifica un nombre de etiqueta que denota una sección específica de datos de configuración.

Los elementos sectionGroup representan el espacio de nombres al que se aplican los valores de configuración.

Una etiqueta <sectionGroup> puede aparecer dentro de una etiqueta <configSections> o dentro de otras etiquetas <sectionGroup>.

Por ejemplo, todos los controladores de sección de configuración de ASP.NET se agrupan en el grupo de sección system.web, como se muestra en el ejemplo

Ejemplo:

```
<sectionGroup name="system.web"
<type="System.Web.Configuration.SystemWebSectionGroup, System.Web,
Version=2.0.0, Culture=neutral, PublicKeyToken= f11d50a3a">
    <!-- <section /> elements. -->
</sectionGroup>
```

1.8.8.3. Valores de la Sección de Configuración

El área de valores de la sección de configuración va a continuación del área de declaración de controladores de sección de configuración y contiene los valores de configuración propiamente dichos.

De manera predeterminada, bien internamente o en uno de los archivos de configuración raíz, hay un elemento de sección de configuración especificado para cada elemento section y sectionGroup en el área configSections. Estos valores predeterminados se pueden ver en el archivo systemroot\Microsoft.NET\Framework\versionNumber\CONFIG\Machine.config. comments.

Un elemento de sección de configuración también podría contener elementos secundarios controlados por el mismo controlador de sección que el elemento primario. Por ejemplo, el siguiente elemento pages contiene un elemento namespaces que carece de un controlador de sección porque lo controla el controlador de sección pages. [www.016]

Ejemplo:

```
<pages
 buffer="true"
 enableSessionState="true"
 asyncTimeout="45"
<!-- Other attributes. --> >    <namespaces>
   <add namespace="System" />
   <add namespace="System.Collections" />
 </namespaces> </pages>
```

1.8.8.4. Uso de Location y Path

De forma predeterminada, todos los valores de configuración definidos dentro de la etiqueta <configuration> de nivel superior se aplican a la ubicación del directorio actual que contiene el archivo web.config y a todos sus subdirectorios.

Opcionalmente, se pueden aplicar valores de configuración a subdirectorios específicos por debajo del archivo de configuración actual mediante la etiqueta <location> y un atributo restrictivo path apropiado.

Si el archivo de configuración es el archivo principal machine.config, se pueden aplicar valores a aplicaciones o directorios virtuales específicos. Si el archivo de configuración es un archivo web.config, se pueden aplicar valores a un archivo, subdirectorio, directorio virtual o aplicación específicos.

Ejemplo:

```
<configuration>
  <location path="EnglishPages\OneJapanesePage.aspx">
  <system.web>
     <globalization
       requestEncoding="Shift-JIS"
       responseEncoding="Shift-JIS"
     /> </system.web> </location> </configuration>
```

1.8.8.5. Bloqueo de Valores de Configuración

Se pueden establecer medidas de seguridad para que los valores no sean reemplazados por otro archivo de configuración situado más abajo en la jerarquía de configuraciones. Para bloquear un grupo de valores, se puede especificar un atributo allowOverride con el valor false, en la etiqueta <location> que los engloba. El siguiente código bloquea valores de suplantación de identidad para dos aplicaciones diferentes. [www.003]

```
<configuration>
  <location path="app1" allowOverride="false"> <system.web>
        <identity impersonate="false" userName="app1"
        password="app1pw" /> </system.web> </location>
  <location path="app2" allowOverride="false"> <system.web>
        <identity impersonate="false" userName="app2"
        password="app2pw" /> </system.web> </location>
</configuration>
```

Observe que si un usuario intenta reemplazar estos valores en otro archivo de configuración, el sistema de configuración indicará un error:

```
<configuration>
  <system.web>
     <identity userName="developer" password="loginpw" />
  </system.web>
```

1.8.8.6. Sección de Configuración ASP.NET Estándar

ASP.NET se suministra con una serie de controladores estándar de secciones de configuración para procesar los valores de configuración de los archivos web.config. La siguiente tabla proporciona una breve descripción de las secciones junto con referencias para obtener más información.

Nombre de Sección	Descripción
<httpModules>	Está encargada de configurar módulos HTTP dentro de una aplicación. Los módulos HTTP participan en el procesamiento de todas las solicitudes en una aplicación. Entre los usos más habituales, se incluyen la seguridad y el inicio de sesión.
<httpHandlers>	Se encarga de asignar direcciones URL de entrada a clases IHttpHandler. Es responsable de asignar direcciones URL de entrada a clases IHttpHandlerFactory. Los datos representados en secciones <httpHandlers> son heredados jerárquicamente por subdirectorios.
<sessionState>	Se encarga de configurar el módulo HTTP de estado de sesión.
<globalization>	Se encarga de configurar los valores de globalización de una aplicación.
<compilation>	Es responsable de todos los valores de compilación utilizados por ASP.NET.
<trace>	Se encarga de configurar el servicio de

	seguimiento de ASP.NET.
<processModel>	Se encarga de configurar los valores del modelo de proceso de ASP.NET en sistemas de servidores Web de IIS.
<browserCaps>	Es responsable del control de los valores del componente de funcionalidad de explorador.

Tabla 1.6 Sección de Configuración ASP.NET

1.8.9. Implementación de Aplicaciones ASP.NET

1.8.9.1. Diseño del Sistema de Archivos de las Aplicaciones ASP.NET

Se puede utilizar ASP.NET para albergar varias aplicaciones Web, cada una identificada mediante un prefijo URL único dentro de un sitio Web.
Por ejemplo:
Un solo servidor web de Microsoft Internet Information Services (IIS) con dos direcciones IP una con alias "www.hotmail.com" y otra "mintranet" y tres sitios lógicos http://mintranet, http://www.hotmail.com, http://www.hotmail.com:81

Cada aplicación ASP.NET Framework con un nombre URL, puede estar en cualquier parte del disco, los directorios de aplicaciones no necesitan estar ubicados centralmente en una parte contigua del sistema de archivos; pueden estar esparcidos por todo un disco.
Por ejemplo:
La aplicación ASP.NET con dirección URL http://www.mintranet.com puede estar en la ruta de acceso física D:\larutaquequiero.

1.8.9.2. Resolución de Referencias de Clases en Ensamblados

Los ensamblados constituyen la unidad de implementación de clases en Common Language Runtime. Al momento de la compilación de un sistema creado en Visual Studio .NET, se producen un nuevo. Aunque es posible hacer que un ensamblado ocupe varios archivos ejecutables portables (PE) (varias DLL de módulos), Visual Studio .NET compilará, de forma predeterminada, todo

el código del ensamblado en una única DLL (1 proyecto de Visual Studio .NET = 1 ensamblado de .NET Framework = 1 DLL física).

Para utilizar un ensamblado en un equipo, se debe implementar en una caché de ensamblado. La caché de ensamblado puede ser global para un equipo o local para una aplicación determinada. En la caché de ensamblado global del sistema, sólo debería colocarse el código que se va a compartir entre varias aplicaciones.

El código específico para una aplicación particular, como la lógica de la mayoría de las aplicaciones web, se debería implementar en la caché de ensamblado local de la aplicación.

Una de las ventajas de distribuir un ensamblado en la caché de ensamblado local de una aplicación es que sólo el código interno de esa aplicación puede tener acceso a él.

También facilita la creación conjunta de versiones de la misma aplicación, ya que las clases son privadas para cada instancia de versión de la aplicación.

Un ensamblado se puede distribuir en la caché de ensamblado local de una aplicación simplemente copiando (o utilizando XCOPY o FTP) los archivos apropiados en un directorio marcado como "ubicación para la caché de ensamblado" de esa determinada aplicación. No es necesario ejecutar ninguna herramienta de registro adicional. Una vez copiados los archivos apropiados y tampoco es necesario reiniciar el equipo. Esto elimina algunas de las dificultades actualmente asociadas a la distribución de componentes COM dentro de aplicaciones ASP (No hace mucho, un administrador tenía que iniciar sesión localmente en el servidor web y ejecutar Regsvr32.exe).

De forma predeterminada, una aplicación ASP.NET Framework se configura automáticamente para utilizar el subdirectorio \\bin, ubicado inmediatamente bajo la raíz de la aplicación, como su caché de ensamblado local.

El directorio \\bin también está configurado para denegar cualquier acceso de un explorador Web, de modo que un cliente remoto no pueda descargar el código y apropiarse del mismo.

1.8.10. Inicio de Aplicaciones ASP.NET y Resolución de Clases

Las aplicaciones ASP.NET Framework se construyen lentamente la primera vez que un cliente solicita un recurso URL de ellas.

Cada aplicación ASP.NET Framework se lanza dentro de un dominio de aplicación exclusivo (AppDomain), que consiste en una nueva construcción de

Common Language Runtime que permite a los hosts de procesos ofrecer código extensivo, seguridad y aislamiento de configuraciones durante la ejecución.
ASP.NET crea manualmente un dominio de aplicaciones cuando se inicia una nueva aplicación. Como parte de este proceso, ASP.NET proporciona valores de configuración para que Common Language Runtime los utilice.
Entre los valores de configuración, están:

- Las rutas de acceso a directorios que componen la caché de ensamblado local. (Nota: es la arquitectura de aislamiento de dominios para aplicaciones de .NET Framework la que permite a cada aplicación mantener su propia caché de ensamblado local).
- Las restricciones de seguridad de la aplicación (donde puede tener acceso la aplicación en el sistema).

Como ASP.NET no tiene conocimiento, en tiempo de compilación, de las aplicaciones que se escriben por encima de él, no puede utilizar referencias estáticas para resolver y hacer referencia al código de las aplicaciones.
ASP.NET, al momento de resolver y hacer referencia al código de las aplicaciones en tiempo de compilación, debe utilizar un enfoque de resolución dinámico de clases y ensamblados para realizar la transición del módulo de ejecución de ASP.NET al código de aplicación.
Los archivos de activación de página y configuración de ASP.NET permiten hacer referencia dinámicamente a una clase de .NET Framework compilada para un objetivo especificando una combinación de nombre de clase y ensamblado.
El formato de cadena para esta unión sigue el modelo classname (identificación del código), assemblyname (identificación del ensamblado). De allí, Common Language Runtime puede utilizar esta referencia simple de cadena para resolver y cargar la clase apropiada.

1.8.10.1. Sustitución de Código

Los ensamblados de .NET Framework normalmente se compilan y distribuyen en un formato PE (ejecutables portables) basado en DLL de Windows. Cuando el cargador de Common Language Runtime resuelve una clase implementada con este tipo de ensamblado, llama a la rutina LoadLibrary de Windows en el archivo (la cual bloquea su acceso en disco) y, a continuación, asigna los datos de código en memoria apropiados para la ejecución. Al estar cargado, el

archivo DLL permanece bloqueado en disco hasta que el dominio de aplicación que hace referencia a él se destruye o se recicle manualmente.

Aunque ASP.NET no puede impedir que Common Language Runtime bloquee una DLL de ensamblado en disco, sí puede garantizar que el módulo de ejecución no cargue nunca realmente las DLL físicas en la caché de ensamblado privada de una aplicación Web. En vez de ello, se realizan copias en servidores de copia de seguridad de las DLL de ensamblados inmediatamente antes de su uso. El módulo de ejecución bloquea y carga entonces estos ensamblados del servidor de copia de seguridad, con lo que no expone los archivos originales.

Como los archivos de ensamblado originales siempre permanecen desbloqueados, existe la libertad de eliminarlos, reemplazarlos o cambiar su nombre sin activar el servidor web o tener que usar una utilidad de registro. ASP.NET mantiene una lista activa de todos los ensamblados cargados dentro del dominio de una aplicación particular y utiliza código de supervisión de cambio de archivos para detectar cualquier actualización en los archivos originales.

1.8.11. Modelo de Procesos

En el nivel de procesos se comparten demasiados recursos y es muy fácil que un error interrumpa todo el proceso de servidor.

Para solucionar este problema, ASP.NET proporciona un modelo de ejecución fuera de proceso, que protege al proceso de servidor del código de usuario. También permite aplicar técnicas heurísticas a la duración del proceso para mejorar la disponibilidad de las aplicaciones web.

El uso de comunicaciones asincrónicas entre procesos permite proporcionar el mejor equilibrio entre rendimiento, escalabilidad y fiabilidad. [www.017]

1.8.11.1. Configuración del Modelo de Procesos

La configuración del modelo de procesos se expone en el archivo de configuración raíz del equipo, Machine.config.

La sección de configuración se denomina <processModel>. El modelo de procesos está habilitado de forma predeterminada (enable="true").

Los ítems pertenecientes a la configuración del modelo de procesos se muestran en la Tabla 1.7

```
<processModel
    enable="true"
    timeout="infinite"
    idleTimeout="infinite"
    shutdownTimeout="0:00:05"
    requestLimit="infinite"
    requestQueueLimit="5000"
    memoryLimit="80"
    webGarden="false"
    cpuMask="0xffffffff"
    userName=""
    password=""
    logLevel="errors"
    clientConnectedCheck="0:00:05"
/>
```

Tabla 1.7 Sección de Configuración <processModel>

La mayoría de los valores de configuración controlan el inicio de un nuevo proceso de trabajo para atender solicitudes (reemplazando discretamente un proceso de trabajo antiguo).
El modelo de procesos admite dos tipos de reciclaje: reactivo y proactivo.

1.8.11.2. Reciclaje Reactivo de Procesos

El reciclaje reactivo de procesos se produce cuando un proceso funciona incorrectamente o no puede atender solicitudes. El proceso suele mostrar síntomas que se pueden detectar, como bloqueos, infracciones de acceso, fugas de memoria, etc., para desencadenar su reciclaje. Es posible controlar las condiciones que desencadena el reinicio de un proceso mediante los valores de configuración descritos en la tabla 1.8. [www.017]

Valor	Descripción
requestQueueLimit	Controla las condiciones de bloqueo. El valor DWORD se establece en el número máximo permitido de solicitudes en cola que, una vez superado, hace que se considere que el proceso de trabajo funciona incorrectamente. Cuando se supera este número, se inicia un proceso nuevo y se reasignan las solicitudes. El valor predeterminado es de 5.000 solicitudes.
memoryLimit	Controla las condiciones de pérdida de memoria. El valor DWORD se establece en el porcentaje de memoria física que puede consumir el proceso de trabajo antes de que se considere que funciona incorrectamente. Cuando se supera este porcentaje, se inicia un proceso nuevo y se reasignan las solicitudes. El valor predeterminado es el 80%.
shutdownTimeout	Especifica el intervalo de tiempo del que dispone el proceso de trabajo para cerrarse discretamente (valor de tipo cadena, con el formato hr:min:seg). Cuando se agote el tiempo de espera, la API de servicios de Internet de ASP.NET cerrará el proceso de trabajo. El valor predeterminado es "00:00:05".

Tabla 1.8 Condiciones que Desencadena el Reinicio de un Proceso

1.8.11.3. Reciclaje Proactivo de Procesos

El reciclaje proactivo de procesos reinicia periódicamente el proceso de trabajo, aunque éste funcione correctamente.
Esto puede ser una forma útil de evitar la denegación de servicio a causa de condiciones que no puede detectar el modelo de procesos.

Es posible reiniciar un proceso después de un número determinado de solicitudes o cuando haya transcurrido un tiempo de espera. [www.017]
Para ver los atributos de configuración observe la Tabla 1.9

Valor	Descripción
timeout	Valor de tipo cadena, con el formato hr:min:seg, que establece el límite de tiempo que, una vez transcurrido, hace que se inicie un nuevo proceso de trabajo para reemplazar al actual. El valor predeterminado es infinite, una palabra clave que indica que no se debe reiniciar el proceso.
idleTimeout	Valor de tipo cadena, con el formato hr:min:seg, que configura el tiempo de inactividad que, una vez transcurrido, hace que se cierre automáticamente el proceso de trabajo. El valor predeterminado es infinite, una palabra clave que indica que no se debe reiniciar el proceso.
requestLimit	El valor DWORD se establece en el número de solicitudes que pueden producirse antes de que se inicie un nuevo proceso de trabajo para reemplazar al actual. El valor predeterminado es infinite.

Tabla 1.9 Opciones para Configurar el Reciclaje Proactivo de Procesos

1.8.11.4. Registro de Eventos del Modelo de Procesos

El modelo de procesos puede escribir eventos en el registro de eventos de Windows cuando se realice el cambio de proceso. Esto se controla mediante el atributo logLevel en la sección de configuración <processModel>. [www.017]

Valor	Descripción
logLevel	Controla el registro de eventos de cambio de proceso en el registro de eventos. El valor puede ser: • Todos: se registran todos los eventos de cambio de proceso. • Ninguno: no se registra ningún evento. • Errores: sólo se registran los eventos inesperados.

Tabla 1.10 Opciones de Configuración del atributo logLevel

Cuando se produce un evento de cambio, si el registro está habilitado para el evento, se escriben los siguientes eventos en el registro de eventos de la aplicación.

Razón del cierre	Tipo de registro de eventos	Descripción
Inesperado	Error	El proceso de trabajo de ASP.NET se cerró de forma inesperada.
requestQueueLimit	Error	Se reinició el proceso de trabajo de ASP.NET porque se superó el límite de la cola de solicitudes.
RequestLimit	Información	Se reinició el proceso de trabajo de ASP.NET porque se superó el límite de solicitudes.
Timeout	Información	Se reinició el proceso de trabajo de ASP.NET porque se superó el intervalo de tiempo de espera.
IdleTimeout	Información	Se cerró el proceso de trabajo de ASP.NET porque se superó el intervalo de tiempo de espera de inactividad.
MemoryLimitExceeded	Error	Se reinició el proceso de trabajo de ASP.NET porque se superó el límite de memoria del proceso.

Tabla 1.11 Atributos del Registro de Eventos

1.8.11.5. Unidades Web

El modelo de procesos permite habilitar la escalabilidad en equipos con varios procesadores mediante la distribución del trabajo entre varios procesos, uno

por cada CPU, con la afinidad de procesador establecida en la CPU correspondiente. Esto elimina la contención de bloqueos entre procesadores, por lo que es ideal para grandes sistemas de multiprocesamiento simétrico (SMP).

Los valores de configuración necesarios para habilitar unidades Web se muestran en la siguiente tabla.

Es necesario reiniciar el servidor para que esta configuración surta efecto.

Valor	Descripción
webGarden	Controla la afinidad CPU. True indica que es necesario asignar a los procesos la afinidad correspondiente a la CPU. El valor predeterminado es False.
cpuMask	Controla el número de procesos y el funcionamiento de la unidad Web. Se inicia un proceso para cada CPU en la que el bit correspondiente de la máscara está establecido en 1. Cuando se establece el valor de UseCPUAffinity en 0, el valor de cpuMask sólo controla el número de procesos de trabajo (número de bits establecidos en 1). El número máximo permitido de procesos de trabajo es el número de CPU. De forma predeterminada, todas las CPU están habilitadas y se iniciarán tantos procesos de trabajo como CPU haya en el equipo. El valor predeterminado es 0xffffffff.

Tabla 1.12 Atributos para la configuración de Unidades Web

El uso de unidades Web produce algunos efectos secundarios que hay que tener en cuenta:

- Si la aplicación utiliza el estado de sesión, debe elegir un proveedor fuera de proceso (servicio de Windows NT o SQL).
- Para cada proceso se miden tanto el estado de la aplicación como las estadísticas de la aplicación y no para cada equipo.
- El almacenamiento en caché se realiza para cada proceso, no para cada equipo. [www.017]

1.8.12. Control de Errores

Cuando se produce un error en una página, ASP.NET envía información sobre el error al cliente.

Los errores se dividen en:

- Errores de configuración: se producen cuando la sintaxis o la estructura de un archivo Web.config de la jerarquía de configuración es incorrecta.
- Errores del analizador: se producen cuando la sintaxis ASP.NET de una página no está bien formada.
- Errores de compilación: se producen cuando las instrucciones del lenguaje de destino de una página son incorrectas.
- Errores de tiempo de ejecución: se producen durante la ejecución de una página, incluso aunque los errores no se detectaran en tiempo de compilación.

De forma predeterminada, la información mostrada para un error de tiempo de ejecución consiste en la pila de llamadas (cadenas de llamadas a procedimientos que nos llevaron a la excepción).

Si el modo de depuración se encuentra habilitado, ASP.NET muestra el número de línea del código fuente donde se originó el error en tiempo de ejecución.

El modo de depuración es una herramienta muy valiosa para depurar la aplicación.

Es posible habilitar el modo de depuración a nivel de página, mediante la siguiente directiva:

<%@ Page Debug="true" %>

También se puede habilitar el modo de depuración en el ámbito de la aplicación, mediante el archivo Web.config del directorio raíz de la aplicación, como se muestra a continuación:

```
<configuration>
    <system.web>
<customErrors defaultRedirect= "genericerror.htm"
mode="remoteonly" />
    </system.web>
</configuration>
```

El modo de depuración implica una penalización de rendimiento.
En la tabla 1.13 se muestra los atributos de configuración y los valores para la etiqueta <customerrors>.

Atributo	Descripción
Mode	Indica si los errores personalizados están habilitados, deshabilitados o sólo se muestran a equipos remotos. Valores: On, Off, RemoteOnly (predeterminado).
DefaultRedirect	Indica la URL predeterminada a la que se debe desviar a un explorador Web si se produce un error. Este atributo es opcional.

Tabla 1.13 Atributos de Configuración y Valores para <customerrors>

El atributo Mode determina si los errores se van a mostrar a clientes locales, remotos o a ambos. Los efectos de cada opción se describen en la tabla 1.14

Modo	Solicitud de host local	Solicitud de host remoto
On	Página de errores personalizada.	Página de errores personalizada.
Off	Página de errores de ASP.NET.	Página de errores de ASP.NET.
RemoteOnly	Página de errores de ASP.NET.	Página de errores personalizada.

Tabla 1.14 Efectos de opción del atributo Mode

Además de desviar a una página común cuando se produce cualquier error, también se pueden asignar páginas de error específicas a códigos de estado de error específicos.
La sección de configuración <customerrors> admite una etiqueta <error> interna que asocia códigos de estado HTTP con páginas de error personalizadas.

```
<configuration>  <system.web>
   <customErrors mode="RemoteOnly" defaultRedirect="/multierror.htm">
     <error statusCode="500" redirect="/error/soporte.htm"/>
     <error statusCode="404" redirect="/error/noencontrado.aspx"/>
     <error statusCode="403" redirect="/error/sinacceso.aspx"/>
```

1.8.13. Seguridad de Aplicaciones ASP.NET

Es esencial en las Aplicaciones Web poder identificar usuarios y controlar el acceso a los recursos. Para esto, el usuario debe presentar sus credenciales, como el par nombre de usuario y contraseña, para ser autenticado.

Cuando se encuentre disponible una identidad autenticada, deberá determinarse si esa identidad puede tener acceso a un recurso específico. Este proceso se conoce como autorización.

ASP.NET e IIS colaboran para proporcionar servicios de autenticación y autorización a las aplicaciones.

Una característica importante de los objetos COM es la capacidad de controlar la identidad con la que se ejecuta el código de objeto COM.

Se conoce como representación al hecho de que un objeto COM ejecute código con la identidad de la entidad solicitante. Opcionalmente, las aplicaciones ASP.NET Framework pueden representar solicitudes.

Es posible que algunas aplicaciones también personalicen dinámicamente el contenido en función de la identidad del solicitante o de un conjunto de funciones al que pertenece la identidad del solicitante. Las aplicaciones ASP.NET Framework pueden comprobar dinámicamente si la entidad solicitante actual participa en una función concreta.

Por ejemplo, es posible que una aplicación intente comprobar si el usuario actual pertenece o ejecuta un rol de Administrador o tiene limitaciones de algún tipo.

1.8.13.1. Arquitectura de seguridad de ASP.NET

En la siguiente ilustración se muestran las relaciones entre los sistemas de seguridad de ASP.NET.

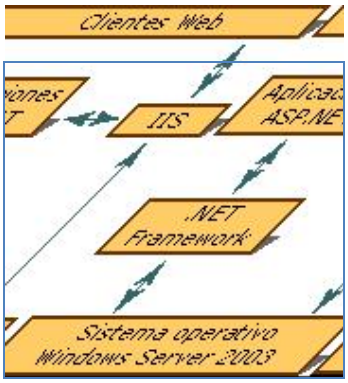


FIGURA 1.1: Arquitectura de la Seguridad en ASP.NET

Como se muestra en la ilustración, todos los clientes Web se comunican con las aplicaciones ASP.NET a través de Servicios de Microsoft Internet Information Server (IIS).

IIS autentica la solicitud si fuera necesario y, a continuación, busca el recurso solicitado (como una aplicación ASP.NET).

Si el cliente está autorizado, el recurso estará disponible.

Cuando se está ejecutando una aplicación ASP.NET, puede utilizar las características de seguridad de ASP.NET integradas. Además, una aplicación ASP.NET puede utilizar las características de seguridad de .NET Framework. [www.018]

1.8.13.2. Autenticación y Autorización

ASP.NET funciona, junto con IIS, para permitir la autenticación de tipo básica, implícita y Windows.

ASP.NET es compatible con el servicio de autenticación de pasaporte de Microsoft, el cual proporciona servicios simples de firma y servicios de perfiles de usuario.

ASP.NET proporciona un robusto servicio para aplicaciones que necesitan usar autenticación basada en formularios.

La autenticación basada en formularios usa "cookies" para autenticar a los usuarios, y permite a la aplicación realizar su propia verificación de credenciales.

Los servicios de autenticación de ASP.NET dependen de los servicios de autenticación suministrados por IIS.

Para poder utilizar una autenticación básica en una aplicación IIS, se debe configurar el uso de la autenticación básica para la aplicación mediante el Administrador de servicios Internet.

En ASP.NET existen dos formas de autorizar el acceso a un recurso dado:

- Autorización de archivos FileAuthorizationModule realiza la autorización de archivos. Realiza una comprobación de la lista de control de acceso (ACL) del archivo de controladores .aspx o .asmx para determinar si un usuario debe tener acceso al archivo. Los permisos de ACL se comprueban para la identidad de Windows (si se habilita la autenticación de Windows) del usuario o para la identidad de Windows del proceso de ASP.NET
- Autorización de URL UrlAuthorizationModule realiza la autorización de URL, que asigna usuarios y funciones a direcciones URL en aplicaciones ASP.NET.

Se puede utilizar para permitir o denegar de forma selectiva el acceso a las partes arbitrarias de una aplicación (normalmente los directorios) para usuarios concretos o funciones.

Con la autorización de URL, permite o deniega explícitamente el acceso a un directorio determinado por nombre de usuario o función. Para ello, se crea una sección authorization en el archivo de configuración para ese directorio. Para habilitar la autorización de URL, basta con especificar una lista de usuarios o funciones en los elementos permitir o denegar de la sección autorización de un archivo de configuración. Los permisos establecidos para un directorio también se aplican a sus subdirectorios. [www.011]

Ejemplo:

```
<authorization>
  <[allow|deny] users roles verbs />
</authorization>
```

Los elementos allow y deny conceden o revocan el acceso, respectivamente.

Cada elemento admite los atributos mostrados en la tabla 1.15 Atributos :

Atributo	Descripción
users	Identifica las identidades (cuentas de usuario) a las que se destina el elemento. Los usuarios anónimos se identifican mediante un signo de interrogación (?). Puede especificar todos los usuarios autenticados mediante un asterisco (*).
roles	Identifica una función (un objeto RolePrincipal) para la solicitud actual a la que se permite o deniega el acceso al recurso.

verbs	Define los verbos HTTP a los que se aplica la acción, como GET, HEAD y POST. El valor predeterminado es "*" que especifica todos los verbos.

Tabla 1.15 Atributos de allow y deny

Ejemplo:

Conceder acceso a la identidad Kim y a los miembros de la función Admins, y deniega el acceso a la identidad John y a todos los usuarios anónimos:

```
<authorization>
  <allow users="Kim"/>
  <allow roles="Admins"/>
  <deny users="John"/>
  <deny users="?"/>
</authorization>
```

Puede especificar varias entidades para los atributos users y roles utilizando una lista separada por comas.

Ejemplo:

```
<allow users="John, Kim, contoso\Jane"/>
```

Ejemplo con los dos tipos de autorización de acceso:

Una aplicación está configurada para permitir acceso anónimo mediante la cuenta IUSR_MYMACHINE. Cuando una solicitud de una página ASP.NET (como "/inicial.aspx") recibe autorización, se realiza una comprobación de los ACL de ese archivo (por ejemplo, "c:\\inetpub\\wwwroot\\ inicial.aspx") para ver si la cuenta IUSR_MYMACHINE tiene permiso para leer el archivo. Si lo tiene, entonces se autoriza el acceso. La autorización de archivo se realiza automáticamente.

Para autorización de URL, se comprueba el usuario anónimo con los datos de configuración obtenidos para la aplicación ASP.NET. Si se permite el acceso a la dirección URL solicitada, la solicitud se autoriza. En este caso, ASP.NET comprueba si el usuario anónimo dispone de acceso a / inicial.aspx (es decir, la comprobación se realiza sobre la propia URL, no sobre el archivo en el que se resuelve la URL en última instancia).

Ésta podría parecer una diferencia sutil, pero permite a las aplicaciones utilizar esquemas de autenticación como los basados en formularios o la autenticación

de Pasaporte, en los cuales los usuarios no se corresponden con un equipo o una cuenta de dominio.

La autorización de archivo se realiza siempre con la cuenta autenticada suministrada por IIS. Si se permite el acceso anónimo, esta cuenta es la cuenta anónima configurada. En cualquier otro caso, se utiliza una cuenta NT. Esto funciona exactamente de la misma forma que ASP.

Las listas ACL de archivos se configuran para un determinado archivo o directorio mediante la ficha Seguridad de la página de propiedades del Explorador.

La autorización de URL es configurada como parte de una aplicación ASP.NET Framework.

Para activar un servicio de autenticación de ASP.NET, es necesario configurar el elemento z en el archivo de configuración de la aplicación.

Este elemento puede tener uno de los valores mostrados en la tabla 1.16.

Valor	Descripción
Ninguno	No hay ningún servicio de autenticación de ASP.NET activo. Observe que los servicios de autenticación de IIS pueden estar aún presentes.
Windows	Los servicios de autenticación de ASP.NET asocian un WindowsPrincipal (System.Security.Principal.WindowsPrincipal) a la solicitud actual para permitir la autorización de usuarios o grupos de NT.
Formularios	Los servicios de autenticación de ASP.NET administran las "cookies" y desvían a los usuarios no autenticados a una página de inicio de sesión. Este método se suele utilizar en conjunción con la opción IIS que permite el acceso anónimo a una aplicación.
Passport	Los servicios de autenticación de ASP.NET proporcionan un envoltorio apropiado para los servicios suministrados por el kit de desarrollo de software para pasaporte (Passport SDK), el cual debe instalarse en el equipo.

Tabla 1.16 Valores de Configuración de la etiqueta <authentication>

Por ejemplo, el siguiente archivo de configuración permite la autenticación basada en formularios (cookie) para una aplicación:

```
<configuration>
  <system.web>
    <authentication mode="Forms"/>
  </system.web>
</configuration>
```

1.8.13.2.1. Autenticación Basada en Windows

La Autenticación de Windows trabaja con la identidad de usuario proporcionada por Servicios de Microsoft Internet Information Server (IIS) como el usuario autenticado en una aplicación ASP.NET.

La autenticación de Windows se implementa en ASP.NET utilizando el módulo WindowsAuthenticationModule. El módulo construye un objeto WindowsIdentity basándose en las credenciales suministradas por IIS y establece la identidad como el valor actual de la propiedad User para la aplicación. [www.019]

La autenticación de Windows es el mecanismo de autenticación predeterminado para las aplicaciones ASP.NET y se identifica como el modo de autenticación para una aplicación utilizando el elemento de configuración authentication.

Ejemplo:

```
<system.web>
  <authentication mode="Windows"/>
</system.web>
```

1.8.13.2.2. Suplantación de la Identidad de Windows

La identidad de Windows suministrada al sistema operativo se utiliza para la comprobación de permisos, como los permisos de archivos NTFS, o para la conexión a una base de datos utilizando seguridad integrada. De forma predeterminada, esta identidad de Windows es la identidad del proceso de ASP.NET.

En Windows 2003, ésta es la identidad del Grupo de aplicaciones IIS del que forma parte la aplicación ASP.NET. De forma predeterminada, ésta es la cuenta NETWORK SERVICE.

Para configurar la identidad de Windows de la aplicación ASP.NET como la identidad de Windows proporcionada por IIS hay que habilitar la suplantación. Es decir, indica a la aplicación ASP.NET que suplante la identidad suministrada por IIS para todas las tareas autenticadas por el sistema operativo Windows, incluyendo el acceso a archivos y a la red.

Para habilitar la suplantación para la aplicación Web, en el archivo Web.config de la aplicación establezca el atributo impersonate del elemento identity en true.

Ejemplo:

```
<system.web>
  <authentication mode="Windows"/>
  <identity impersonate="true"/>
</system.web>
```

1.8.13.2.3. Autenticación Basada en Formularios

La autenticación basada en formularios es un servicio de autenticación de ASP.NET que permite a las aplicaciones suministrar su propia interfaz de inicio de sesión y hacer su propia verificación de credenciales. ASP.NET permite autenticar usuarios y desviar a los usuarios no autenticados hacia la página de inicio de sesión, además de realizar todas las tareas de administración de "cookies".

Para que una aplicación pueda utilizar autenticación basada en formularios, se debe configurar <authentication> con la opción Forms y denegar el acceso a los usuarios anónimos.

Ejemplo:

```
<authentication mode="Forms">
  <forms name="SavingsPlan" loginUrl="/Logon.aspx">
    <credentials passwordFormat="SHA1">
      <user name="Kim"
          password="07B7F3EE06F278DB966BE960E7CBBD103DF30CA6"/>
      <user name="John"
          password="BA56E5E0366D003E98EA1C7F04ABF8FCB3753889"/>
    </credentials>
  </forms>
</authentication>
```

Los administradores usan autenticación basada en formularios para configurar el nombre de la "cookie" a usar, el tipo de protección, la dirección URL para la página de inicio de sesión, el tiempo de validez de la "cookie" y la ruta de acceso que se debe utilizar para la "cookie" suministrada. [www.019]
Los Atributos para el elemento <Forms>, el cual es un subelemento del elemento <authentication> se muestran en la tabla 1.17:

Atributo	Descripción
loginUrl	URL de inicio de sesión a la que se desvían los usuarios no autenticados. Puede estar en el mismo equipo o en uno remoto. Si es un equipo remoto, ambos equipos deben utilizar el mismo valor para el atributo decryptionkey.
Name	Nombre de la "cookie" HTTP que se va a utilizar a efectos de autenticación. Observe que si varias aplicaciones desean utilizar servicios de autenticación basados en formularios en un único equipo, cada uno debería configurar un valor de "cookie" única. Para evitar originar dependencias en direcciones URL, ASP.NET utiliza "/" como valor de la ruta de acceso al configurar "cookies" de autenticación, de modo que éstas se vuelvan a enviar a todas las aplicaciones del sitio.
timeout	Tiempo, en minutos enteros, tras el cual la "cookie" caduca. El valor predeterminado es 30. El atributo timeout indica la caducidad con un valor continuo de x minutos contados desde el momento en que se recibió la última solicitud. Para evitar efectos negativos relacionados con el rendimiento y advertencias de los exploradores Web que tienen activadas las advertencias de "cookies", la "cookie" se actualiza si ha transcurrido más de la mitad del tiempo.
Path	Ruta de acceso que se utiliza para la "cookie"

	suministrada. Para evitar problemas con la escritura de las rutas, se utiliza "/" como valor predeterminado, ya que los exploradores Web distinguen estrictamente entre mayúsculas y minúsculas cuando devuelven "cookies". Las aplicaciones en un entorno de un servidor compartido deberían utilizar esta directiva para mantener "cookies" privadas.
protection	Método utilizado para proteger los datos de las "cookies". Los valores posibles son los siguientes: • All: utiliza validación y cifrado de datos para proteger la "cookie". • All es el valor predeterminado (y sugerido). • None: se utiliza con sitios que utilizan "cookies" sólo para personalización y que tienen requisitos de seguridad menores. Tanto el cifrado como la validación, se pueden deshabilitar. Aunque se debería tener precaución al utilizar "cookies" de este modo, esta opción proporciona el mejor rendimiento entre los métodos de personalización mediante .NET Framework. • Encryption: permite cifrar la "cookie" mediante TripleDES o DES, pero no se realiza validación de datos. Este tipo de "cookie" puede sufrir ataques en lo que se refiere a la selección de texto sin formato. • Validation: el contenido de la "cookie" no está cifrado, pero se valida para comprobar que no ha sido modificado durante la transmisión. Para crear la "cookie", la clave de validación se concatena en un búfer con los datos de la "cookie" y, a continuación, se genera un MAC

	que se agrega a la "cookie" saliente.

Tabla 1.17 Atributos para el elemento <Forms>

Tras haber configurado la aplicación, se debe proporcionar una página de inicio de sesión.

La "cookie" utilizada por la autenticación basada en formularios consiste en una versión lineal de la clase System.Web.Security.FormsAuthenticationTicket.

La información incluye el nombre de usuario (pero no la contraseña), la versión de autenticación basada en formularios utilizada, la fecha en la que se emitió la "cookie" y un campo para datos opcionales específicos de la aplicación. [www.019]

El código de la aplicación puede revocar o quitar las "cookies" de autenticación mediante el método FormsAuthentication.SignOut. Este método permite quitar la "cookie" de autenticación independientemente de si es temporal o permanente.

Se pueden suministrar servicios de autenticación basados en formularios con una lista de credenciales válidas que utilizan la configuración.

Ejemplo:

```
<authentication mode="Forms">
  <forms name="SavingsPlan" loginUrl="/Logon.aspx">
    <credentials passwordFormat="SHA1">
      <user name="Kim"
          password="07B7F3EE06F278DB966BE960E7CBBD103DF30CA6"/>
      <user name="John"
           password="BA56E5E0366D003E98EA1C7F04ABF8FCB3753889"/>
    </credentials>
  </forms>
</authentication>
```

La aplicación puede entonces realizar una llamada a FormsAuthentication. Authenticate, con el nombre de usuario y la contraseña, y ASP.NET se encargará de verificar las credenciales. Las credenciales se pueden almacenar en texto no cifrado o como código "hash" de tipo SHA1 o MD5, según los siguientes valores del atributo passwordFormat:

Tipo Hash	Descripción

Clear	Las contraseñas se almacenan en texto no cifrado
SHA1	Las contraseñas se almacenan en compendios SHA1
MD5	Las contraseñas se almacenan en compendios MD5

Tabla 1.18 Valores del Atributo passwordFormat

1.8.13.2.4. Autorización de Usuarios y Funciones

ASP.NET controla el acceso de los clientes a los recursos URL.
Se puede configurar para el método HTTP que se utilice para realizar la solicitud (GET o POST) y, también, para permitir o denegar el acceso a grupos de usuarios o funciones.

Ejemplo :

```
<authorization>
        <allow users="jdoe@contoso.com" />
        <allow roles="Admins" />
        <deny users="*" />
</authorization>
```

Los elementos que se permiten en las directivas de autorización pueden ser allow o deny. Cada elemento allow o deny debe contener un atributo users o un atributo roles. Se pueden especificar varios usuarios o funciones en un único elemento mediante una lista de valores separados por comas.
Ejemplo:

```
<allow users="John,Mary" />
```

El método HTTP se puede indicar mediante el atributo Verb:

```
<allow VERB="POST" users=" Diego, Dyan " />
<deny VERB="POST" users="*" />
<allow VERB="GET" users="*" />
```

Este ejemplo permite a Dyan y Diego utilizar el método POST en los recursos protegidos, mientras que sólo permite utilizar el método GET al resto de los usuarios.
Existen dos nombres de usuario especiales:

- *: Todos los usuarios
- ?: Usuarios anónimos (no autenticados)

Ejemplo:

```
<authorization>
   <deny users="?" />
</authorization>
```

La autorización de direcciones URL se evalúa jerárquicamente y las reglas que se utilizan para determinar el acceso son las siguientes:

- Las reglas relevantes para la URL se obtienen de la jerarquía y con ellas se construye una lista combinada de reglas.
- Las reglas más recientes se colocan al principio de la lista.
- Las reglas se comprueban hasta encontrar una que se cumpla. Si la regla es admisible, el acceso se concede. En caso contrario, el acceso se rechaza.

El archivo Web.config predeterminado de nivel superior para un determinado equipo permite el acceso a todos los usuarios. A menos que una aplicación se configure de forma contraria (y suponiendo que un usuario reciba autenticación y pase la comprobación ACL para autorización de archivos), el acceso se concede.

Cuando se comprueban las funciones, la autorización de URL recorre en sentido descendente la lista de funciones configuradas.

Lo que esto significa para la aplicación es que se puede utilizar una clase personalizada que implemente System.Security.Principal.IPrincipal para suministrar su propia semántica de asignación de funciones.

1.8.14. Rendimiento de las Aplicaciones ASP.NET

Aquellas aplicaciones Web que presenten muchas características no serán de utilidad si no funcionan correctamente. Las exigencias de las aplicaciones Web son de tal naturaleza que, ahora más que nunca, se espera del código que haga más en menos tiempo.

ASP.NET proporciona varias mejoras de rendimiento integradas. Por ejemplo, las páginas sólo se compilan una vez y se almacenan en caché para posibles solicitudes posteriores.

Como estas páginas compiladas se guardan en disco, seguirán siendo válidas incluso después de reiniciar completamente el servidor. ASP.NET también almacena en caché objetos internos, como variables de servidor, para acelerar el acceso del código del usuario.

ASP.NET aprovecha todas las mejoras de rendimiento de la biblioteca Common Language Runtime: compilación incremental (Just-in-time), una biblioteca Common Language Runtime optimizada para equipos de un solo procesador o multiprocesador, etc.

1.8.14.1. Métricas de Rendimiento

- Rendimiento: número de solicitudes que una aplicación Web puede atender por unidad de tiempo, medido en solicitudes/segundo. El rendimiento puede variar en función de la carga (número de subprocesos cliente) del servidor.
- Tiempo de respuesta: tiempo transcurrido entre la emisión de una solicitud y el primer byte devuelto al cliente desde el servidor. Éste suele ser el aspecto del rendimiento que mejor puede percibir el usuario cliente. El tiempo de respuesta puede variar independientemente de la tasa de rendimiento.
- Tiempo de ejecución: tiempo que tarda en procesarse una solicitud, generalmente entre el primer y el último byte devuelto al cliente desde el servidor. El tiempo de ejecución afecta directamente al cálculo del rendimiento.
- Escalabilidad: medida de la capacidad de una aplicación de ofrecer mejor rendimiento a medida que se le asignen más recursos (memoria, procesos o equipos). Generalmente se trata de una medida de la tasa de cambio del rendimiento con respecto al número de procesadores.

La base de la programación de aplicaciones con buen rendimiento es lograr un equilibrio entre estas métricas.

El análisis de las métricas unidas, nos pueden ofrecer una imagen aproximada del rendimiento de una aplicación. [www.020]

1.8.14.2. Sugerencias para Mejorar el Rendimiento de las Aplicaciones ASP.NET

a. Deshabilite el estado de sesión cuando no lo utilice:

No todas las páginas y aplicaciones requieren estado de la sesión para cada usuario. Si no es necesario, deshabilítelo completamente. Esto se consigue fácilmente mediante una directiva para páginas, como la siguiente:

```
<%@ Page EnableSessionState="false" %>
```

b. Elija cuidadosamente el proveedor de estado de sesión:

ASP.NET proporciona tres formas diferentes de almacenar datos de sesión para la aplicación: estado de sesión en proceso, estado de sesión fuera de proceso como un servicio de Windows y estado de sesión fuera de proceso en una base de datos SQL.

Cada una de las formas tiene ventajas, pero el estado de sesión en proceso es con mucho la mejor solución. Si sólo se almacenan pequeñas cantidades de datos volátiles en el estado de sesión, sería conveniente utilizar el proveedor en proceso.

Las soluciones fuera de proceso son útiles principalmente en escenarios de unidad Web, o en situaciones en las que no se pueden perder datos en caso de reinicio del servidor o del proceso.

c. Evite realizar demasiados viajes de ida y vuelta al servidor:

El marco de trabajo de página de formularios Web es una de las mejores características de ASP.NET, porque permite reducir en gran medida la cantidad de código que se debe escribir para realizar una tarea. El acceso mediante programación a elementos de página con controles de servidor y el modelo de control de eventos de devolución automática son posiblemente las características que permiten ahorrar más tiempo.

Existen formas apropiadas y formas no tan apropiadas de utilizar estas características, y es importante saber cuándo es apropiado utilizarlas.

Normalmente, una aplicación sólo necesita realizar un viaje de ida y vuelta al servidor para recuperar o almacenar datos. La mayoría de las manipulaciones de datos pueden realizarse en el cliente, entre viajes de ida y vuelta.

Por ejemplo:

La validación de entradas de formulario suele tener lugar en el cliente antes de que el usuario envíe datos.

Si se tiene que devolver información al servidor, no se debe hacer un viaje de ida y vuelta al servidor.

d. Utilice pocos controles de servidor y de forma apropiada:

Los controles de servidor no siempre son la mejor opción. En muchos casos se puede lograr el mismo objetivo mediante procesamiento simple o sustitución de enlace de datos.

e. Evite utilizar demasiado el estado de vista de un control de servidor:

La administración automática de estados es una característica que permite a los controles de servidor volver a llenar sus valores en un viaje de ida y vuelta sin que sea necesario programar código. Sin embargo esta característica no es libre, ya que el estado de un control se pasa al servidor y se recibe del servidor en un campo de formulario oculto. Debe saber cuándo resulta útil ViewState y cuando no.

La opción ViewState está habilitada de forma predeterminada para todos los controles de servidor. Para deshabilitarla, establezca el valor de la propiedad EnableViewState del control en false.

También puede desactivar ViewState para la página. Esto es útil cuando no se realizan devoluciones automáticas desde una página.

Tenga en cuenta que también se admite este atributo mediante la directiva User Control.

f. Utilice el método Response.Write para concatenar cadenas:

Utilice el método HttpResponse.Write en las páginas o controles de usuario para concatenar cadenas. Este método ofrece servicios de búfer y concatenación muy eficaces. En el caso de que pretenda realizar una concatenación compleja.

g. No utilice las excepciones del código:

Las excepciones son costosas y raramente se producen en el código. Nunca debe utilizar las excepciones como una forma de controlar el flujo normal de un programa. Si es posible detectar en el código una condición que puede causar una excepción, debe hacerlo en lugar de esperar a detectar la excepción antes de controlar la condición.

h. Utilice enlace en tiempo de compilación en código de Visual Basic o JScript:

Una de las ventajas de Visual Basic, VBScript y JScript es su naturaleza de lenguaje sin tipos. Es posible crear variables simplemente utilizándolas sin necesidad de una declaración explícita de tipo. Al realizar una asignación de un tipo a otro, las conversiones también se realizan automáticamente. Esto puede ser una ventaja y una desventaja, ya que el enlace en tiempo de ejecución es una utilidad muy costosa en términos de rendimiento.

i. Almacene en caché datos y resultados siempre que sea posible:

El modelo de programación ASP.NET proporciona un mecanismo sencillo para almacenar en caché resultados o datos de páginas cuando no sea necesario calcularlos dinámicamente para cada solicitud.

Puede diseñar las páginas con almacenamiento en caché con el fin de optimizar los puntos de la aplicación en los que espera que haya más tráfico. El uso apropiado del almacenamiento en caché es más importante que ninguna otra característica de la plataforma .NET Framework con el fin de mejorar el rendimiento del sitio, a veces en un orden de magnitud e incluso más. [www.021]

j. Habilite el uso de unidades Web para equipos multiprocesador:

El modelo de procesos de ASP.NET permite habilitar la escalabilidad en equipos con varios procesadores distribuyendo el trabajo entre varios procesos, uno para cada CPU, cada uno con la afinidad de procesador establecida en la CPU correspondiente. Esta técnica se denominada Uso de unidades Web y permite mejorar en gran medida el rendimiento de algunas aplicaciones.

En las últimas versiones de .net Framework la función processModel tiene una autoconfiguración para el manejo o trabajo con multiprocesadores.

k. Deshabilitar el modo de depuración:

La sección <compilación> de la configuración de ASP.NET controla si una aplicación se compila en modo de depuración o no. El modo de depuración degrada en gran medida el rendimiento. Recuerde siempre que debe deshabilitar este modo antes de implantar una aplicación de producción o medir el rendimiento. [www.021]

1.8.14.3. Recomendaciones para la Configuración de las Aplicaciones ASP.NET

En el desarrollo de aplicaciones web ASP debemos tener en consideración los siguientes ítems para la configuración:

a. Custom Errors deshabilitado:

Al tener deshabilitado esta opción, los errores que se producen en la aplicación se muestran a todos. Los valores adecuados son On y RemoteOnly, en el primero se puede especificar páginas personalizadas para mostrar mensajes de error más amigables.

b. Dejar habilitado el seguimiento de página (Tracing):

Esta característica de ASP.NET contiene información bastante detallada de todo lo que pasa en la aplicación como registro de instrucciones relacionadas con el principio y el final de métodos de la ejecución, tales como Init, Render y PreRender; variables o encabezados de formulario y objetos QueryString, jerarquía de controles, estado de sesión y estado de aplicación.

Un atacante puede utilizar toda esta información accediendo simplemente a trace.axd. Para lograr vulnerar nuestras aplicaciones. [www.022]
El valor adecuado para esta opción es:

```
<trace enabled="false" localOnly="true">
```

c. Depuración habilitada:
Tener esta opción habilitada, no sólo afecta al rendimiento de la aplicación, sino también que se muestran errores más detallados.
El valor adecuado para esta opción es:

```
<compilation debug="false">
```

d. Cookies accesibles desde código de cliente:
Usar cookies del tipo HttpOnly, técnica que mitiga de algún modo el robo de cookies. Hay que tener en cuenta que esta característica no es la panacea para asegurar nuestra aplicación, además si se habilita esta opción a nivel de la aplicación, puede traer ciertos problemas al momento de trabajar con cookies en código de cliente.

e. Cookies de estado de sesión enviadas por Url:
Es recomendable usar cookies para almacenar el ID de la sesión, de este modo se evitaría que un atacante suplante a un usuario válido al robar este valor usando algún sniffer.

f. Cookies de autenticación enviadas por Url:
Es recomendable usar cookies para almacenar valores de autenticación de usuario o sesión.

g. No usar SSL para transmitir las cookies de autenticación:

El uso de SSL en teoría permite la comunicación segura de datos, pero no es estrictamente necesario arriesgarnos al tener varias opciones más seguras de hacerlo.

h. Uso de slidingExpiration en la autenticación:

El tener este valor habilitado (<forms slidingExpiration="true">), permite que el tiempo de vida de la sesión vaya cambiando a medida que el usuario vaya interactuando con la aplicación. Sin embargo, con esta opción deshabilitada sólo se dispone de un tiempo fijo para la sesión, de esta manera si alguien suplanta a usuarios legítimos sólo tendrá un espacio de tiempo limitado.

i. Cookies de autenticación no únicas:

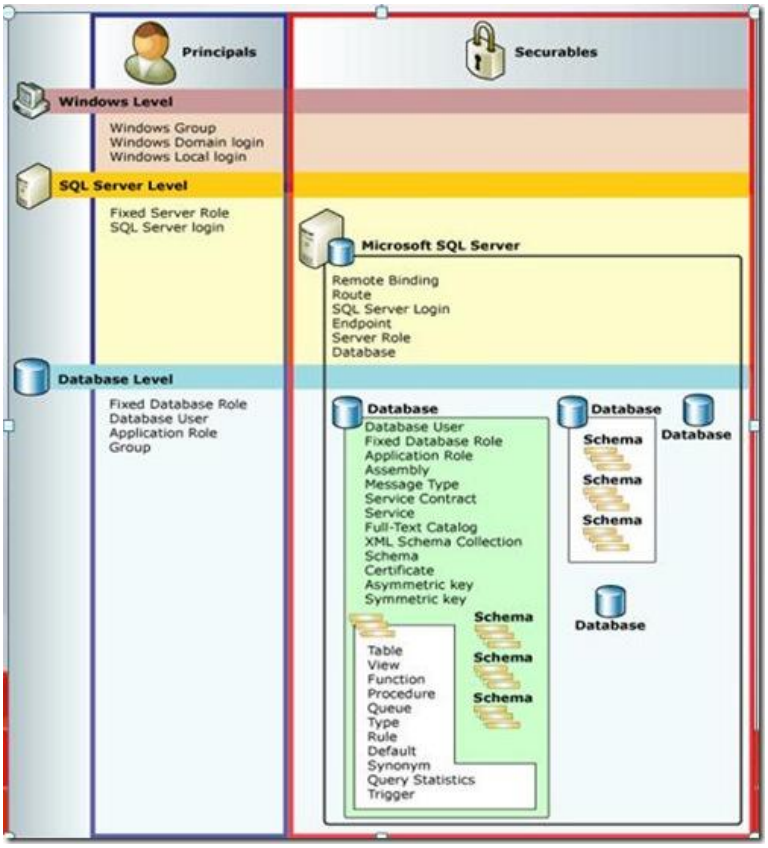
Uso de diferentes valores para el nombre de la cookie de autenticación (<forms name="otro_valor">) en aplicaciones ASP.NET diferentes (para los casos en los que existen más de 2 aplicaciones en un mismo dominio).

j. Almacenar las credenciales de usuario en Web.config:

En determinado momento un pirata podría acceder al archivo Web.config y si allí tiene todas las credenciales de los usuarios de una aplicación, sería un completo desastre, por lo que es conveniente tener este tipo de información en un lugar no estándar o conocido. [www.022]

CAPITULO II

APLICACIONES WEB CON SQL SERVER 2005



2.1.APLICACIONES WEB UTILIZANDO SQL SERVER 2005

2.2.SEGURIDAD SQL SERVER 2005 APLICADA EN UNA RED

2. Aplicaciones Web con SQL Server 2005

2.1.Aplicaciones Web Utilizando SQL Server 2005

El presente de las Aplicaciones Informáticas, están orientadas al trabajo en la Web y con respuestas en tiempo real con respecto al procesamiento de la información administrada por el mismo.

En la actualidad, la capacidad de los Sistemas Web, no se pueden ver mermadas por la inconsistencia de información o procesamiento de los datos y mucho menos por el acceso a los mismos.

SQL Server 2005 nos brinda múltiples ventajas en el momento de acoplamiento con aplicaciones web ya sea de una intranet o internet.

Su rendimiento con aplicaciones en la web, acceso y administración de la información contenida en sus bases de datos, esta fijada por varias de sus características tales como:

- Compatibilidad con XML: Podemos simplificar la integración de sistemas de servicios de fondo y la transferencia de datos a través de servidores de seguridad mediante XML por su compatibilidad total.
- Acceso Web a los datos: Conexión de las bases de datos de SQL Server 2000 y cubos OLAP de manera flexible, mediante el Web sin necesidad de ninguna programación adicional.
- Alojamiento de aplicaciones: Gracias a la compatibilidad con varias instancias, SQL Server permite aprovechar totalmente las inversiones en hardware de forma que múltiples aplicaciones se pueden ejecutar en un solo servidor o externamente.
- Seguridad: Todas las aplicaciones son seguras en cualquier entorno de red, con la seguridad basada en funciones y el cifrado de archivos y de la red.
- Alta disponibilidad: Aumenta la disponibilidad de las aplicaciones empresariales con el trasvase de registros, las copias de seguridad en línea y los clústeres de conmutación por error.
- Escalabilidad: Podemos escalar las aplicaciones hasta 32 CPU y 64 gigabytes (GB) de RAM.
- Vistas con particiones distribuidas: es factible repartir la carga entre varios servidores para obtener más escalabilidad.
- Vistas indexadas: podemos obtener rendimiento del hardware existente al almacenar los resultados de las consulta y reducir los tiempos de respuesta.
- VI SAN: Podemos mejorar el rendimiento general del sistema con la

compatibilidad integrada con una red de área de sistema virtual (VI SAN, Virtual System Área Network).

- Duplicación: Con SQL Server 2000 se puede implementar duplicaciones de mezcla, transaccionales y de instantáneas con sistemas heterogéneos.
- Administración de bases de datos simplificada: Las características de ajuste y mantenimiento automáticos permiten a los administradores centrar su atención en otras tareas críticas.
- Productividad del desarrollador mejorada: Las funciones definidas por el usuario, la integridad referencial en cascada y el depurador integrado de Transact SQL le permiten la reutilización del código para simplificar el proceso de desarrollo.
- Servicios de transformación de datos: Podemos automatizar las rutinas de extracción, transformación y carga de datos de orígenes heterogéneos.
- Extensión de las aplicaciones: La compatibilidad con el acceso por parte de dispositivos, como unidades portátiles de Microsoft Windows CE, proporciona acceso más amplio a las aplicaciones y extiende su base de usuarios. [www.023]

2.2.Seguridad SQL Server 2005 Aplicada en una Red

En SQL Server nos encontramos con tres niveles o capas en los cuales podemos gestionar la seguridad.

El primero de ellos se encuentra a nivel de servidor, en él podemos gestionar quién tiene acceso al servidor y quién no, además de gestionamos que roles va a desempeñar. Para que alguien pueda acceder al servidor debe tener un inicio de sesión (login) asignado, y a éste se asignaremos los roles o funciones que puede realizar sobre el servidor.

La siguiente barrera de seguridad es a nivel de base de dato. Para que un login tenga acceso a una base de datos, tenemos que crear en ella un usuario (user).

Todas y cada una de las bases de datos deben tener un usuario y un password para que pueda usar un login de autenticación para su acceso. Análogamente, el que un usuario tenga acceso a una base de datos no quiere decir que tenga acceso a todo su contenido, ni a cada uno de los objetos que la componen, e aquí la tercera barrera de seguridad. Para que esto ocurra tendremos que irle concediendo o denegando permisos sobre cada uno de los objetos que la componen. [www.023]

A continuación podemos observar un gráfico que refleja este modelo:

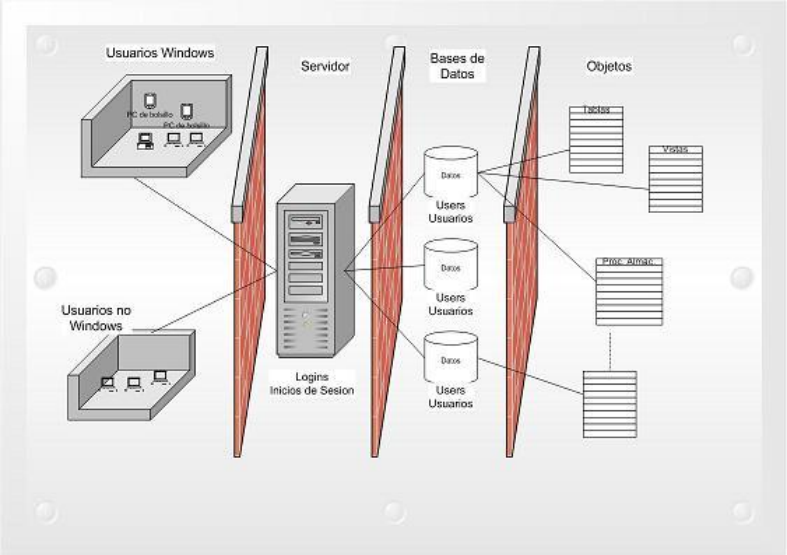


FIGURA 2.1: Capas de Seguridad de SQL Server 2005

2.2.1. Asegurables de SQL Server 2005

SQL Server administra una colección de entidades que se protegen mediante permisos, a esto se le conoce como “asegurables”.

Las entidades asegurables mas conocidas son los servidores y las bases de datos.

SQL Server administra las acciones que se toman sobre dichas entidades asegurables comprobando en cierta forma que se les ha otorgado los permisos a dicho objeto como se muestra en la Figura 2.2 [www.012]

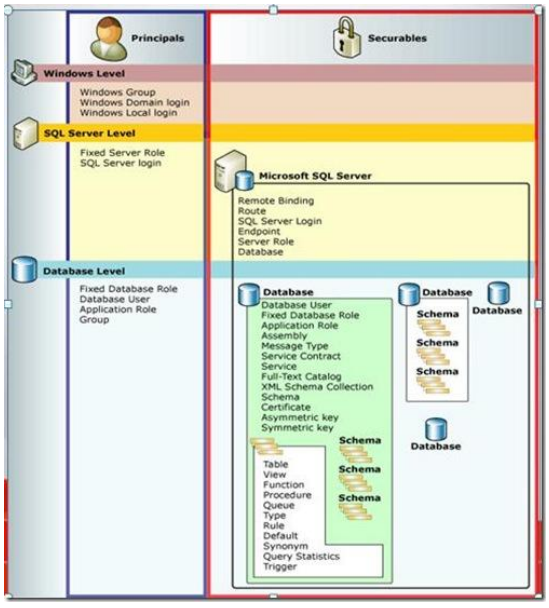


FIGURA 3: Capas de Seguridad de SQL Server 2005

Los individuos, grupos y procesos que pueden solicitar recursos a SQL Server se les llaman “principales”.
Se debe tomar en cuenta el ámbito de su definición (Puede ser Windows, Servidor, o de base de datos), y si es un principal primario o un principal secundario.
Un ejemplo es:
Un principal primario es un inicio de sesión de Windows, y de secundario es un grupo de Windows tal como observamos en la Figura 3.
Existen niveles principales:

a. Principales de nivel de Windows

- Inicio de sesión en dominio de Windows.
- Inicio de sesión local de Windows.
- Grupo de Windows.

b. Principales de nivel de SQL Server

- Inicio de sesión de SQL Server (puede estar asignado a un principal de nivel de Windows).
- Función de servidor.

c. Principales de nivel de base de datos

- Usuario de la base de datos (puede estar asignado a un principal de nivel de servidor).
- Función de base de datos, puede ser una función de base de datos fija o una función definida por el usuario.
- Función de aplicación.

Las entidades asegurables son aquellos recursos que están controlados mediante autorizaciones en SQL Server, podemos checar que algunas entidades asegurables pueden estar dentro de otras, y crean lo que son jerarquías anidadas que se llaman “ámbitos”, y que también se pueden controlar mediante autorizaciones.
Los ámbitos asegurables son servidor, base de datos y esquema.
Podemos checar cada uno de ellos, de manera global, sin profundizar en ellos.
[www.012]

2.2.1.1. Ámbito Asegurable: Servidor

Contiene las siguientes entidades asegurables:

- Inicios de sesión.
- Extremos de http.
- Certificados.
- Notificaciones de eventos (eventos de DDL, eventos de rendimiento, eventos de error, eventos de idioma, eventos de recurso, eventos de seguridad, eventos de copia de seguridad).
- Bases de datos.

2.2.1.2. Ámbito Asegurable: Base de Datos

Contiene las siguientes entidades asegurables:

- Usuarios
- Funciones
- Funciones de aplicación
- Ensamblados
- Tipo de mensaje
- Contrato de servicio
- Servicio
- Catálogo de texto
- Eventos de DDL
- Esquema XML (sólo en vista previa)
- Esquema

2.2.1.3. Ámbito Asegurable: Esquema

Contiene las siguientes entidades asegurables:

- Tabla
- Vista
- Función
- Procedimiento
- Cola
- Tipo
- Regla
- Valor predeterminado
- Sinónimo
- Agregado

2.2.2. Informática Fiable (Trustworthy Computing) y SQL Server 2005

La iniciativa de la Informática Fiable, delimita un marco que define los pasos necesarios para respaldar la seguridad informática y tomar las medidas que servirán al desarrollo y el mantenimiento de un entorno seguro. Estos pasos ayudan a proteger la confidencialidad, la integridad y la disponibilidad de datos y sistemas en cada fase del ciclo de vida del software (el diseño, la entrega, el mantenimiento). [www.025]

Para Microsoft, los objetivos de la iniciativa de la Informática Fiable son:

- Reducir problemas potenciales de seguridad a partir del diseño y testeo de productos.
- Reducir el área de ataques potenciales a partir de la inhabilitación de funciones que pueden resultar innecesarias. También asegurar que, durante su instalación, el producto elija los valores de configuración correctos para todas las opciones.
- Proporcionar las herramientas y las guías necesarias que fomentan las acciones de protección, detección, defensa, recuperación y mantenimiento.
- Crear la documentación necesaria que comunique de forma regular la última información sobre seguridad, para que los mismos clientes puedan ocuparse de la seguridad e integridad de su entorno de SQL Server.

En consonancia con los cuatro preceptos de la iniciativa de la Informática Fiable, Microsoft y SQL Server adoptaron las siguientes medidas:

a. Seguridad en el diseño

Durante el desarrollo de SQL Server 2005 se realizó numerosas intervenciones en materia de seguridad.

Para cada amenaza potencial de seguridad, se realizó un análisis de la situación y realizó trabajos de diseño y testeo suplementarios que apuntan a neutralizar problemas potenciales de seguridad. Como resultado de estos esfuerzos de diseño, SQL Server 2005 incluye nuevas funciones de seguridad.

b. Seguridad por defecto

Para su instalación, SQL Server 2005 elige los valores de configuración correctos para todas las opciones, asegurándose así de que la instalación de los sistemas se realice en el estado seguro por defecto.

c. Seguridad en el desarrollo

Microsoft creó contenido para que las organizaciones puedan desarrollar SQL Server con las credenciales de seguridad apropiadas y para que comprendan los pasos y permisos requeridos. Las herramientas de desarrollo de SQL Server proporcionan la información necesaria para comprender las decisiones que deben tomarse durante la etapa de desarrollo. Por otra parte, las actualizaciones de seguridad son fáciles de encontrar y de instalar. Las herramientas también se encuentran disponibles para enfrentar los riesgos de seguridad que corren las organizaciones.

d. Comunicaciones

Para asegurar el soporte del desarrollo de SQL Server, Microsoft difunde comunicaciones permanentes sobre cuestiones de seguridad. La página sobre Recursos de Seguridad sirve como un archivo central con toda la información sobre seguridad relacionada con SQL Server, incluyendo amenazas de seguridad y parches y herramientas que sirven para mitigar dichas amenazas.
SQL Server 2005 incorpora mejoras y funciones de seguridad que coinciden con los objetivos de la iniciativa de la Informática Fiable. Estas funciones y mejoras recaen en las siguientes áreas:

- Restricción del acceso de usuarios al servidor

  SQL Server 2005 proporciona más control de acceso a SQL Server y permite que los administradores controlen dicho acceso a través de distintas políticas.

- Inhabilitación de servicios y restricción de la configuración del servicio

  Los administradores podrán restringir el acceso a los recursos de SQL Server en función del alcance otorgado al administrador y con una fina variedad de opciones. Los administradores también contarán con un sistema manejable que no viola el principio de menos privilegios. Al contar con ciertos servicios inhabilitados por defecto para nuevas instalaciones de servidores, los administradores se involucrarán de un modo más activo en la elección de los servicios que pretenden habilitar.

- Reducción del área de ataques potenciales en las nuevas funciones

Empezando por la configuración e instalación de SQL Server, el área de ataques potenciales se encontrará minimizada. A través del ciclo de desarrollo del producto, se ha repasado y evaluado la seguridad de las nuevas funciones para ayudar a reducir la superficie de ataque.

2.2.3. Métodos de Autenticación

Como introducción debemos tener claro cuales son los significados de Login y User.

Los Login son accesos al servidor. El hecho de que alguien acceda a al servidor a través de un login no quiere decir que cual puedan acceder a las bases de datos que allí se encuentran. Para poder acceder a cada una de las bases de datos necesita un user. [www.011]

User es un usuario de la base de datos. Un user proporciona acceso a la base de datos, pero esto tampoco quiere decir que pueda hacer cualquier operación sobre la base de datos, el user tiene la capacidad de utilizar solamente los recursos asignados al mismo por un administrador de cuentas user.

2.2.3.1. Autenticación Windows

Ventajas:

e. Duración de passwords.
f. Atributos de passwords (longitud).
g. Auditoría.
h. Bloqueo de cuentas por reintentos.
i. Kerberos.

Desventajas:

- De no haber un controlador de Dominio disponible no se podrá establecer conexión.
- Todo cambio en la seguridad tendrá efecto al iniciar una sesión.
- El DBA necesita derechos administrativos en el servidor Windows para hacer estos cambios.

2.2.3.2. Autenticación Mixta (SQL y Windows)

Ventajas:

- El DBA no necesita derechos administrativos en el servidor Windows para hacer estos cambios

Desventajas:

- No ofrece el control y flexibilidad de Windows

2.2.4. Agregación y Eliminación de Logins al Servidor

Una de las grandes ventajas que posee SQL Server 2005 es la capacidad de crear un número indefinido de usuarios, los que van a tener roles, o capacidades dentro de una o varias bases de datos existentes en el servidor y la capacidad de acceso al uso y administración de las mismas mediante diversas herramientas aplicadas a un propósito definido, ya sea de monitoreo, control ó acceso.
Para la eliminación de un login tomamos en cuenta si vamos a cambiar de login y al momento de eliminar uno crearemos o asignaremos otro a ese acceso de recurso.

2.2.5. Restricción de Acceso al Servidor

Un administrador de la Base de Datos, tiene la capacidad de revocar o restringir el acceso a uno o más usuarios mediante la edición de sus roles o privilegios de usuario concedidos al momento de su creación.
La revocación es utilizada para los administradores temporales o desarrolladores de bases de datos, los mismos que durante el desarrollo e implementación de la base de datos, necesitan un acceso continuo y temporal.
La restricción es utilizada para que nuestros usuarios ya sean temporales o de base tengan acceso a determinados recursos para la administración en el servidor de los diversos recursos y campos existentes. [www.026]

2.2.6. Roles

Un rol es un conjunto de privilegios definidos para un determinado login o user perteneciente al Servidor.

2.2.6.1. Tipo de Roles

- Fixed Server Roles: Rol fijado o establecido en un servidor como por ejemplo Windows Server versión 'X'.
- Fixed Database Roles: Rol fijado o establecido en la Base de Datos.
- User-defined Database Roles: Rol definido para un user de Base de Datos.
- Application Roles: Rol definido para una determinada aplicación.

a. Server Roles

Le dan al DBA (Administrador de Base de Datos) la capacidad de asignarles a los usuarios permisos a nivel servidor.
El / los DBAs deberían pertenecer a un grupo de Windows 2000 que se asigne al fixed-server role Sysadmin. [www.026]

Server Role	Descripción
Sysadmin	Puede llevar a cabo cualquier tarea
Serveradmin	Puede realizar cambios en los seteos del administrador, incluso apagarlo.
Setupadmin	Encargado de la configuración de servidores vinculados
Securityadmin	Encargado de Manejo de logins, creación de base de datos, logs, errores, users, passwords.
Processadmin	Encargado de manejo de procesos en ejecución.
Dbcreator	Con capacidad de crear modificar y eliminar bases de datos.
Diskadmin	Administrador de discos
Bulkadmin	Puede ejecutar tareas de Bulk (asignación de tamaño o volumen).

Tabla 2.1 Grupos de Roles de un Servidor SQL 2005

b. Database Roles

Le dan al DBA y al owner (propietario) de cada base de datos la capacidad de asignarles a los usuarios permisos administrativos a nivel base de datos.

Database Role	Descripción
Db_owner	Puede llevar acabo cualquier tarea en una base de datos.
Db_accessadmin	Puede agregar o remover IDs.
Db_securityadmin	Maneja permisos, owerships y roles.
Db_ddladmin	Manejo de todos los ddl menos grant, revoke o deny.
Db_backupoperator	Capacidad de realizar backups, dbcc y checkpoint.
Db_datareader	Accede a cualquier tabla de la base asignada.
Db_datawriter	Modifica cualquier tabla de la base asignada.
Db_denydatareader	No puede acceder a los datos.
Db_denydatawriter	No puede modificar datos en las tablas

Tabla 2.2 Roles de Usuarios de un servidor SQL 2005

c. User-Definied Database Roles

Le dan al DBA y al owner de la base de datos la capacidad de manipular los derechos de forma más personalizada.
Es el único tipo de rol que se puede crear y es siempre a nivel de la base de datos.

d. Application Roles

Le dan al DBA la capacidad de asignarle a una aplicación un conjunto de derechos.
La asignación de un password para este rol se la puede asignar con o sin encriptación.
Estos roles no contienen usuarios, solo se activan.
Cuando se activa el role, los derechos actuales se ignoran.

No se puede acceder a otras BBDD, salvo si existe el user Guest en ellas y tiene permisos.

Un role de aplicación no se puede desactivar, se debe cerrar la conexión.

Para proteger la password, se la puede grabar en el Registry donde solo tenga acceso la aplicación.

2.2.7. Permisos

Se pueden dar permisos sobre columnas en las tablas, pero sobre filas se debería hacer mediante stored procedures o funciones que realicen el filtrado.

Los permisos se deberían otorgar sobre estos objetos no sobre las tablas directamente.

Al trabajar con permisos se deben considerar:

- La naturaleza del permiso.
- El tipo de permiso.
- Permisos efectivos.

2.2.7.1. Naturaleza de los permisos

Se los puede clasificar por:

- Objecto.
- Statement.
- Predefinidos.
- Implicitos / Derivados.

a. Naturaleza de los permisos – Objeto

Se le otorgan a un usuario sobre un objeto determinado de la BBDD: tabla, vista, stored procedure o función.

Incluyen SELECT, INSERT, UPDATE, DELETE,

REFERENCES y EXECUTE

b. Naturaleza de los permisos – Statement

Se le otorgan a un usuario para permitirle crear objetos en una BBDD.

Incluyen los Create : Database, Default, Index, View, Function, Procedure, Rule, Schema, Table, Trigger, Backup Database y Backup Log.

c. Naturaleza de los permisos – Predefinidos

Son los que se adquieren por el hecho de pertenecer a un Server role o a un Database role. Estos permisos son parte de la definición misma del role y no se pueden modificar.

d. Naturaleza de los permisos – Implícitos/ Derivados

Son los que se tienen por el hecho de ser el owner del objeto: control total sobre él.

2.2.7.2. Tipo de permiso

Los tipos de permiso son:

- Grant
- Deny
- Revoke

a. Tipo de permiso - Grant

Otorga explícitamente el permiso para manipular datos o ejecutar stored procedures en la base actual.

Ejemplo:

```
GRANT SELECT, INSERT, UPDATE, DELETE
ON Tabla_Ejemplo TO Role / Grupo / User
```

b. Tipo de permiso – Deny

Elimina el GRANT / REVOKE que hubiere.

Ejemplo:

```
DENY INSERT, UPDATE, DELETE
ON Tabla_Ejemplo TO Role / Grupo / User
```

c. Tipo de permiso - Revoke

Elimina el GRANT / DENY que hubiere y no deja activado ninguno en particular.

Ejemplo:

```
REVOKE SELECT, INSERT, UPDATE, DELETE
ON Tabla_Ejemplo TO Role / Grupo / User
```

Permisos efectivos

Se otorgan a usuarios y roles y son específicos por cada BBDD. [www.026]

2.2.8. Consideraciones de Control de Seguridad en SQL Server 2005

a. Consideraciones generales sobre los Passwords

Las passwords deberían ser suficientemente seguras para no ser 'adivinadas'.
Si se usa seguridad de SQL, el DBA debería asignar las passwords y no utilizar cosas simples, comunicándolas por mail o por un mensaje telefónico.
Si se emplea una política de seguridad en las passwords hay que estar en condiciones de mantenerla. [www.027]

b. Consideraciones sobre el usuario SA

El usuario SA creado por defecto en la instalación debe tener las siguientes consideraciones:

- Debe tener un passwords complejas (aún si no use seguridad de SQL).
- Solo se mantiene por compatibilidad.
- Para los DBAs, se debe utilizar un grupo de Windows y agregarlo en el fixed-server role SysAdmin.

c. Documentación a mantener

En todo servidor se debería mantener actualizado lo siguiente:

- Modo de autenticación.
- Miembros del grupo Administradores Locales.
- Miembros de SysAdmin y todo fixed-server role.
- Miembros de los roles de usuario.
- Los permisos dados sobre cada objeto.
- Todo cambio de seguridad en el servidor.

d. Mantenimiento del servidor

Es fundamental en todo servidor mantener actualizado el sistema operativo y el SQL Server, esto incluye:

- Service packs.
- Parches.

e. Seguridad de los scripts

Las unidades o carpetas de los scripts de administración del servidor deberían estar acotadas a los DBAs.
Nunca se debe colocar passwords dentro de los scripts que usen seguridad de SQL Server, accedan a FTPs, etc.

f. Consideraciones de Usuarios

Ningun usuario deberia:

- Tener Master como base por defecto.
- Poder crear objetos en Master / Msdb / Model.

g. Consideraciones de Bases de Datos

Se debe eliminar las bases de Datos Pubs y Northwind de los servidores de producción.
Se debe tener control sobre el role Public.
Remover Guest de todas las bases y quitarlo de Model.

h. Seguridad física

El hardware debe estar protegido del acceso irrestricto en sitios con llave y gabinetes con traba.
Nunca dejar una sesión abierta.
Los archivos de SQL deberían estar en particiones NTFS con su correspondiente seguridad aplicada.
Solo deberían accederlos: la cuenta de los servicios / SYSTEM / local Administrators. [www.027]

i. Seguridad lógica

No es tarea del DBA, pero es fundamental que este integrado todo
Firewalls, DMZs (demilitarized zone, red perimetral), etc.

j. Seguridad del Registro

Everyone debería tener permiso de Read solamente
Full Control otorgado a:

- La cuenta de servicios de SQL server
- Local SYSTEM

Las Instancias de SQL usan juegos separados de claves

k. Logins y Permisos

Las recomendaciones de Microsoft con respecto a la creación de Logins y permisos es la siguiente:

- Agregar los usuarios de W2003 en grupos de W2003.
- Crear un único Login para ese grupo.
- Crear un usuario en la BBDD para ese Login.
- Agregar ese usuario a un User Database Role.
- Asignar permisos sobre vistas y stored procedures a ese role.

CAPITULO III

DESARROLLO E IMPLEMENTACIÓN DE APLICACIONES INFORMÁTICAS WEB

3.1.Metodología para Desarrollo de Software

3.2.Desarrollo Ágil de Software

3.3.Diseñando de Aplicaciones Distribuidas Orientadas a la Web

3. Desarrollo e Implementación de Aplicaciones Informáticas Web

3.1. Metodología para Desarrollo de Software

Una metodología de desarrollo de software es un conjunto de pasos y procedimientos que deben seguirse para desarrollar software. [www.028]

3.1.1. Conceptos Generales

Metodología: Conjunto de procedimientos, técnicas, herramientas y un soporte documental que ayuda a los desarrolladores a realizar nuevo software.

Tarea: Actividades elementales en que se dividen los procesos.

Procedimiento: Definición de la forma de ejecutar la tarea.

Técnica: Herramienta utilizada para aplicar un procedimiento. Se pueden utilizar una o varias técnicas.

Herramienta: Para realizar una técnica, podemos apoyarnos en las herramientas software que automatizan su aplicación.

Producto: Resultado de cada etapa.

Una metodología está compuesta por:

- Cómo dividir un proyecto en etapas.
- Qué tareas se llevan a cabo en cada etapa.
- Qué restricciones deben aplicarse.
- Qué técnicas y herramientas se emplean.
- Cómo se controla y gestiona un proyecto.

3.1.2. Etapas en una Metodología de Desarrollo Software

Las etapas en las que se debe desarrollar un proyecto, son básicamente:

- Requisitos.
- Análisis.
- Diseño Preliminar.
- Diseño Detallado.
- Implementación y Pruebas

a. Requisitos

Los requisitos son una lista de tareas que nuestro sistema debe ejecutar de una manera u otra con un punto de vista eficaz para el usuario y óptimo para el desarrollador.

La recopilación de los requisitos lo haremos a los futuros usuarios del Sistema mediante:

- Entrevistas con todos los futuros usuarios de nuestro sistema.
- Preguntas cerradas y/o Abiertas.
- Encuestas.

La siguiente tarea del recolector de requisitos es ordenar la información para que la interpretación a darse durante el desarrollo del Sistema sea óptima y precisa.

b. Análisis

Durante el análisis vamos a definir más claramente qué es lo que va a hacer nuestro sistema.

Para ello debemos:

- Identificar actores. En lenguaje UML, actores son los usuarios y cualesquiera otros sistemas con los que se pueda comunicar nuestro programa.
- Identificar casos de uso. Un caso de uso es una determinada actividad que un actor desea ejecutar con nuestro sistema. Debemos obtener una lista de actividades que los actores desearan realizar en nuestro sistema. Se debe considerar que las tareas a ejecutarse sean un conjunto de acciones que conforme una actividad y no caer un sub actividades.
- Detallar casos de uso. Realizamos un desglose y descripción de las actividades a realizarse por determinado actor Es decir, explicamos por escrito, desde el punto de vista del actor (usuario), qué es lo que tiene que hacer y qué es lo que va a hacer el sistema al ejecutar determinada acción.
- Realizar el Diagrama de clases del negocio. Es un diagrama de clases de objetos que tienen sentido para el usuario. Este diagrama nos ayuda a saber que los que saben del tema (futuros usuarios) y los que hacen el sistema (desarrollador) están hablando de lo mismo y no han existido malentendidos en las entrevistas o errores por falta de conocimiento. Esta

acción nos servirá de base para el diseño detallado orientado a objetos. [www.029]

c. Diseño Preliminar

En el diseño preliminar tratamos de establecer la arquitectura de nuestro sistema. La arquitectura es un esquema de en qué módulos/paquetes vamos a dividir nuestro programa, qué librerías. Si el sistema es suficientemente grande, quizás vaya en varios ejecutables, una arquitectura cliente/servidor, etc.

Al observar los casos de usos, deberíamos ver qué cosas podemos hacer comunes o como librerías aparte, clases comunes, módulos o subsistemas que podamos reutilizar.

En este punto y con los casos de uso en general, debemos tener cuidado. Según una crítica generalizada a los casos de uso, estos llevan a un diseño funcional y no a uno orientado a objetos. Debemos tratar de pensar en objetos y almacenarlos juntos en la misma librería cuando tengan una estrecha relación entre sí. Mediante esta consideración, es buena idea tratar de agrupar las clases del diagrama de clases del negocio en paquetes y tratar de desarrollar la arquitectura a partir de ellas.

Es importante en este paso definir las interfaces y relaciones entre paquetes. Para ello puede servir de ayuda hacer los diagramas de secuencia de los casos de uso mostrando los actores, los paquetes y los mensajes entre ellos. Según vayan creciendo los diagramas de secuencia por aquello de ir entrando en detalles, podremos ir extrayendo sub casos de uso.

d. Diseño Detallado

En el diseño detallado ya se entra a nivel de clases y métodos. Por cada paquete, módulo o subsistema que hayamos extraído en el paso anterior y siguiendo siempre los casos de uso, debemos ir detallando las clases que vamos a implementar y los métodos que van a tener. Detallamos aun más los casos de uso y las interfaces de las clases.

En este punto pueden ser de ayuda los patrones de diseño. Las personas que ha ido diseñando sistemas a lo largo de la historia, se ha encontrado con problemas (¿cómo hago que la clase A se entere que la clase B ha cambiado sin necesidad de que B vea a A?, ¿En qué clase pongo el método que suma las clases A y B? ¿Como me aseguro que de esta clase sólo se haga una instancia?, etc.). Algunos de dichos problemas salen con mucha frecuencia (como los indicados de

ejemplo) y se han establecido soluciones "estándar", que funcionan bien. Dichas soluciones se llaman patrones.

Hay incluso patrones a nivel de arquitectura. Es bastante conocido, por ejemplo, el patrón separación modelo-vista. En nuestro caso "modelo" sería el conjunto de clases del sistema. "Vista" sería la interfaz usuario-sistema.

Es imprescindible que sólo la "vista" vea al "modelo" y nunca al revés. Si lo hacemos al revés y queremos llevarnos nuestro "modelo", tendremos que llevarnos también parte de la "vista".

e. Implementación y Pruebas

De las pruebas podemos decir que hay que escribir los "casos de prueba". Básicamente son como la descripción de los casos de uso, pero indicando datos concretos que el operador va a introducir y qué resultados exactos debe dar nuestro sistema.

3.1.3. Desarrollo Iterativo e Incremental

El Sistema debe desarrollarse en varias iteraciones. Es decir, hacer un poco de requisitos, un poco de análisis, un poco de diseño, un poco de codificación y pruebas y vuelta a empezar.

Hay varios motivos para realizar un desarrollo de esta manera:

- No eternizarse en un paso. Por mucho análisis que hagamos, nunca estaremos seguros de haber acabado y siempre se presentarán más requerimientos después de un período de prueba. Debemos fijarnos plazos determinados para el desarrollo de actividades por ejemplo:
    - Obtener los máximos requisitos posibles en el tiempo fijado. Ordenarlos y fijarnos en los más importantes.
    - Obtener los máximos casos de uso y actores posibles. Ordenarlos por importancia (más importantes para el usuario, más difíciles de implementar o que ayuden a definir lo mejor posible la arquitectura) y detallar sólo los primeros en la lista.
    - Obtener la arquitectura para esos primeros casos de uso.
    - Diseño detallado para esos primeros casos de uso, siempre con un plazo.
    - Codificar y probar.

    - Vuelta a empezar con los requisitos (¿han cambiado?, ¿hay nuevos?), los casos de uso (buscar más casos de uso, detallar los siguientes en importancia), etc.

- Realizar versiones intermedias de nuestro programa. Al momento de implementar varios casos de uso, podemos hacer una versión beta de muestra al cliente. Este punto es especialmente útil cuando el cliente no sabe muy bien qué es lo que quiere. [www.029]

3.1.4. Clasificación de las Metodología de Desarrollo Software

Las metodologías se clasifican de la siguiente forma:

- Estructuradas.
    - Orientadas a procesos
    - Orientadas a datos jerárquicos
    - Mixtas
- No estructuradas.
    - Orientadas a objetos
    - Sistemas de tiempo real

3.1.4.1. Metodología Estructurada

3.1.4.1.1. Metodologías Orientadas a Procesos

La ingeniería del software se basa en el modelo básico de entrada/proceso/salida de un sistema.

Está compuesta por:

- Diagrama de flujo de datos (DFD).
- Diccionario de datos.
- Especificaciones de proceso.

Por ejemplo las metodologías de DeMarco, Gene y Sarson, Yourdon.

a. Método de DeMarco

- Construir el modelo físico actual (DFD físico actual)
- Construir el modelo lógico actual (DFD lógico actual).
- Crear un conjunto de modelos físicos alternativos.
- Estimar los costes y tiempos de cada opción.
- Seleccionar un modelo
- Empaquetar la especificación

b. Método de Gane y Sarson

- Construir el modelo lógico actual (DFD lógico actual).
- Construir el modelo del nuevo sistema: elaborar una especificación estructurada y construir un modelo lógico de datos en tercera forma normal que exprese el contenido de los almacenes de datos.
- Seleccionar un modelo lógico.
- Crear el nuevo modelo físico del sistema.
- Empaquetar la especificación

3.1.4.1.2. Metodologías Orientadas a Datos Jerárquicos

Son metodologías basadas en la información. Primero se definen las estructuras de datos y, a partir de éstos, se derivan los componentes procedimentales.

La estructura de control del programa debe ser jerárquica y se debe derivar de la estructura de datos del programa.

El proceso de diseño consiste en definir primero las estructuras de los datos de entrada y salida, mezclarlas todas en una estructura jerárquica de programa y después ordenar detalladamente la lógica procedimental para que se ajuste a ésta estructura.

El diseño lógico debe preceder y estar separado del diseño físico

Por ejemplo las metodologías de Jackson, Warnier, Warnier-Orr.

3.1.4.2. Metodología no Estructurada

Metodologías Orientadas a Objeto

La orientación a objetos unifica procesos y datos encapsulándolos en el concepto de objetos.

La esencia del desarrollo orientado a objetos es la identificación y organización de conceptos del dominio de la aplicación y no tanto de su representación final en un lenguaje de programación.

Tiene dos enfoques distintos:

- Revolucionario, puro u ortodoxo. Rompen con las metodologías tradicionales.

Por ejemplo las metodologías OOD de Booch, CRC/RDD de Wirfs-Brock.

- Sintetista o evolutivo. Toman como base los sistemas estructurados y conforman elementos de uno y otro tipo.

Por ejemplo la metodología OMT de Rumbourgh.

Consideraciones Sobre Metodologías OO

- Se eliminan fronteras entre fases debido a la naturaleza iterativa del desarrollo orientado al objeto.
- Aparece una nueva forma de concebir los lenguajes de programación y su uso al incorporarse bibliotecas de clases y otros componentes reutilizables.
- Hay un alto grado de iteración y solapamiento, lo que lleva a una forma de trabajo muy dinámica.

Aspectos Positivos de las Metodologías OO

- Son interactivas e incrementales.
- Fácil de dividir el sistema en varios subsistemas independientes.
- Se fomenta la reutilización de componentes.

Sistemas de Tiempo Real

Procesan información orientada al control más que a los datos.

Se caracterizan por concurrencia, priorización de procesos, comunicación entre tareas y acceso simultáneo a datos comunes. [www.028]

Metodologías para Sistemas de Tiempo Real

- Manejo de interrupciones.
- Comunicación y sincronización entre tareas.
- Gestión de procesos concurrentes.
- Respuesta oportuna ante eventos externos.
- Datos continuos o discretos.
- Se está produciendo una evolución de las metodologías orientadas
- a objetos para desarrollos de sistemas de tiempo real.

3.1.5. Impacto de la Metodología en el Entorno de Desarrollo

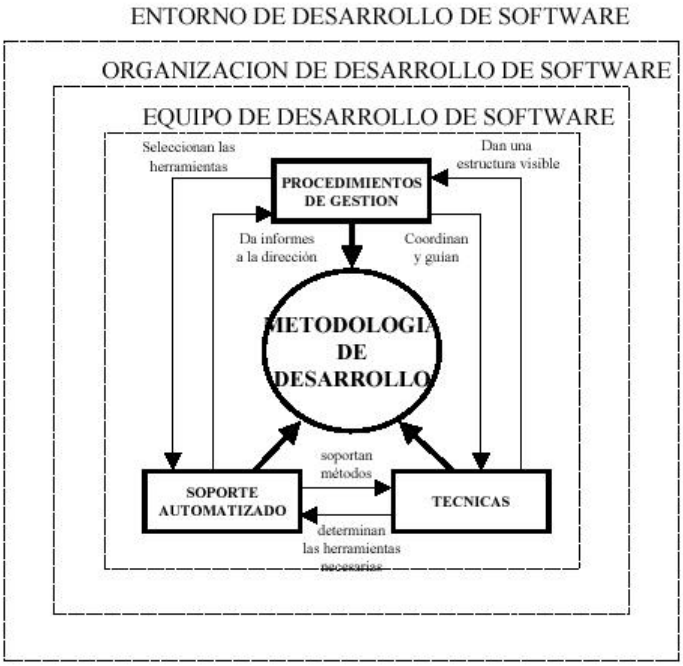


FIGURA 3.1: Entorno de Desarrollo mediante metodologías

3.1.6. Características Deseables de una Metodología

En una metodología se necesita:

- Existencia de reglas predefinidas.
- Cobertura total del ciclo de desarrollo.
- Verificaciones intermedias.
- Planificación y control.
- Comunicación efectiva.
- Utilización sobre un abanico amplio de proyectos.
- Fácil formación.

- Herramientas CASE.
- Actividades que mejoren el proceso de desarrollo.
- Soporte al mantenimiento.
- Soporte de la reutilización de software

3.1.7. Principales Metodologías de Desarrollo

a. Metodología Merise

Fases de la Metodología:

- Estudio Preliminar
- Estudio Detallado
- Implementación
- Realización y puesta en marcha

b. M

METODOLOGIA SSADM

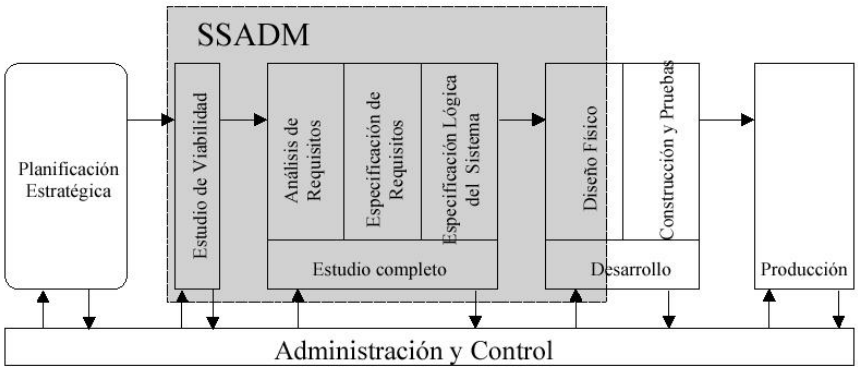


FIGURA 3.2: Metodología SSADM

c. Metodología Métrica

Consiste en las siguientes fases para el desarrollo de una aplicación web:

FASE 0: Plan de Sistemas de Información

FASE 1: Análisis de Sistemas

FASE 2: Diseño de Sistemas

FASE 3: Construcción de Sistemas

FASE 4: Implantación de Sistemas

3.2. Desarrollo Ágil de Software

Se entiende como Desarrollo ágil de software a un paradigma de Desarrollo de Software basado en procesos ágiles. Los procesos ágiles de desarrollo de software, conocidos anteriormente como metodologías livianas, intentan evitar los tortuosos y burocráticos caminos de las metodologías tradicionales enfocándose en la gente y los resultados.

Es un marco de trabajo conceptual de la ingeniería de software que promueve iteraciones en el desarrollo a lo largo de todo el ciclo de vida del proyecto. Existen muchos métodos de desarrollo ágil; la mayoría minimiza riesgos desarrollando software en cortos lapsos de tiempo.

El software desarrollado en una unidad de tiempo es llamado una iteración, la cual debe durar de una a cuatro semanas. Cada iteración del ciclo de vida incluye: planificación, análisis de requerimientos, diseño, codificación, revisión y documentación. Una iteración no debe agregar demasiada funcionalidad para justificar el lanzamiento del producto al mercado, pero la meta es tener un demo (sin errores) al final de cada iteración. Al final de cada iteración el equipo vuelve a evaluar las prioridades del proyecto.

Los métodos Agiles enfatizan las comunicaciones cara a cara en vez de de la documentación.

La mayoría de los equipos Agiles están localizados en una simple oficina abierta, a veces llamadas "plataformas de lanzamiento" (bullpen en inglés). La oficina debe incluir revisores, diseñadores de iteración, escritores de documentación y ayuda y directores de proyecto. Los métodos ágiles también enfatizan que el software funcional es la primera medida del progreso. Combinado con la preferencia por las comunicaciones cara a cara, generalmente los métodos ágiles son criticados y tratados como "indisciplinados" por la falta de documentación técnica. [www.030]

3.3. Diseño de Aplicaciones Distribuidas Orientadas a la Web

El diseño de aplicaciones web modernas involucra la división de una aplicación en múltiples capas; la interface de usuario, la capa media de objetos de negocios, y la capa de acceso a datos. Puede ser útil identificar los tipos de procesamiento que podemos esperar que una aplicación realice. Muchas aplicaciones pueden, al menos, hacer lo siguiente:

- Cálculos u otros procesos de negocios.
- Ejecución de reglas de negocios.
- Validación de datos relacionados al negocio.
- Manipulación de datos.
- Ejecución de las reglas de datos relacional.
- Interactuar con aplicaciones externas o servicios.
- Interactuar con otros usuarios.

Nosotros podemos tomar estos tipos de servicios e ingresarlos dentro de los tres grupos o capas que a continuación se resumen:

- Interface de usuario (Capa de Presentación)
    - Interactuar con otros usuarios.
    - Interactuar con aplicaciones externas o servicios.
- Procesos de negocios (Capa de Negocios)
    - Cálculos u otros procesos de negocios.
    - Ejecución de reglas de negocios.
    - Validación de datos relacionados al negocio.
- Procesos de datos (Capa de Servicios de Datos).
    - Manipulación de datos.
    - Ejecución de las reglas de datos relacional.

La división de estos procesos de aplicaciones y su distribución entre diferentes procesos cliente/servidor es conocido como Procesamiento Distribuido. Generalizando estos procesos dentro de estas tres categorías o capas es una distribución lógica y no refleja necesariamente alguna opción de diseño físico sobre computadoras, terminales u otros equipos.

Se puede desarrollar una aplicación cliente/servidor distribuida basada sobre estas tres capas de Presentación, Lógica de Negocios y Servicios de Datos y tener la aplicación entera corriendo sobre una simple computadora.

De igual manera, se puede esparcir estas tres capas a través de un gran número de diferentes computadoras sobre una red, desarrollando así una aplicación cliente/servidor de tres capas. [www.031]

a. Capa de Presentación

La capa de Presentación provee las aplicaciones web con una interface de usuario (IU). Aquí se presenta información a los usuarios y acepta entradas o respuestas del usuario para ser usadas por un sistema.

Idealmente, la IU no desarrolla ningún procesamiento de negocios o reglas de validación de negocios. Por el contrario, la IU debería relegar sobre la capa de negocios para manipular estos asuntos. Esta consideración es importante, especialmente hoy en día, debido a que es muy común para una aplicación tener múltiples IU, o para sus clientes o usuarios, que le solicitan que elimine una IU y la remplace con otra.

Una de las mayores dificultades y factores importantes cuando desarrollamos aplicaciones cliente/servidor es mantener una separación completa entre la presentación, la lógica de negocios y los servicios de datos.

b. Capa de Negocios

Toda aplicación tiene código para implementar reglas de negocios, procesos relacionados a los datos o cálculos y otras actividades relativas a los negocios. Colectivamente este código es considerado para formar la capa de negocios. Otra vez, uno de los principios del diseño lógico cliente/servidor, la lógica de negocios debe mantenerse separada de la capa de presentación y de los servicios de datos. Esto no significa necesariamente que la lógica de negocios está en cualquier parte, por el contrario, esta separación es en un sentido lógico.

Hay muchas formas de separar la lógica de negocios. En términos orientados a objetos, se debería encapsular la lógica de negocios en un conjunto de objetos o componentes que no contienen presentación o código de servicios de datos. Teniendo separada lógicamente la lógica de negocios de ambas, la capa de presentación y servicios de datos, se ganará en flexibilidad en término de donde puede almacenar físicamente la lógica de negocios. Por ejemplo, usted puede seleccionar almacenar la lógica de negocios sobre cada estación de cliente, u optar por ejecutar la lógica de negocios sobre un servidor de aplicaciones, permitiendo a todos los clientes acceder a un recurso centralizado.

Los objetos de negocios son diseñados para reflejar o representar negocios. Ellos se convierten en un modelo de sus entidades de negocios e

interrelaciones. Esto incluye tanto objetos físicos como conceptos abstractos. Estos son algunos ejemplos de objetos del mundo real: un empleado, un cliente, un producto, una orden de compra.

Todos estos son objetos en el mundo físico, y la idea en su totalidad detrás de usar objetos de negocios de software, es crear una representación de los mismos objetos dentro de la aplicación web. Sus aplicaciones web pueden hacer que estos objetos interactúen unos con otros como ellos lo hacen en el mundo real. Por ejemplo, un empleado puede crear una orden de compra a un cliente que contiene una lista de productos. Siguiendo esta lógica se puede crear objetos de negocios de una orden conteniendo el código necesario para administrarse a si mismo, así nunca se necesitará replicar código para crear ordenes, solo se usará el objeto. Similarmente, un objeto cliente contiene y administra sus propios datos. Un buen diseño de un objeto cliente contiene todos los datos y rutinas necesitadas para representarlo a través del negocio completo, y puede ser usado a través de toda la aplicación de ese negocio.

No toda la lógica de negocio es la misma. Alguna lógica de negocio es un proceso intensivo de datos, requiriendo un eficiente y rápido acceso a la base de datos. Otras no requieren un frecuente acceso a los datos, pero es de uso frecuente por una interface de usuario robusta para la validación en la entrada de campos u otras interacciones de usuarios. Si se necesita una validación al nivel de pantallas y quizás cálculos en tiempos reales u otra lógica de negocios, podríamos considerar este tipo de lógica de negocios para ser parte de la IU, ya que en su mayor parte es usada por la interface de usuario. [www.29]

Una alternativa de solución es dividir la capa de lógica de negocios en dos:

- Objetos de negocios de la IU (propiedades y métodos usados por el diseñador de interface de usuario).
- Objetos de negocios de datos (mecanismos de persistencia e interacción con bases de datos).

c. Capa de Servicios de Datos

Muchas aplicaciones interactúan con datos, los almacenan en alguna forma de bases de datos. Hay algunas funciones básicas que son comunes a todos los procesos. Estas incluyen:

- Crear datos,

- leer datos,
- actualizar datos y
- eliminar datos.

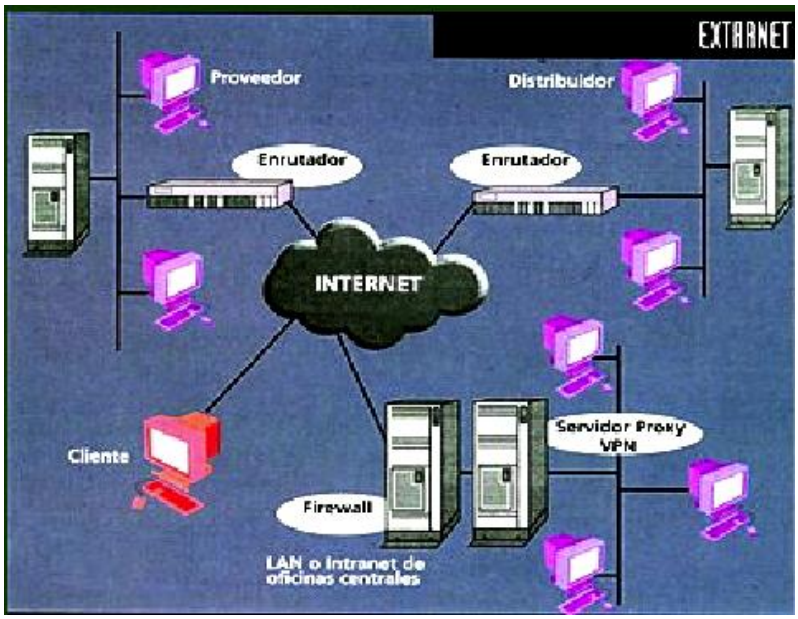
Adicionalmente, nosotros tenemos servicios más avanzados disponibles, tales como: búsquedas, ordenamientos, filtrados, etc.

La adopción de un diseño distribuido de aplicaciones empresariales, aumenta la reusabilidad, reduce la cantidad de recursos, y los costes necesarios de desarrollo y mantenimiento.

Este nuevo enfoque de diseño pone en manos de los desarrolladores no solo la funcionalidad que demandan las aplicaciones, sino también la seguridad, rapidez y flexibilidad.

CAPITULO IV

SEGURIDAD APLICADA A SERVIDORES, SITIOS Y APLICACIONES WEB



4.1.Seguridad Física y Ambiental del Equipamiento que Soporta el Sitio Web

4.2.Resguardo de la Información

4.3.Control de Cambios

4.4.Seguridad para Servidores Web

4.5.Seguridad en Páginas Web

4.6.Seguridad y Transacciones en la Web

4.7.Seguridad del Sistema Operativo y las Comunicaciones

4.8.Seguridades en la Red

4.9.Herramientas Informáticas Aplicadas a la Seguridad Web

4.10. Seguridad en Aplicaciones Web Dinámicas

4 Seguridad Aplicada a Servidores, Sitios y Aplicaciones Web

4.1. Seguridad Física y Ambiental del Equipamiento que Soporta el Sitio Web

Existen varias consideraciones a tomar en cuenta para preservar la seguridad y correcto funcionamiento de servidores y por ende los sitios web alojados en el mismo, entre los más necesarios tenemos:

- Ubicar los equipos informáticos en un ambiente con acceso restringido sólo a personas autorizadas.
- Registrar el ingreso y egreso de personas al ambiente que alberga al equipamiento (como mínimo de personas ajenas al área informática) de manera de permitir la posterior revisión de los accesos ocurridos. El administrador del Servidor tiene la capacidad de obtener reportes de los accesos tanto correctos como incorrectos, realizados por determinado usuario.
- Prohibir comer, beber y fumar dentro del ambiente donde reside el equipamiento.
- Adoptar controles adecuados para minimizar el riesgo de amenazas potenciales, por: robo o hurto, incendio, explosivos, humo, inundaciones o filtraciones de agua (o falta de suministro), polvo, vibraciones, efectos químicos, interferencia en el suministro de energía eléctrica (cortes de suministro, variación de tensión), radiación electromagnética, derrumbes, etc.
- Implementar un sistema de refrigeración que permita mantener una temperatura adecuada para el procesamiento. Prever equipamiento alternativo en caso de falla del primario.
- Revisar regularmente las condiciones ambientales para verificar que las mismas no afecten de manera adversa el funcionamiento de las instalaciones del equipamiento.
- Asegurar la continuidad del suministro de energía del equipamiento, en función a la criticidad del mismo, mediante:
    - La disposición de múltiples líneas de suministro para evitar un único punto de falla en el suministro de energía. De ser posible, asignar el equipamiento de procesamiento y comunicaciones una fase diferente a la del resto de las instalaciones.
    - El suministro de energía in interrumpible (UPS) para asegurar el apagado regulado y sistemático o la ejecución continua del equipamiento. Los planes de contingencia deberán contemplar las

acciones que han de emprenderse ante una falla de la UPS. Los equipos de UPS serán inspeccionados y probados periódicamente (a intervalos no mayores a seis meses) para asegurar que funcionan correctamente y que tengan la autonomía requerida.

- La disposición de un generador de respaldo para los casos en que el equipamiento deba continuar funcionando ante una falla prolongada en el suministro de energía. Se deberá disponer de un adecuado suministro de combustible para garantizar que el generador pueda funcionar por un período prolongado. Cuando el encendido de los generadores no sea automático, se deberá asegurar que el tiempo de funcionamiento de la UPS permita el encendido manual de los mismos. Los generadores deberán ser inspeccionados y probados periódicamente para asegurar que funcionen según lo previsto.

- Proteger el cableado de energía eléctrica y de comunicaciones que transporta datos o brinda apoyo a los servicios de información contra intercepción o daño, mediante:
  - La utilización de pisoducto o cableado embutido en la pared, siempre que sea posible.
  - La separación de los cables de energía de los cables de comunicaciones para evitar interferencias.
  - La protección del tendido del cableado troncal (“backbone”) de ser posible mediante la utilización de ductos blindados.

- Someter el equipamiento a tareas de mantenimiento preventivo, de acuerdo con los intervalos de servicio y especificaciones recomendados por el proveedor.

Cuando el sitio web se mantenga bajo la modalidad “housing” o “hosting”, deberán definirse acuerdos de nivel de servicio o cláusulas contractuales con el proveedor de manera que se contemplen las pautas de seguridad detalladas previamente. [www.032]

4.2. Resguardo de la Información

Se debe tomar en cuenta las siguientes consideraciones:

- Efectuar copias de seguridad periódicas del sitio, con un esquema adecuado de frecuencia, tipo de resguardo, rotación y reutilización de medios de almacenamiento. Las copias deben ser almacenadas en un lugar físicamente seguro, de manera de prevenir el daño, robo o pérdida de las mismas (ej.: a causa de robo, incendio, etc.).

- Probar periódicamente la correcta restauración de las copias de seguridad.
- Efectuar copias de seguridad de los logs del sistema y almacenarlas según un esquema de rotación lo suficientemente largo.
- En casos de eliminación de información o bien la reutilización de medios de almacenamiento, efectuar la eliminación de la información de forma segura, en función a la criticidad de la misma.

4.3. Control de Cambios

Se debe considerar las siguientes recomendaciones:

- Gestionar los cambios de forma de garantizar que los mismos se efectúan de forma autorizada y segura. Documentar e implementar procedimientos para la solicitud, evaluación, diseño, prueba e implementación de los cambios.
- Implementar un mecanismo de solicitud de cambios formalizado.
- Documentar e implementar un procedimiento de análisis de los cambios solicitados, incluyendo la evaluación del impacto del cambio sobre la seguridad de la información.
- Probar los cambios junto con el usuario para garantizar que cumplan con los requerimientos.
- La implementación de cambios en el sitio (ya sea de diseño o de contenido) debe encontrarse formalmente autorizada y documentada.
- Mantener una bitácora de los cambios realizados en la configuración del sistema. Esta bitácora no debe ser almacenada en el mismo servidor y debe estar debidamente resguardada.
- Documentar los procedimientos para solucionar problemas comunes y lo referidos a mantenimiento periódico.
- Mantener un entorno de desarrollo/prueba separado física y lógicamente del sitio productivo.
- Definir perfiles de acceso a los diferentes ambientes de acuerdo a las funciones de los usuarios. Restringir el acceso para agregar, modificar o eliminar información de diseño o de contenido del sitio al personal autorizado. El personal de desarrollo no debe tener acceso al entorno donde se aloja el sitio, sino únicamente al entorno de desarrollo/prueba.
- Mantener un registro y control de versiones anteriores del sitio.

4.4. Seguridad para Servidores Web

Los Servidores Web son la principal fuente de entretenimiento de los hackers y crackers debido a la vulnerabilidad que pueden poseer durante la ejecución de procesos, conexiones o fallas de configuración ya sean por un software de baja fiabilidad o el desconocimiento de su óptimo uso.

Riesgos

Hoy en día las conexiones a servidores web son sin duda las más extendidas entre usuarios de Internet, hasta el punto de que muchas personas piensan que este servicio (http, puerto 80 tcp) es el único que existe en la red.

Lo que en un principio se diseño para que unos cuantos físicos intercambiaran y consultaran artículos fácilmente, en la actualidad mueve a diario millones de dólares y es uno de los pilares fundamentales de cualquier empresa: es por tanto un objetivo muy atractivo para cualquier pirata.

4.4.1. Medidas de Seguridad

La seguridad de un servidor web debe estar diseñada en capas, cada capa súper pone a la anterior. [www.033]

El esquema de seguridad debe constar de 4 tipos de medidas de seguridad básica:

- Preventivas
- Reactivas
- Detección
- Recuperación

a. Medidas Preventivas

Su objetivo es prevenir o dificultar la intrusión. Por ejemplo: Firewalls.

b. Medidas Reactivas

Reaccionan ante la amenaza tomando una contra medidas ejemplo: Firewall de aplicaciones, Mod_security, Sistemas de detección de ataques de fuerza bruta (antivirus), entre otros.

c. Medidas de Detección

Ayudan a detectar si una intrusión a ocurrido o está en proceso ejemplo, HIDS como Tripwire, chkrootkit, antivirus, scaners, etc.

d. Medidas de Recuperación

Ayudan a recuperar la operatividad luego de una intrusión ejemplo: backups e imágenes de disco.

Al implementar un esquema de seguridad se deben planear líneas defensivas, cada línea refuerza la anterior, de modo que el fallo o la vulnerabilidad de una línea no comprometan la instalación.

4.5. Seguridad en Páginas Web

Teniendo en cuenta algunas consideraciones:

- Que la página Web es puesta on-line en Internet o una Intranet por un proveedor del servicio de conexión a Internet ó un administrador de una determinada LAN.
- La actualización y modificación del servicio corren por cuenta del dueño por medio de un acceso al sitio del proveedor protegido por algún tipo de sistema de seguridad (por ej.: un sistema de passwords)

El administrador de un Servidor Web (prestante de servicio de alojo) y el creador de Web Sites deben tener en consideración los siguientes aspectos:

4.5.1. Operatividad

La información que contenga la página debe estar siempre en condiciones operativas para quienes acceden a la misma puedan recorrerla sin problemas, sin encontrar fallas, faltas o cualquier tipo de anomalías.

Responsabilidad del encargado de la página: la información agregada o modificada por un responsable de la página debe ser ingresada en ella en los formatos establecidos y verificado su correcto funcionamiento dentro de la estructura de programación.

Responsabilidad del prestador del servicio: todo sistema de computadoras está expuesto a fallas de hardware, software y de tipo externo como fallas de suministro eléctrico. Para ello deberá tener un plan de soluciones y un mantenimiento preventivo. Además debe garantizar que si una información es colocada en la página según las especificaciones y procedimientos acordados, ésta funcionará correctamente.

Se debe considerar que el ingreso de un intruso al sistema puede provocar daños que afecten

La operatividad.
El prestador del servicio deberá tener un sistema de seguridad, por ejemplo, en barreras de protección como firewalls, proxis, etc., que impidan los accesos no autorizados. [www.034]

4.5.2. Integridad

De nada sirve que una información esté en condiciones operativas si es incompleta o está alterada.

Para que una información resulte inútil no es necesario que sea destruida, puede ser suficiente una acción tan sutil como cambiar los unos por ceros.

La integridad de la información que se muestra en una página Web es uno de los factores más importantes de la seguridad, pues de él dependen el interés y la credibilidad de la página.

La integridad de la página puede ser dañada por fallas de hardware o software, o atacada por intrusos en el sistema que modifican el contenido de las páginas.

Responsabilidades del propietario de la página: la información que es agregada o modificada en la página debe estar en condiciones de integridad cuando llega a ella, y tratando de que se mantenga hasta que termine, pues puede ser afectada por la transmisión hasta el sitio o por algún problema de su funcionamiento o seguridad.

Responsabilidades del prestador del servicio: asegurar la integridad de la información que contiene una página Web, en lo que atañe a accesos no autorizados al sistema. Los bugs (errores estructurales) de los programas utilizados también pueden ser la puerta de entrada para los accesos no autorizados. Se debe exigir que el sistema del proveedor esté depurado de este tipo de fallas. [www.32]

4.5.3. Privacidad

Es lógico pensar que quien quiere que una información sea privada no debe colocarla en una página Web. Pero puede ocurrir que parte de la información esté reservada a usuarios registrados o que exista algún tipo de restricción.

Responsabilidades del dueño de la página: el dueño de la página debe definir y separar claramente cuál es la información de dominio público y cual de acceso restringido, y manejarlas en zonas separadas en la programación de su página.

Responsabilidades del prestador del servicio: el acceso restringido a parte de la información de una página debe ser sustentado por el prestador del servicio asegurando que los mecanismos de control de acceso de la página funcionen correctamente en su sistema, tanto en seguridad del servidor web para control de sitios como del acceso al servidor en su estado físico. [www.034]

4.6. Seguridad y Transacciones en la Web

En la actualidad las aplicaciones web tienen a su cargo la administración de información realmente importante y delicada, la cual, no puede ser sorprendida y robada, debido a su grado de importancia y confidencialidad.
Por ello la seguridad en la transacciones de datos e información en la red debe ser lo más segura posible.
Algunas de las medidas más importantes de seguridad básicas a tener en cuenta son:

- Encriptación de Datos
- Firmas Digitales

4.6.1. Encriptación de Datos

Es una técnica para ocultar datos de manera que sólo puedan ser vistos por aquellos que deben verlos. Consiste en reemplazar un mensaje enviado con un algoritmo difícil de adivinar y descifrar.
Los servidores seguros tratan de encriptar los datos entre el navegador y el servidor.
En algún momento durante el ciclo de compras, después que los datos llegan al servidor seguro, el sistema debe desencriptar los datos. Aun si los datos son desencriptados sólo por un instante, la información podría ser interceptada por algún pirata. Crear un sistema en el que la información permanezca encriptada a lo largo del ciclo es prácticamente imposible.

La configuración más segura es una que transmita la información al propietario de la empresa en formato encriptado, pase la información a una computadora que no esté en internet y luego desencripte la información.

Además si en una empresa se utiliza un mismo algoritmo para encriptar y desencripar datos, se necesitará que alguna tercera pieza de datos desencripte el código, que seria una clave. Esto sólo funcionará si tanto la persona transmisora

como la parte receptora conocen la clave. Si la persona receptora no conoce la clave, tiene que enviar la clave a esa parte, y está puede ser interceptada. Para evitar este problema se tiene que buscar un medio diferente de transmitir la clave para su uso posterior. [www.034]

4.6.2. Firmas Digitales

Ofrece un método de encriptación de datos que evita tener que compartir claves para leer mensajes.

Es la técnica llamada encriptación de clave pública, donde cada usuario tiene dos claves: una clave pública y una clave privada.

Los algoritmos de encriptación y desencriptación son adaptados de manera que sólo la clave pública puede desencriptar los datos encriptados por la clave privada. Por consiguiente, puede transmitir con libertad la clave pública al mundo.

4.7. Seguridad del Sistema Operativo y las Comunicaciones

Se deben considerar los siguientes aspectos:

- Mantener el sistema operativo actualizado, con los últimos parches de seguridad disponibles. Las actualizaciones críticas deben ser probadas antes de ser implementadas en el ambiente de producción de manera de asegurarse que los cambios no afecten la operatoria del sitio web.
  Configurar el sistema operativo de manera segura, implementando las mejores prácticas para el hardening de plataformas. Esto incluye entre otras actividades, deshabilitar cuentas de ejemplo, cambiar claves por defecto, etc. [www.032]

- Administrar adecuadamente las cuentas de usuario procurando:
  - Otorgar acceso al sistema operativo sólo a personal autorizado, con privilegios mínimos que le permitan cumplir con sus funciones.
  - Establecer una política de uso de las cuentas críticas (administrador de plataformas, etc.), así como procedimientos detallados.
  - Controlar la vigencia de los accesos otorgados a los usuarios respecto del sistema operativo.
  - Asegurar la posibilidad de identificar unívocamente a todos los usuarios del sistema a través de sus cuentas.

- Implementar políticas de contraseñas fuertes y de control de acceso a toda la información almacenada en el servidor de forma restrictiva.
- Registrar las actividades críticas ejecutadas en el sistema operativo a través de los logs propios de la plataforma, de manera de permitir el rastreo de información relacionada con un posible incidente de seguridad.
- De ser posible, almacenar los logs en un servidor distinto a donde se generan.
- Mientras sea posible, utilizar los paquetes oficiales (cuando se trate de una distribución de Linux o BSD) y evitar compilar programas a mano o instalar paquetes desde fuentes no oficiales.
- Implementar y mantener actualizado un software antivirus. Efectuar controles preventivos periódicos.
- Desinstalar aquellos servicios que no sean utilizados en el servidor.
- No utilizar servicios que puedan ser utilizados por atacantes para recabar información del sistema operativo. (ej.: finger, rstat, rusers).
- Utilizar conexiones, siempre que se requieran, que permitan el cifrado de la información y la autenticación de las partes para la administración remota de equipos, la transferencia de archivos y el intercambio de información (ej.: SSH, SFTP, SCP, SSL).
- Ubicar el/los servidores donde se encuentra alojado el sitio web de internet en un segmento de la red separado de la red interna del Organismo (ej.: DMZ).
- Proteger con un firewall externo la red donde se ubica el servidor, además de un firewall funcionando en el mismo equipo.

4.8. Seguridades en la Red

Las fallas de seguridad en la red son un gran problema cuando la calidad de comunicación e integridad de información se ven en peligro.

Existen dos tipos de ítems a tomar muy en cuenta en cuestión de seguridad de la red:

- Seguridad de la infraestructura de la red.
- Seguridad del contenido.

Seguridad en la infraestructura de la red incluye la protección física de los dispositivos que proporcionan conectividad de red y evitan el acceso no autorizado al software de administración existente en los mismos.

Seguridad del contenido se refiere a la protección de la información contenida en los paquetes transmitidos por la red y la información contenida en los dispositivos conectados a ésta.

Se deben implementar herramientas para proporcionar seguridad al contenido de los mensajes individuales sobre los protocolos subyacentes que rigen la forma en que los paquetes se formatean, direccionan y envían.

Debido a que el re ensamblaje y la interpretación del contenido se delega a programas que se ejecutan en sistemas individuales de origen y destino, muchos de los protocolos y herramientas de seguridad deben implementarse también en esos sistemas.

Las medidas de seguridad que se deben tomar en una red son:

- Impedir la divulgación no autorizada o el robo de información,
- Impedir la modificación no autorizada de información, y
- Impedir la denegación de servicio.

Los medios para lograr estos objetivos incluyen:

- Garantizar la confidencialidad.
- Mantener la integridad de la comunicación.
- Garantizar la disponibilidad.

a. Confidencialidad

Solamente los receptores autorizados y designados (usuarios, dispositivos, etc.), deben tener la capacidad de leer la información circulante.

La restricción de acceso a la información se la logra únicamente mediante autenticación de cuentas, contraseñas seguras y encriptación de la misma. Tan solo de ésta manera se asegura la confidencialidad y reduce la divulgación ó robo de datos o información utilizada en la red.

b. Integridad de las Comunicaciones

La integridad de datos significa que la información no se alteró durante la transmisión de origen a destino.

La integridad de datos se ve afectada cuando se daña la información, ya sea en forma intencional o accidental, antes de que el receptor correspondiente la reciba.

La integridad de origen es la confirmación de que se validó la identidad del emisor. Se compromete la integridad del origen cuando un usuario o dispositivo

falsifica su identidad y proporciona información incorrecta al destinatario al enviar información alterada.

El uso de firmas digitales, algoritmos de hash y mecanismos de checksum son formas de proporcionar integridad de origen y de datos a través de la red para evitar la modificación no autorizada de información.

c. Disponibilidad

Tener Disponibilidad de información o datos en una red quiere decir el acceso (con autorización) a los mismos de una manera confiable y oportuna.
Los dispositivos firewall de red, junto con los software antivirus de los equipos de escritorio y de los servidores pueden asegurar la confiabilidad y solidez del sistema para detectar, repeler y resolver esos ataques.

4.9. Herramientas Informáticas Aplicadas a la Seguridad Web

Además de las herramientas propias de los administradores web proporcionados por los Servidores Web y Sistemas Operativos, existen determinados programas que pueden ser de gran utilidad en el momento de administrar seguridades en el trabajo sobre una red.
Alguna de estas herramientas son los firewalls, wrappers y proxies, que ofrecen una buena línea de defensa para los propietarios de servidores web y administradores de sistemas.

Los firewalls pueden ser software o hardware que protege los puertos y evita que los piratas penetren al sistema. Los firewalls permiten que tengan acceso al sistema sólo ciertos nombres de dominio confiables.

Los wrappers se encuentran disponibles en CERT al igual que en otros archivos en Internet. Los wrappers se ejecutan como una capa de software alrededor de su otro software. Un usuario que se conecta a FTP primero entraría en contacto con el wrapper, el cual luego habilitaría al FTP. El usuario no sabe que existe el wrapper y no puede detectar ninguna diferencia en el sistema.
Los wrappers son interesantes porque son flexibles. Pueden actuar como firewalls y en realidad pueden rechazar usuarios con base en sus nombres de usuarios al igual que en sus nombres de dominios. Además permite crear callejones sin salida que permiten atrapar piratas.

El proxy es un método permite ocultar datos por medio de re-enrutamiento de las solicitudes. Es útil para usuarios que están detrás de una firewall. Los usuarios establecen una dirección proxy de su navegador para que apunte hacia su servidor Web. El servidor Web maneja entonces la dirección real de los datos hacia el mundo exterior. Esto reduce la dirección que el usuario está tomando cuando deja su sistema, permitiéndole al usuario enrutar los datos a través de los agujeros en sus propias firewalls.
Otra de las ventajas es que las solicitudes pueden ser filtradas por el software del servidor. Al filtrar la información, puede restringir el contenido y rastrear el uso al igual que modificar la información en ese instante.

Los servidores proxy también pueden ser dirigidos a otros servidores proxy, lo cual les permite ocultar datos en forma efectiva.

Otra ventaja de los servidores proxy es que los servicios como FTP, Telnet, Gopher, NetnNews, etc., pueden ser enrutados a servidores diferentes. Esto le permite distribuir diversas cargas de servidor Web a diferentes servidores físicos. Además de beneficiarse con el ocultamiento de los datos, ser reduce la carga del servidor. [www.034]

4.10. Seguridad en Aplicaciones Web Dinámicas

4.10.1. Principios Básicos de Seguridad

a. Validación de la entrada y salida de información

   La entrada y salida de datos es el principal mecanismo que dispone un atacante para enviar o recibir código malicioso contra el sistema. Por tanto, siempre debe verificarse que cualquier dato entrante o saliente es apropiado y en el formato que se espera. Las características de los datos deben estar predefinidas y verificarse en todas las ocasiones.

   Las aplicaciones Web interpretan las solicitudes HTTP para determinar cómo responder a los pedidos de los usuarios. Se debe validar todo parámetro de entrada, incluyendo URLs, la cadena de consulta (query string), los encabezados, las “cookies” y/o los campos de formularios (normales o escondidos) teniendo en cuenta:

- Tipo de datos (string, integer, real, etc.).
- Conjunto de caracteres permitidos.
- Longitud mínima y máxima.
- Si el valor nulo es permitido.
- Si el parámetro es requerido o no.
- Si los duplicados son permitidos.
- El rango numérico.
- Valores específicos permitidos (enumeración).
- Patrones específicos (expresiones regulares).
- Juego de caracteres permitidos.

Estas validaciones son importantes ya que de no existir, los atacantes podrían manipular cualquier parte de la solicitud HTTP, para tratar de evitar los mecanismos de seguridad del sitio y/o cambiar la lógica de ejecución de las aplicaciones logrando resultados no deseados.

b. Diseños simples

Los mecanismos de seguridad deben diseñarse para ser lo más sencillo posible, huyendo de sofisticaciones que compliquen excesivamente la vida a los usuarios. Si los pasos necesarios para proteger de forma adecuada una función o modulo son muy complejos, la probabilidad de que estos pasos no se ejecuten correctamente es muy elevada. [www.035]

Para el diseño, se debe considerar:

- Incorporar la seguridad en el diseño de los sistemas, definiendo, implementando y probando los controles necesarios.
- Otorgar acceso a las aplicaciones web sólo a personal autorizado, con privilegios mínimos que le permitan cumplir con sus funciones.
- Establecer una política de uso de las cuentas críticas (administrador de sistema, administrador de base de datos, etc.), así como procedimientos detallados.
- En caso de que la aplicación web utilice un motor de bases de datos, otorgar a los usuarios, según su perfil, sólo los permisos indispensables para el correcto funcionamiento de la aplicación.

- Implementar medidas de seguridad en las bases de datos de manera de garantizar la integridad de la información y evitar amenazas como ser la inferencia de información o la agregación.
- Implementar políticas de contraseñas fuertes y de control de acceso a las aplicaciones web.
- Controlar la vigencia de los accesos otorgados a los usuarios a cada aplicación web.
- Asegurar la posibilidad de identificar unívocamente a todos los usuarios del sistema a través de sus cuentas.
- Registrar las actividades críticas ejecutadas en las aplicaciones web, de manera de permitir el rastreo de información relacionada con un posible incidente de seguridad.

c. Utilización y reutilización de componentes de confianza

Cuando exista un componente que resuelva un problema de forma correcta y eficaz, lo más recomendable es utilizarlo y no tratar de crear algo existente.

d. Defensa en profundidad

No se debe confiar en que un componente realizará su función de forma permanente y ante cualquier situación. Debemos disponer de los mecanismos de seguridad suficientes para que cuando un componente del sistema falle ante un determinado evento, otros sean capaces de detectarlo.

e. Tan seguros como el eslabón más débil

Nunca debemos fiarnos de los mecanismos de seguridad "exteriores", sino que es preciso identificar cuales son los puntos precisos en los que deben establecerse las medidas de seguridad en nuestras aplicaciones.

Entre más fuerte sea la estructura de seguridad de nuestro sistema, mejor será la seguridad que la preceda.

f. La "seguridad gracias al desconocimiento" no funciona

El hecho de ocultar algo no impide que, a medio o largo plazo, llegue a ser descubierto. Tampoco es ninguna garantía de que tampoco será descubierto a corto plazo. Cualquier vulnerabilidad debe ser corregida a tiempo y no dejar nada a la suerte pues puede ser la mella que degrade la seguridad.

g. Verificación de privilegios.

Los sistemas deben diseñarse para que funcionen con los menos privilegios posibles. Igualmente, es importante que los procesos únicamente dispongan

de los privilegios necesarios para desarrollar su función, de manera que queden compartimentados.

h. Ofrecer la mínima información

Ante una situación de error o una validación negativa, los mecanismos de seguridad deben diseñarse para que faciliten la mínima información posible. De la misma forma, estos mecanismos deben estar diseñados para que una vez denegada una operación, cualquier operación posterior sea igualmente denegada.

i. Configuración y Mantenimiento Adecuados

Para un correcto mantenimiento de aplicaciones debemos:

- Mantener actualizadas las aplicaciones web desarrolladas por terceros, con los últimos parches de seguridad disponibles. Las actualizaciones críticas deben ser probadas antes de ser implementadas en el ambiente de producción de manera de asegurarse que los cambios no afecten la operatoria de la aplicación.
- Configurar la aplicación de manera segura, teniendo en cuenta las características particulares del tipo y versión. Esto incluye entre otras actividades, deshabilitar cuentas de ejemplo, cambiar claves por defecto, etc.

j. Manejo Adecuado de Errores

Para un correcto manejo de errores, se deben presentar al usuario mensajes de error simples y al mismo tiempo se debe registrar el error con mayor detalle para uso interno. De esta forma, la causa del error, ya sea una falla en el sitio o un intento de ataque, podrá ser revisada y analizada posteriormente.

El manejo de errores debe cubrir no sólo los datos de entrada provistos por el usuario, sino también todo error que pueda ser generado por componentes internos tales como llamadas al sistema, consultas a bases de datos o cualquier otra función interna.

El manejo inadecuado de errores puede introducir diversos problemas de seguridad en un sitio Web. Por ejemplo, la exhibición de información detallada de mensajes de error, como el contenido de variables, nombres de directorios e información sobre la base de datos, puede revelar detalles de la

implementación que no deben ser expuestos bajo ninguna circunstancia a usuarios no autorizados.

k. Almacenamiento Seguro

Toda información sensible (por ejemplo, contraseñas, números de tarjetas de crédito, registros contables, etc.) almacenada por las aplicaciones Web, ya sea en una base de datos, en un sistema de archivos o en algún otro dispositivo debe encontrarse cifrada.

En el caso de almacenar contraseñas, se debe utilizar funciones de una sola vía (por ejemplo SHA-1) y además utilizar una semilla o salt aleatoria para que dos claves iguales se almacenen de manera distinta.

Se deberá minimizar el uso del cifrado con el objeto de disminuir el riesgo de fallas criptográficas. Por ejemplo, en lugar de cifrar los números de tarjetas de crédito y guardarlos, simplemente se puede solicitar a los usuarios que los ingresen nuevamente. Si la criptografía debe ser usada, es conveniente:

- Tener Adecuados Mecanismos de Autenticación y Autorización: Se deben implementar controles de acceso en las aplicaciones Web, para evitar la ejecución de funciones por parte de usuarios no autorizados o usuarios con niveles de permisos insuficientes. Dichos controles consisten en:

  Autenticación: determinan los usuarios que pueden acceder a la aplicación.

  Autorización: determinan lo que pueden hacer los usuarios "autorizados".

  Las aplicaciones Web deben establecer sesiones para mantener el rastro del flujo de acciones de cada usuario. Para el establecimiento y mantenimiento de dichas sesiones se deben implementar mecanismos que garanticen la protección de las credenciales en tránsito y del identificador de sesión.

  Aquellos formularios susceptibles de abusos por parte de scripts o robots (sistemas de encuestas, registro de cuentas de correo, comentarios de foros, etc) deben utilizar un captcha que asegure que se trata de un humano quien realiza la operación. [www.032]

- Adecuados Mecanismos de No Repudio:

En función a la criticidad de las transacciones efectuadas en las aplicaciones Web, se debe evaluar la necesidad de implementar mecanismos para garantizar el No Repudio por parte de los usuarios.

CAPITULO V

DESARROLLO DEL APLICATIVO WEB

5.1.Metodología

5.2.Mapa del Sitio Web

5.3.Características del Aplicativo Web

5.4.Implementación

5.5.Arquitectura

5. DESARROLLO DEL APLICATIVO WEB

Con el uso de ASP.NET nosotros podemos tener un modelo de programación más unificado, con orientación a objetos, mejoras de seguridad en código e implementación y una nueva forma de escribir potentes aplicaciones web completas.

ASP.NET ofrece código más fácil de escribir, reutilizar y compartir, mejora el rendimiento y escalabilidad ofreciendo acceso a lenguajes compilados, el desarrollo de la capa de presentación y capa de negocios es más intuitivo gracias a los Formularios Web, y su base de desarrollo orientada a objetos facilita la reutilización de todos y cada uno de sus componentes, tiene soporte de eventos, controles y funciones de caché para la optimización de recursos al momento de ejecución de aplicaciones.

La gratificante facilidad de distribución e instalación de los aplicativos ASP.NET se ven respaldados por la capacidad de actualización de sitios web enteros sin la necesidad de parar el servidor y obstruir el desempeño de ejecución del mismo en la red.

Los aplicativos ASP.NET corren sobre el servidor web de IIS (internet information services), el cual goza de diversas aplicaciones de seguridad para el uso de información en la red como son encriptación de datos, administración de logins, protección a nivel de servidor, entre otros ofrecidos por el Sistema Operativo y el entorno de seguridad del mismo.

El fácil acoplamiento, compatibilidad, seguridad y alto rendimiento que se produce al trabajar con soluciones desarrolladas en ASP.NET y SQL Server 2005 como servidor de base de datos, nos brindan un acogedor ambiente de trabajo compartido.

Las ventajas ofrecidas por ASP.NET hacen de esta tecnología una de las más tentadoras para el desarrollo de los aplicativos web; por ello y todo el soporte brindado mencionado anteriormente se ha escogido ASP.NET como la mejor opción de desarrollo e implementación de aplicaciones web.

La Factibilidad de diseño, desarrollo de aplicaciones web e implementación de las mismas usando tecnología brindada por Microsoft no es un problema desde el punto de vista económico ya que la Universidad Técnica del Norte, cuenta con un acuerdo

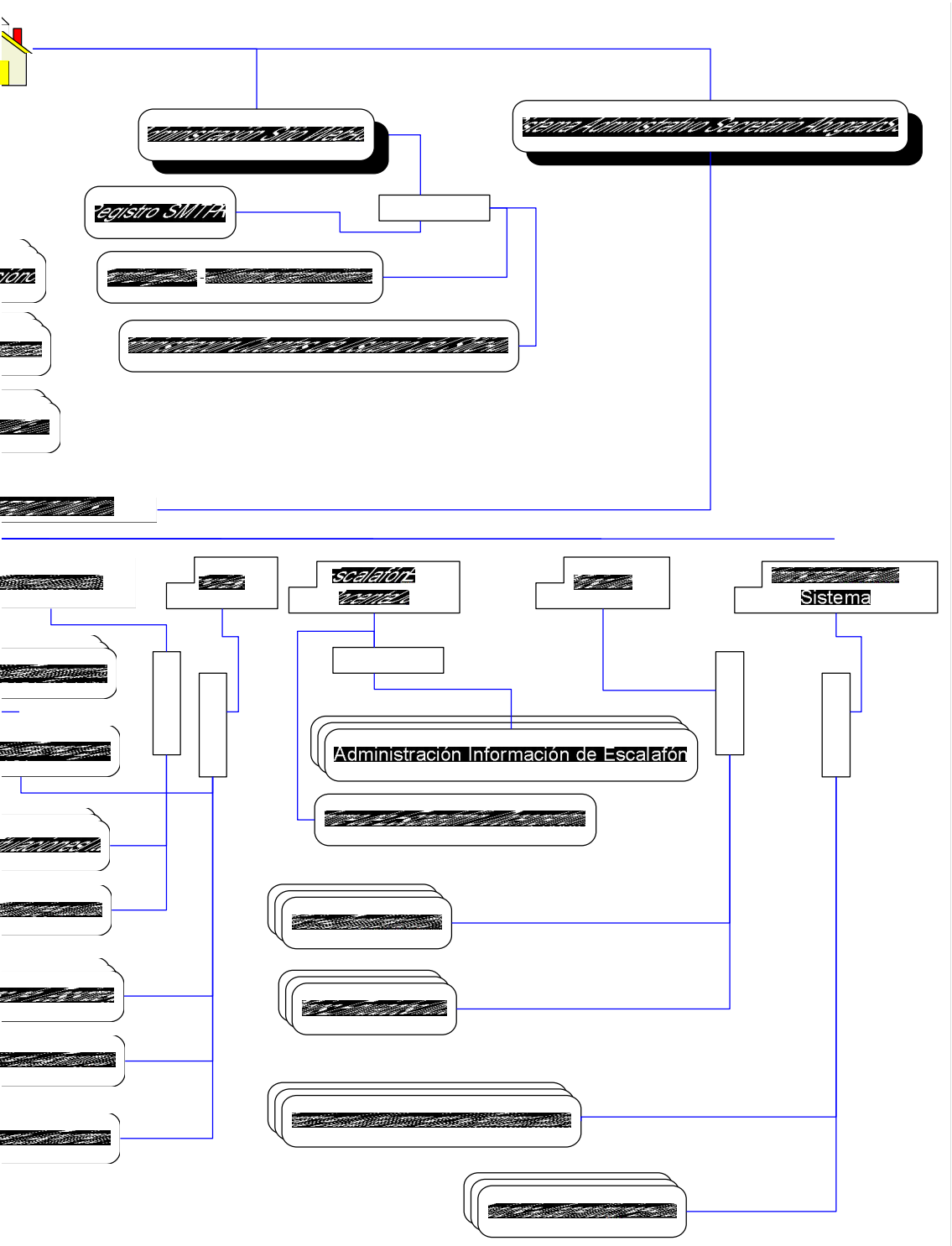
con Microsoft, en el cual se permite el uso gratuito de software de su propiedad para el desarrollo académico.

En vista de las múltiples ventajas mencionadas anteriormente incluido en ellas la factibilidad de uso gratuito de software Microsoft, se ha elegido el desarrollo del Sistema Administrativo Departamental Secretario Abogado FICA, usando ASP.NET como lenguaje de desarrollo, SQL Server 2005 como servidor de base de datos e Internet Information Services como servidor web.

5.1. Metodología

La metodología usada en la planificación, estudio de viabilidad y requerimientos, desarrollo implementación y producción de la aplicación web es la de SSADM descrita en el CAPITULO 3.

La planeación estratégica, estudios de viabilidad, estudio de requerimientos, diseño, desarrollo y producción del Aplicativo Web se puede encontrar en la sección de entregables. [Ver Anexos]



ecretario – Abogado

5.3. Características del Aplicativo Web

Después de haberse desarrollado un estudio preliminar para el diseño, desarrollo e implementación del Sistema Administrativo Departamental Secretario – Abogado se ha considerado la necesidad de implementar los siguientes módulos en el Sistema con la finalidad de cubrir los requisitos solicitados:

a. Módulo Web Site Informativo para Egresados y Titulados de la FICA.
b. Módulo Administrativo para el Web Site Informativo.
c. Módulo Administración de Información Académica.
d. Módulo Control de Asistencia Docente FICA.
e. Módulo Control de Egresos FICA.
f. Módulo Control de Titulaciones FICA.
g. Módulo Control Informativo de Tesis.
h. Módulo Administración de Información de Escalafón Docente FICA.
i. Módulo Administración de Oficios Envío – Recepción.
j. Módulo Administrativo del Sistema Administrativo Departamental Secretario Abogado FICA.

La misión del Sistema Administrativo Departamental Secretario Abogado FICA es la automatización de las actividades departamentales y procesos involucrados en cada una de las mismas.

Las actividades tomadas en consideración se detallan en cada uno de los módulos del Sistema.

a. Características del Módulo Web Site Informativo para Egresados y Titulados de la FICA

- Registro de Usuarios del Sitio.
- Actualización Personal de Información de Registro de Usuario del Sitio.
- Acceso a Sesiones de Usuario Registrados del Sitio.
- Reporte de Leyes y Reglamentos de la Universidad Técnica del Norte.
- Reporte de Tesis Aprobadas en la FICA.
- Reporte de Requisitos para Egresamiento y Titulación.
- Reporte de Trámites de Egreso y Titulación.
- Reporte de Información de Registro de Usuarios del Sitio.

b. Características del Módulo Administrativo para el Web Site Informativo

- Registro del Servidor de Mail SMTP.

- Registro de Mail de Administrador del Web Site.
- Eliminación de Información de Registro de Usuarios del Sitio.
- Reporte de Egresados Registrados en el Sitio.
- Reporte de Titulados Registrados en el Sitio.

c. Características del Módulo Administración de Información Académica

- Ingreso, Eliminación y Reporte de Materias dictadas en la FICA.
- Ingreso, Eliminación y Reporte de Escuelas existentes en la FICA.
- Ingreso, Eliminación y Reporte de Carreras Existentes en la FICA.
- Ingreso, Modificación, Eliminación y Reporte de Responsables de Carreras FICA.
- Ingreso, Eliminación y Reporte de Tipos de Docentes.
- Ingreso, Modificación, Eliminación y Reporte de Docentes.
- Ingreso, Modificación, Eliminación y Reporte de Carga Horaria.
- Ingreso, Modificación, Eliminación y Reporte de Carga Horaria Extra.
- Ingreso, Eliminación y Reporte de Docente – Carrera.
- Ingreso, Modificación, Eliminación y Reporte de Materia Docente.
- Ingreso, Modificación, Eliminación y Reporte de Horario Docente.
- Ingreso, Eliminación y Reporte de Requisitos de Egresamiento.
- Ingreso, Eliminación y Reporte Requisitos de Titulación.
- Ingreso, Eliminación y Reporte de Materias FICA desde el Sistema Académico Universitario hacia el Sistema Administrativo Secretario – Abogado.
- Ingreso, Eliminación y Reporte de Tipo - Docente FICA desde el Sistema Académico Universitario hacia el Sistema Administrativo Secretario – Abogado.
- Ingreso, Eliminación y Reporte de Docentes FICA desde el Sistema Académico Universitario hacia el Sistema Administrativo Secretario – Abogado.
- Ingreso, Eliminación y Reporte de Docentes - Materia FICA desde el Sistema Académico Universitario hacia el Sistema Administrativo Secretario – Abogado.
- Ingreso, Eliminación y Reporte de Carga Horaria de los Docentes FICA desde el Sistema Académico Universitario hacia el Sistema Administrativo Secretario – Abogado.

- Ingreso, Eliminación y Reporte de Escuelas FICA desde el Sistema Académico Universitario hacia el Sistema Administrativo Secretario – Abogado.
- Ingreso, Eliminación y Reporte de Carreras FICA desde el Sistema Académico Universitario hacia el Sistema Administrativo Secretario – Abogado.
- Ingreso, Eliminación y Reporte de Docente Carreras FICA desde el Sistema Académico Universitario hacia el Sistema Administrativo Secretario – Abogado.

d. Características del Módulo Control de Asistencia Docente FICA

- Registro, Modificación y Eliminación de Asistencia Docente Diaria.
- Cierre de Edición de Informes de Asistencia Docentes FICA.
- Generación de Informe Mensual de Asistencia de Docentes de Planta y Contrato y Horas Extras.
- Generación de Desglose de Asistencia Docente Mensual.
- Generación de Desglose de Recuperación de Horas Clase Mensual.

e. Características del Módulo Control de Egresos FICA

- Ingreso, Modificación y Eliminación de Egresados.
- Ingreso, Modificación, Eliminación y Reporte de Requisitos Faltantes por Egresado.
- Reporte de Egresados con Filtros por Facultad, Escuela, Cédula, Por Períodos y Escuela, Por Períodos y Facultad, Por Requisitos y Escuela, Por Requisitos y Facultad.

f. Características del Módulo Control de Titulaciones FICA

- Ingreso, Modificación y Eliminación de Titulados.
- Ingreso, Modificación, Eliminación y Reporte de Requisitos Faltantes por Titulado.
- Reporte de Titulados con Filtros por Facultad, Escuela, Cédula, Por Períodos y Escuela, Por Períodos y Facultad, Por Requisitos y Escuela, Por Requisitos y Facultad.

g. Características del Módulo Control Informativo de Tesis

- Ingreso, Modificación, Eliminación y Reporte de Información de Tesis.
- Subir al Servidor Información Ejecutiva de Tesis.
- Vista de Información Ejecutiva de Tesis.

h. Características del Módulo Administración de Información de Escalafón Docente FICA

- Ingreso, Modificación, Eliminación y Reporte de Información de Escalafón Docente.

i. Características del Módulo Administración de Oficios Envío – Recepción

- Ingreso, Modificación, Eliminación y Reporte de Oficios Resumidos.
- Subir Oficios Completos al Servidor.
- Visualizar Oficios Completos existentes en el Servidor.

j. Características del Módulo Administrativo del Sistema Administrativo Departamental Secretario Abogado FICA

- Ingreso, Modificación, Eliminación y Reporte de Períodos.
- Habilitar e Inhabilitar Períodos Académicos.
- Ingreso, Modificación, Eliminación y Reporte de Usuarios del Sistema Académico.

5.4. Implementación

5.4.1. Herramientas Informáticas utilizadas en la implementación del Aplicativo Web

- Microsoft Visual Studio 2005 como Herramienta de Desarrollo.
- ASP.NET como Tecnología para la Codificación.
- Power Designer como Herramienta de Diseño de la Base de Datos.
- Microsoft SQL Server 2005 como Servidor de Base de Datos.
- SQL Server Management Studio como Herramienta de Administración de la Base de Datos.
- Macromedia Fireworks para remodelación de Interfaz de Usuario.
- Internet Information Service como Servidor Web.
- Windows Server

El Aplicativo Web ha sido dividido en dos subdirectorios para el uso distribuido de los mismos entre los usuarios correspondientes, “Usuarios de Sitio” y “Usuarios de sistema”.

Para la publicación del Aplicativo Web y durante el período de pruebas del mismo, se lo realizará en el Servidor Web con las siguientes direcciones URLs:

Para el Sistema Administrativo Departamental Secretario Abogado:

http://172.20.16.13/ Lawyer/Abogado.aspx

Para el Sitio Web Informativo Dedicado a Egresados y Titulados:

http://172.20.20.13/ Lawyer/index.aspx

5.5. Arquitectura

5.5.1. Arquitectura del Sistema Administrativo Departamental Secretario Abogado

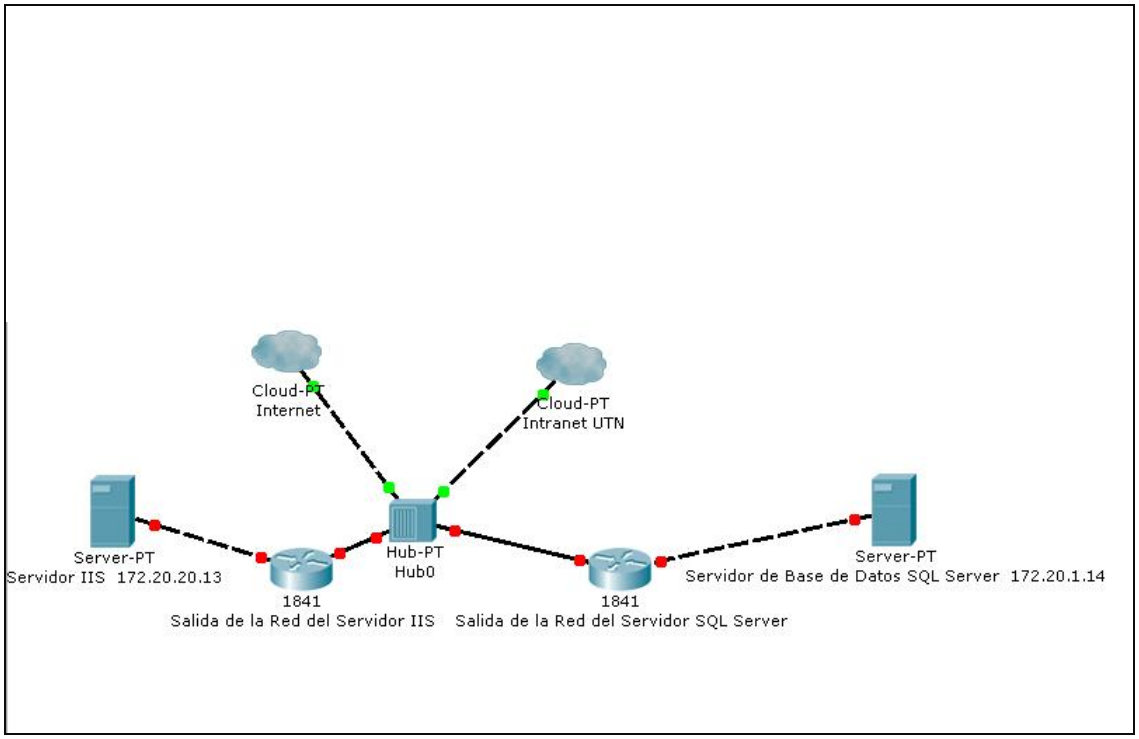


Figura 5.1 Funcionamiento de la Aplicación Web

Como podemos observar en la figura 5.1, la Ejecución de la Aplicación Web se da gracias a un Servidor Web, el mismo que interactúa con un Servidor de Base de Datos SQL Server 2005 a través de una intranet o internet, mostrándose en los navegadores con acceso a la intranet o internet.

5.5.2. Casos de Uso del Sistema Administrativo Departamental Secretario Abogado

Un diagrama de casos de uso es la representación gráfica de parte o el total de los actores y casos de uso del sistema, incluyendo sus interacciones. Todo sistema

tiene como mínimo un diagrama Main Use Case, que es una representación gráfica del entorno del sistema (actores) y su funcionalidad principal (casos de uso).

Un diagrama de casos de uso muestra los distintos requisitos funcionales que se esperan de una aplicación o sistema y cómo se relaciona con su entorno (usuarios u otras aplicaciones).

En los siguientes Diagramas de Casos de Uso se muestra al o los actores los cuales representan a los integrantes del dpto. Secretario-Abogado, navegadores de la red y usuarios invitados de los diversos módulos integrantes del Sistema Administrativo Secretario - Abogado.

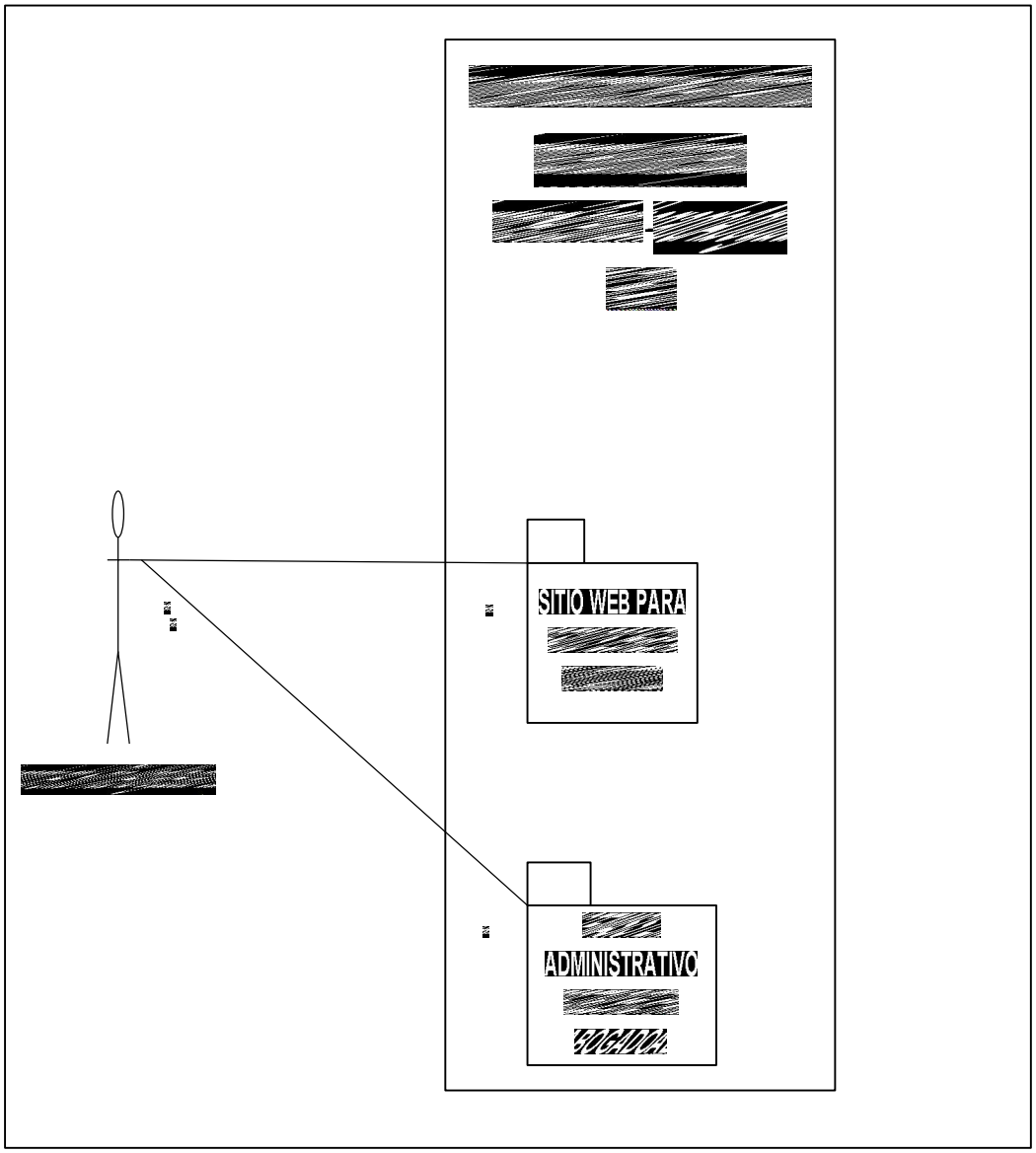


Figura 5.2 Caso de Uso Administración Central del Sistema

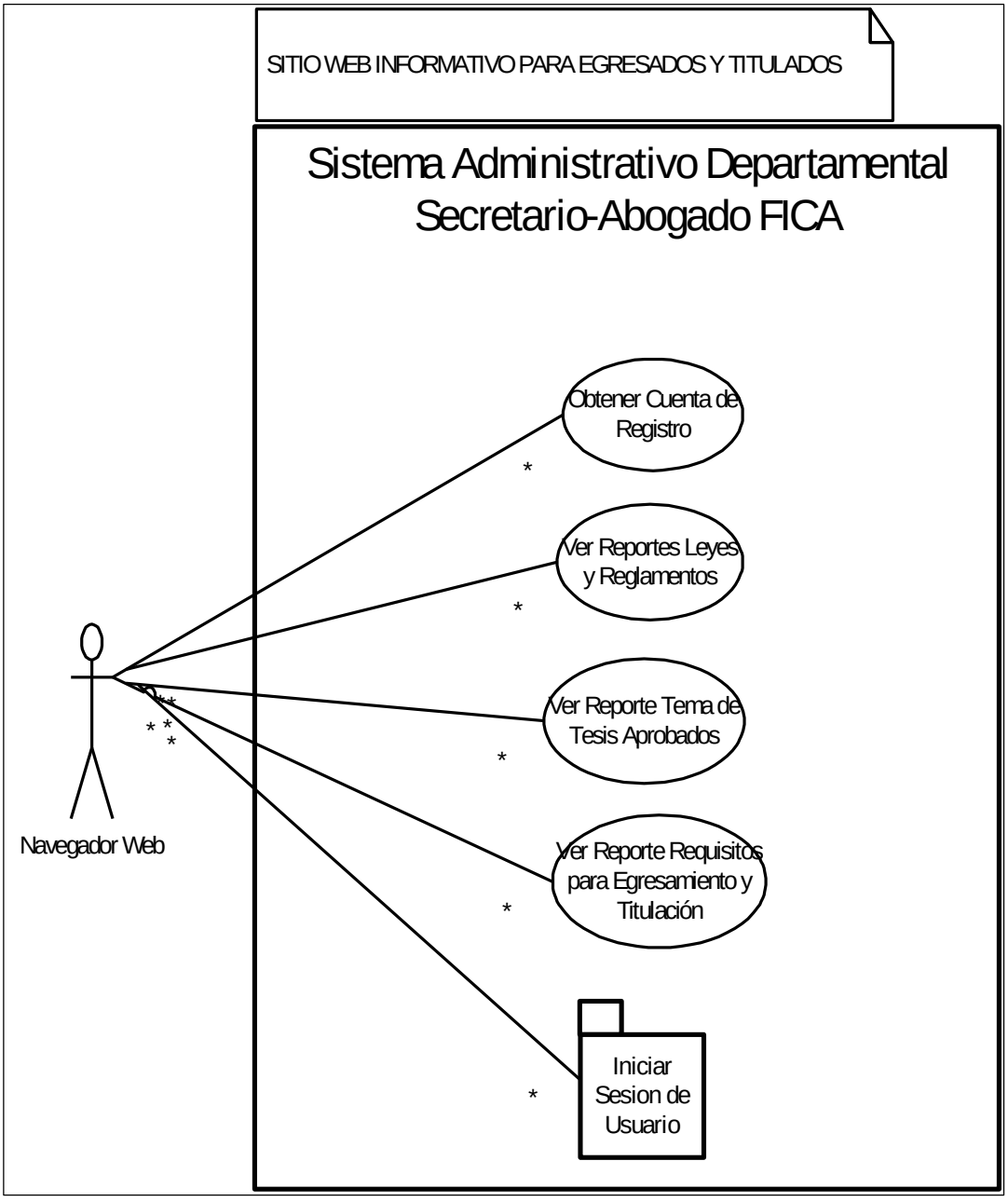


Figura 5.3 Caso de Uso Sitio Web Informativo

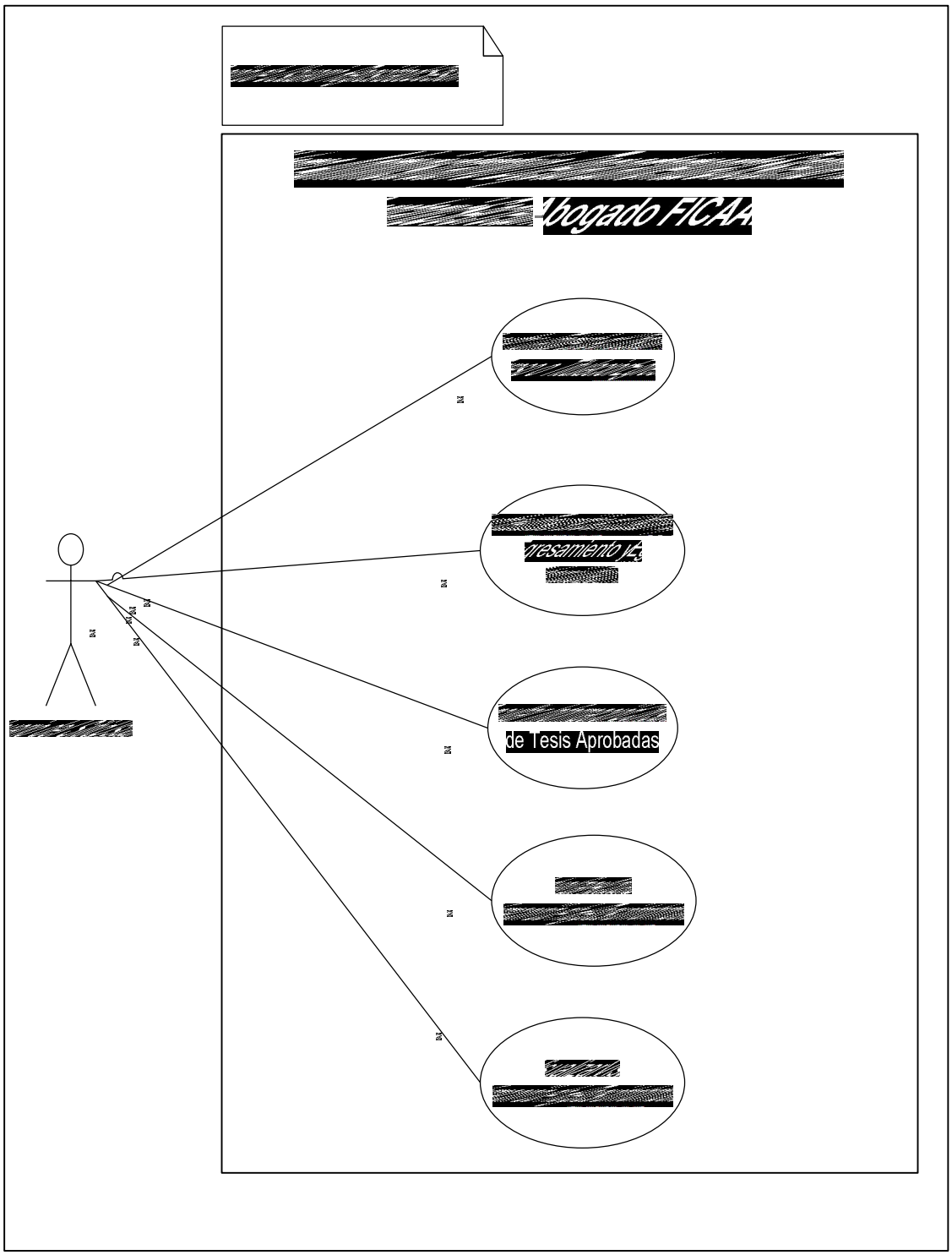


Figura 5.4 Caso de Uso Sesión de Usuario Registrado Sitio Web Informativo

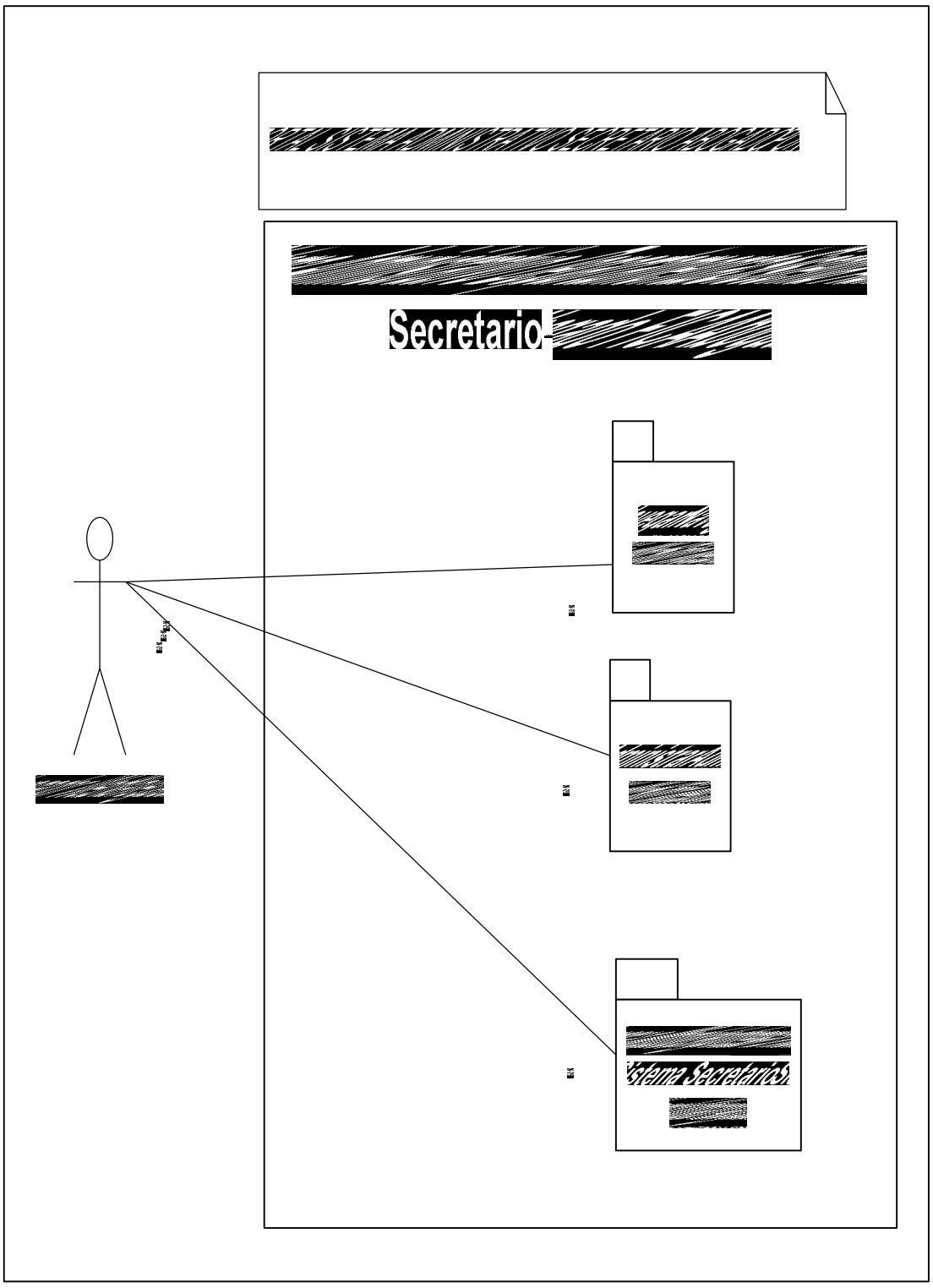


Figura 5.5 Caso de Uso Portal Administrativo del Sistema

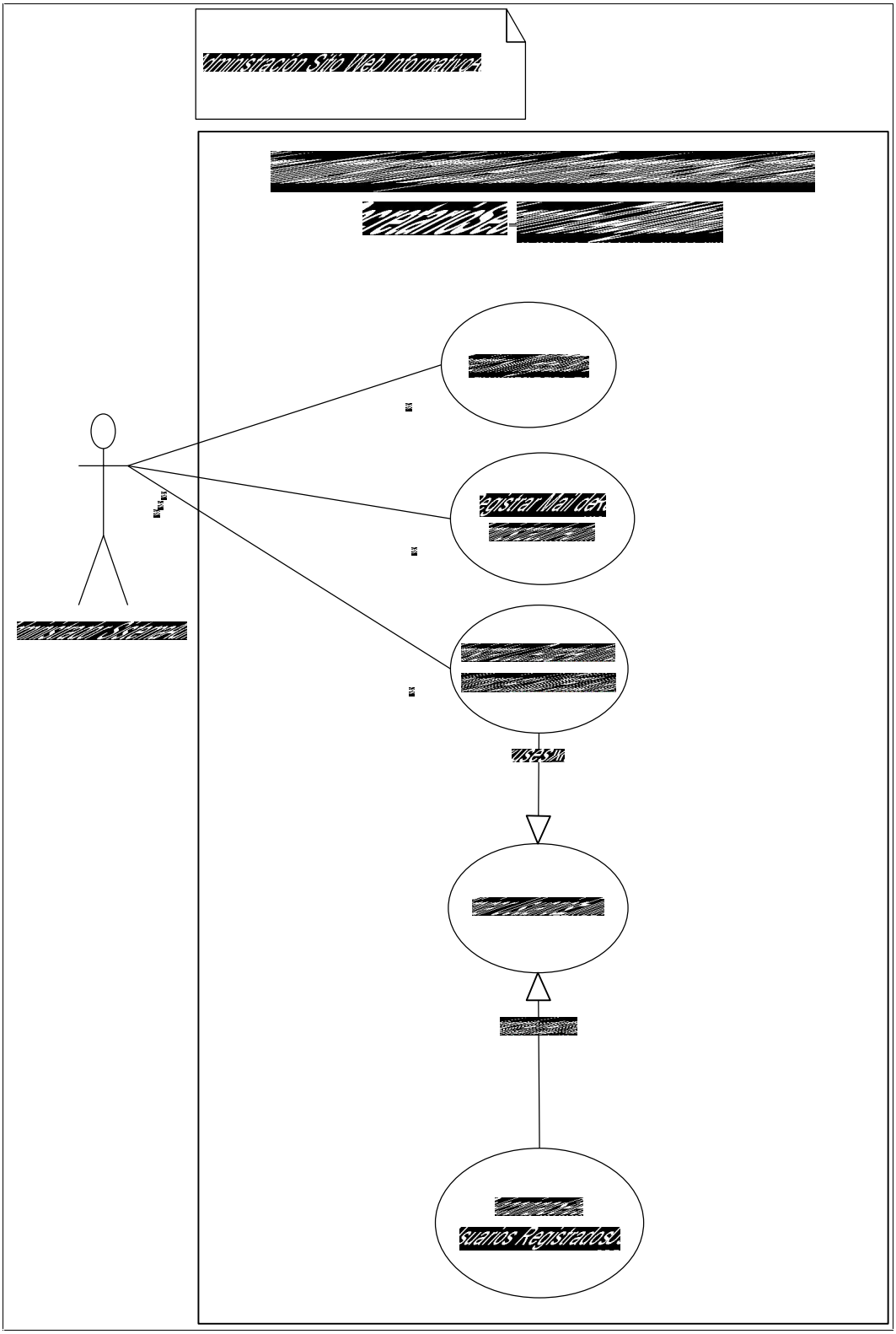


Figura 5.6 Caso de Uso Administración Sitio Informativo

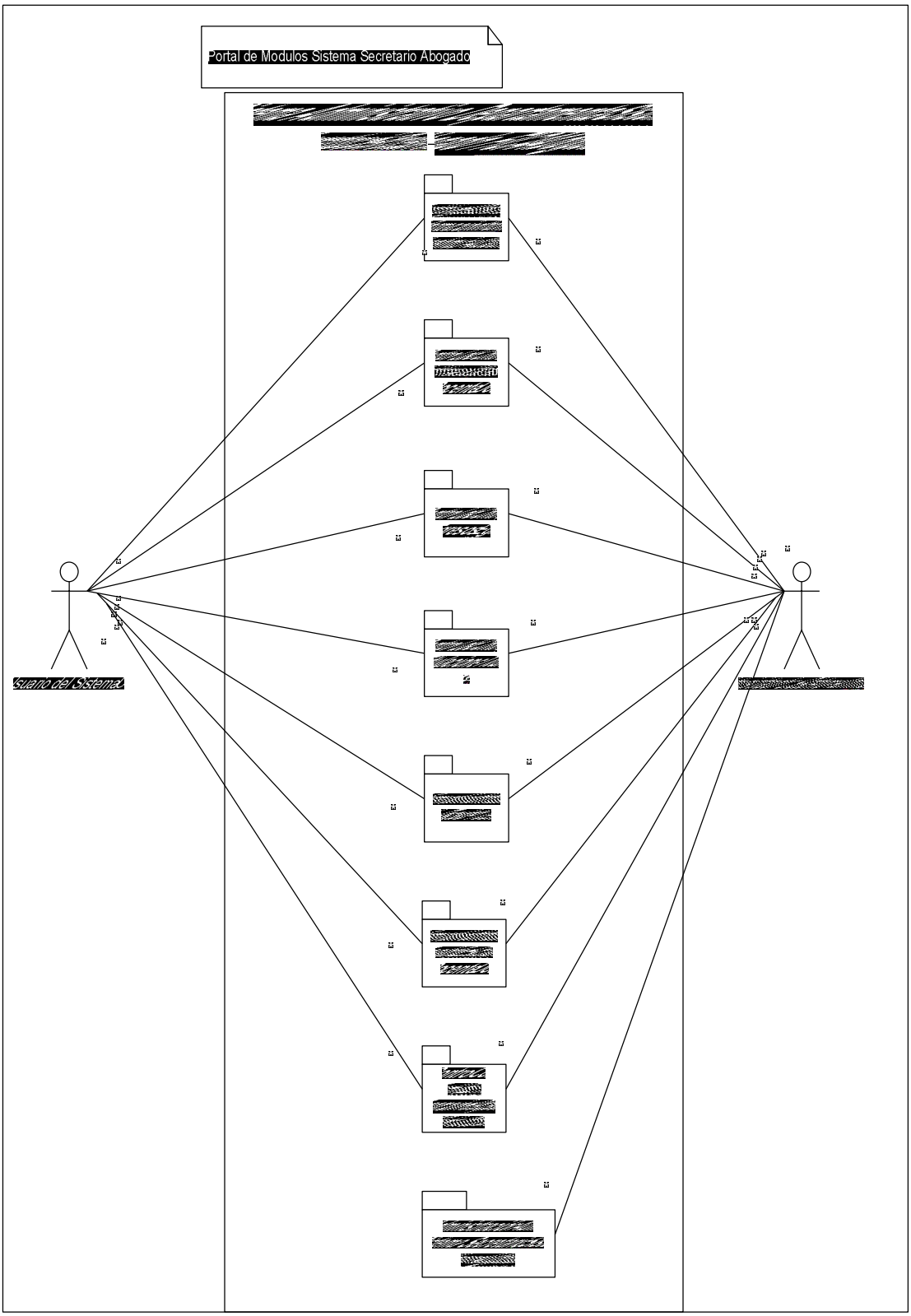


Figura 5.7 Caso de Uso Portal de Módulos del Sistema

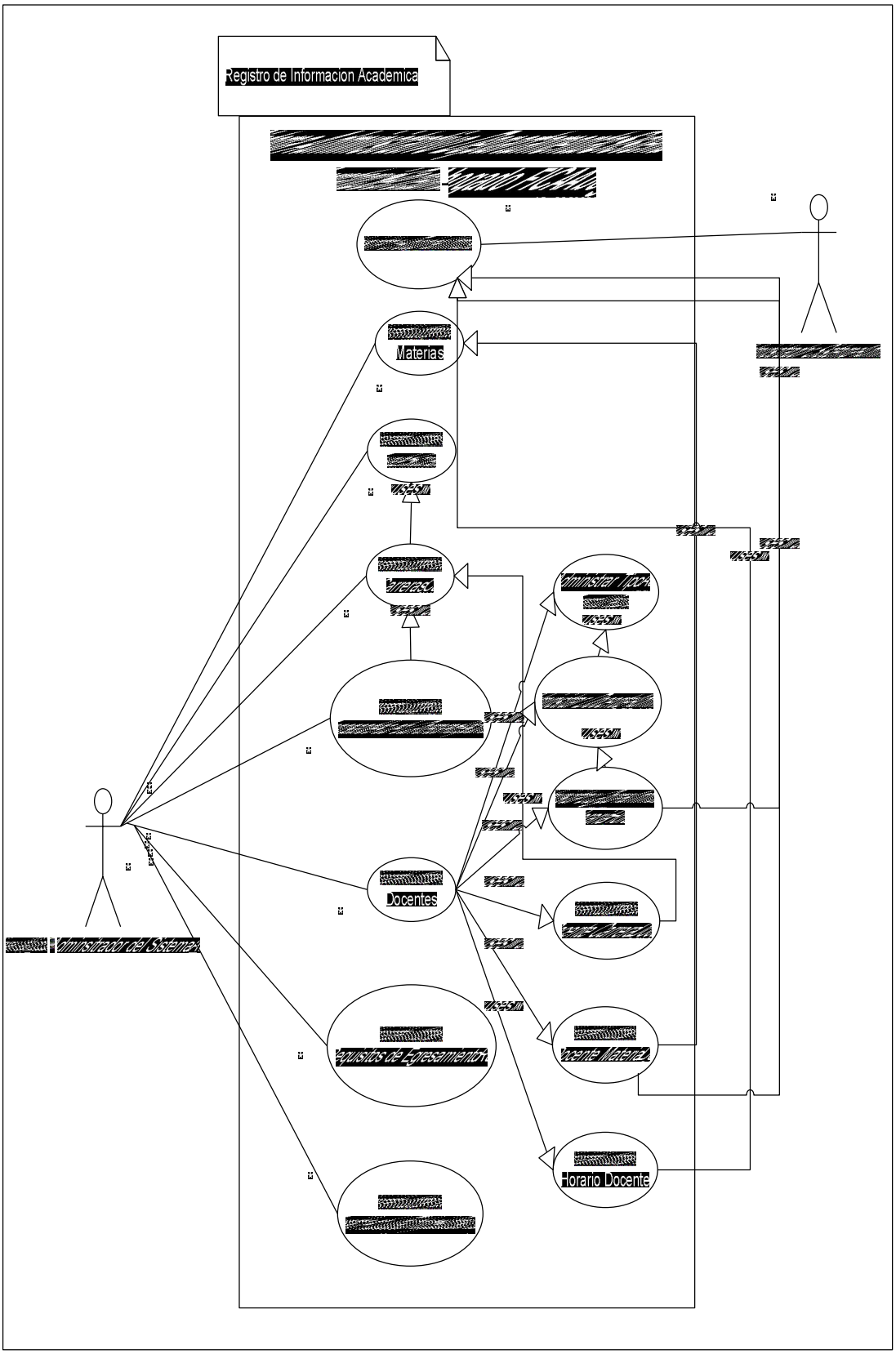


Figura 5.8 Caso de Uso Información Académica

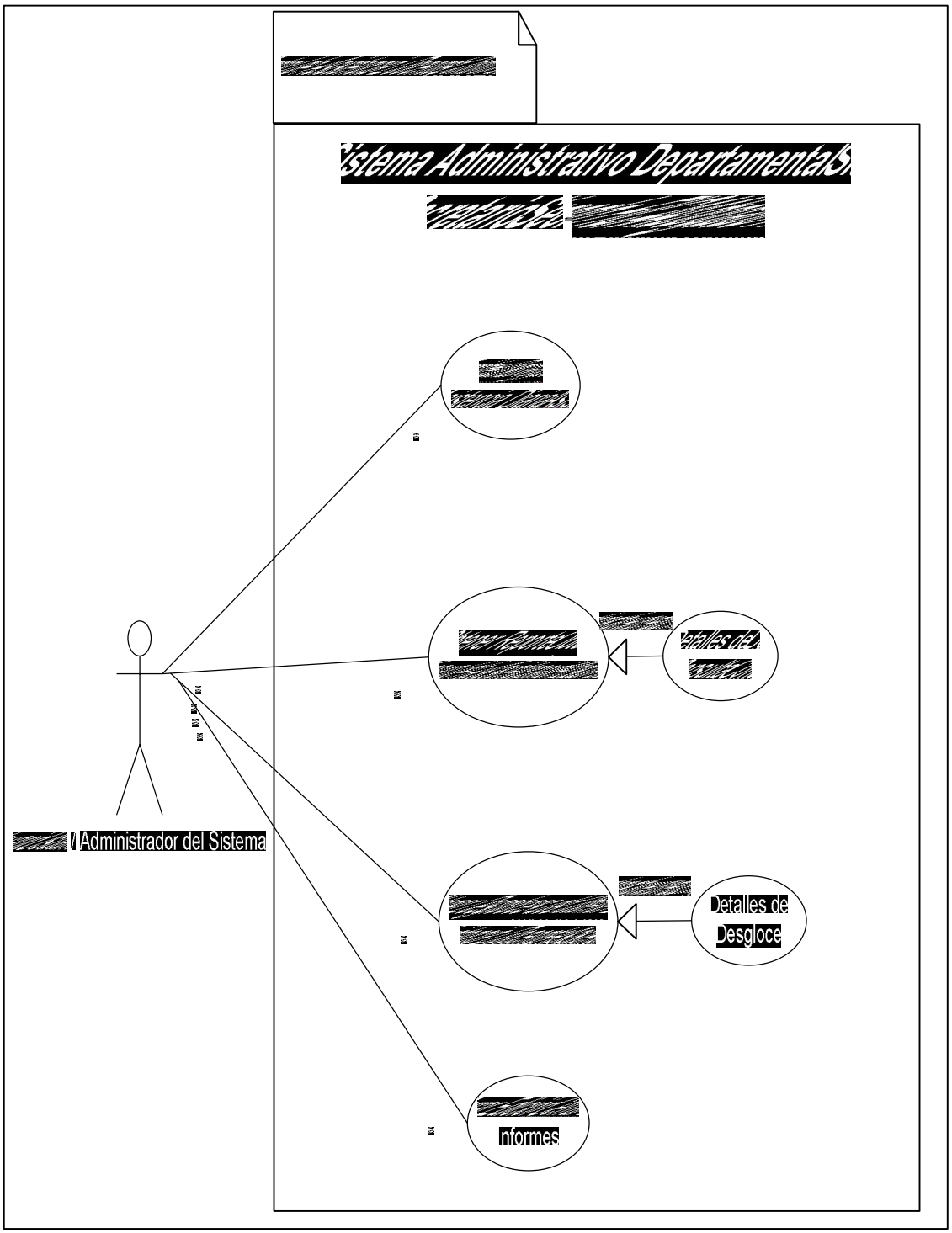


FIGURA 5.9 Caso de Uso Control de Asistencia

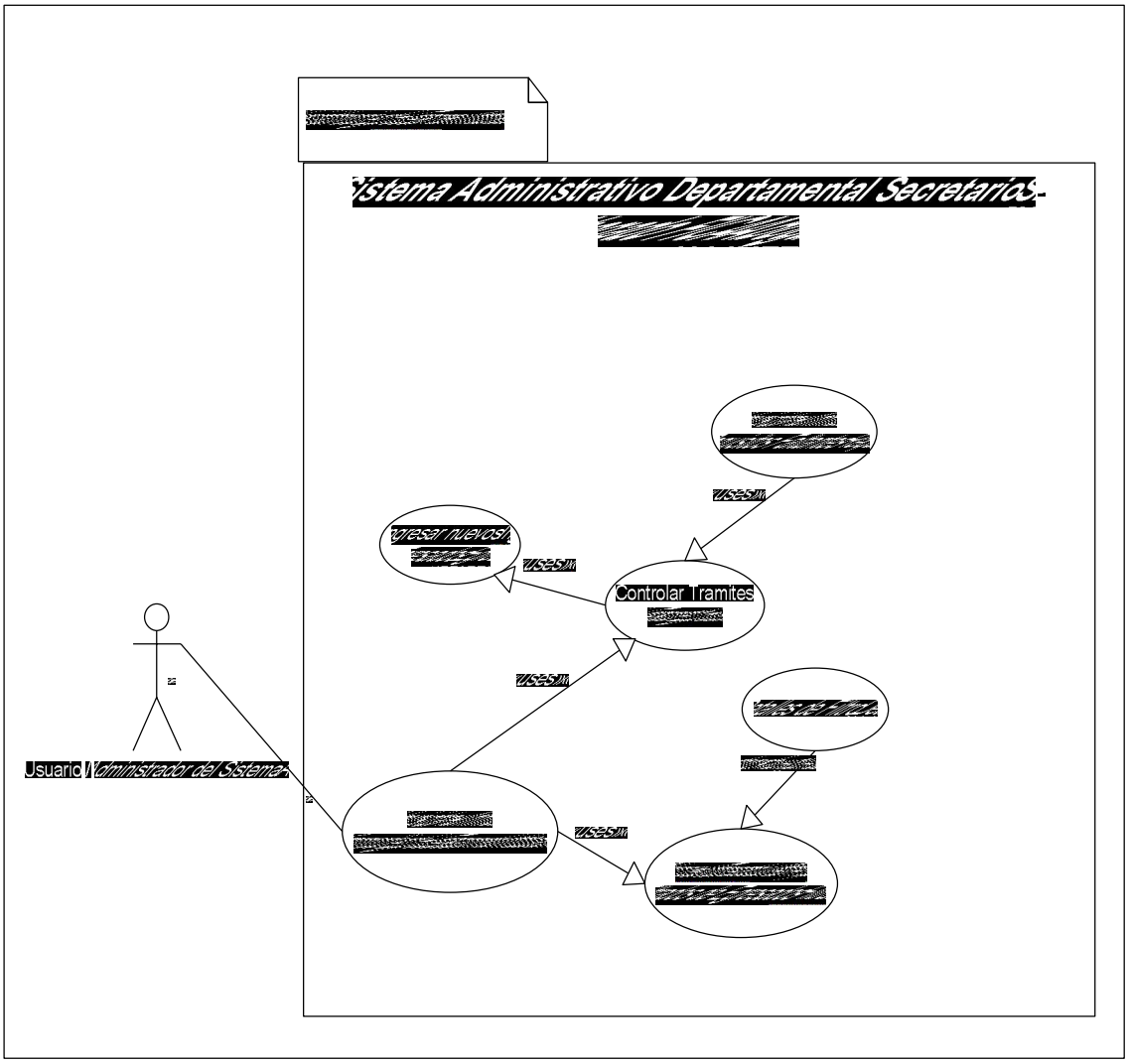


Figura 5.10 Caso de Uso Administración de Egresos

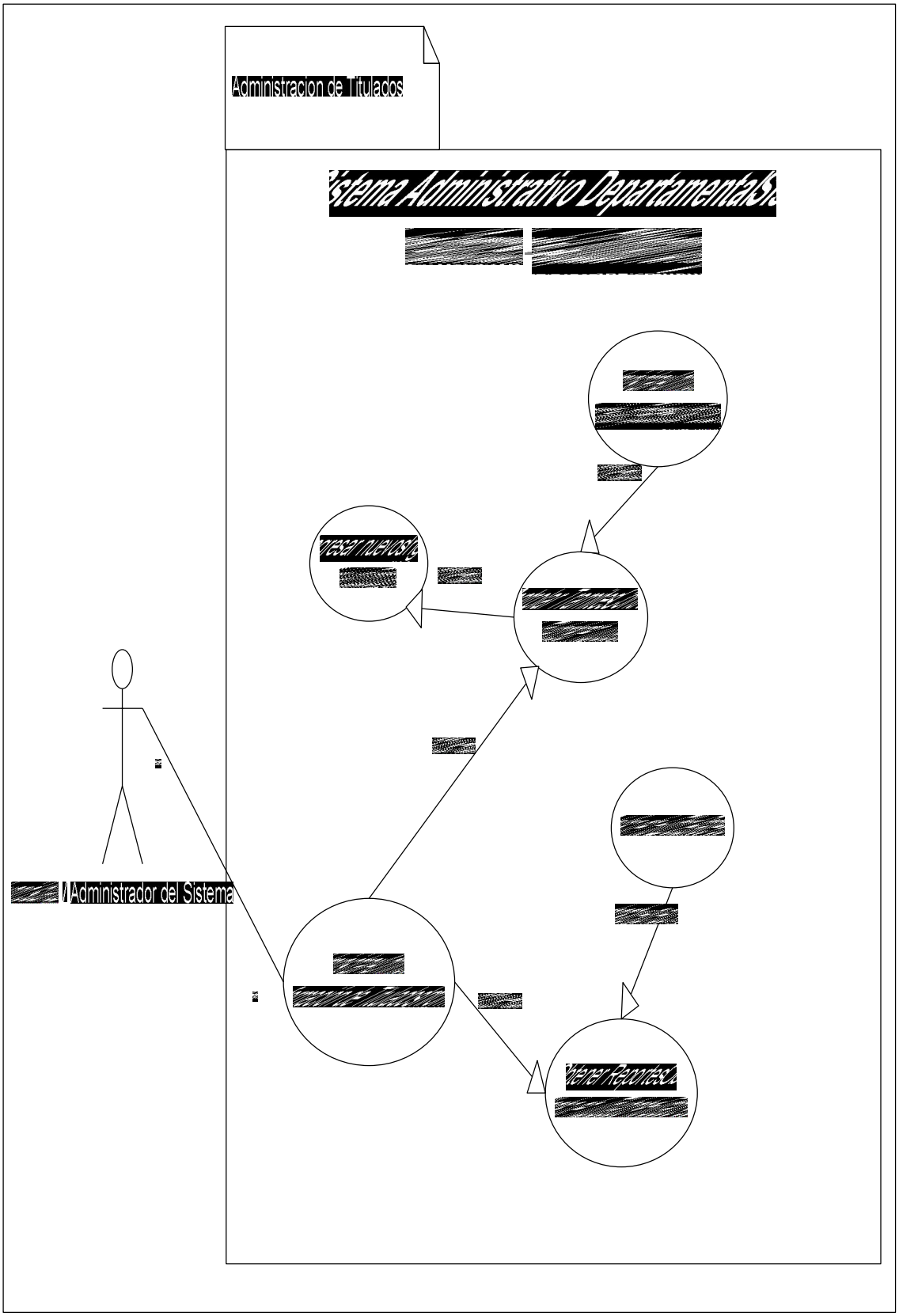


FIGURA 5.11 Caso de Uso Administración de Titulaciones

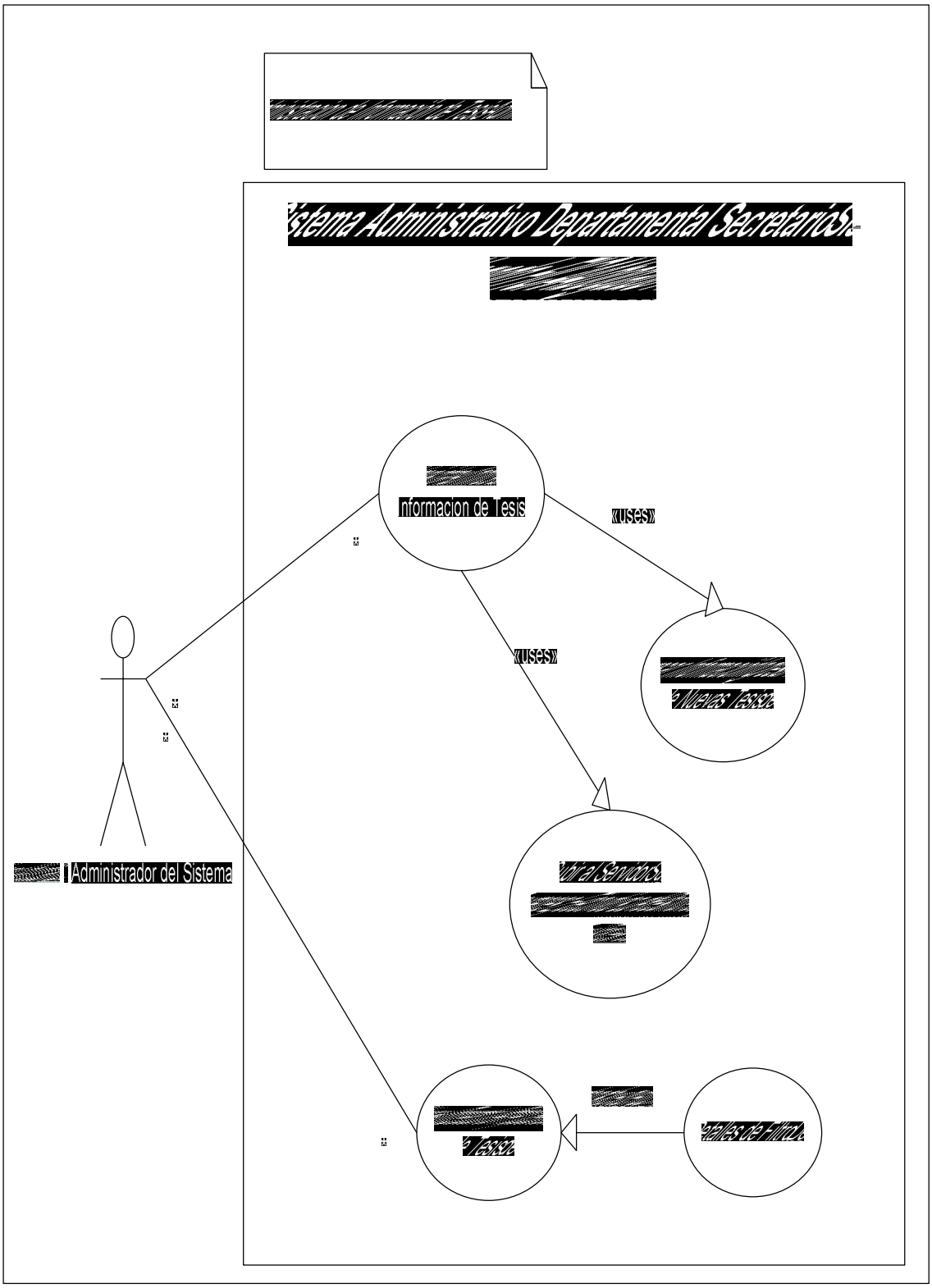


Figura 5.12 Caso de Uso Administración de Tesis

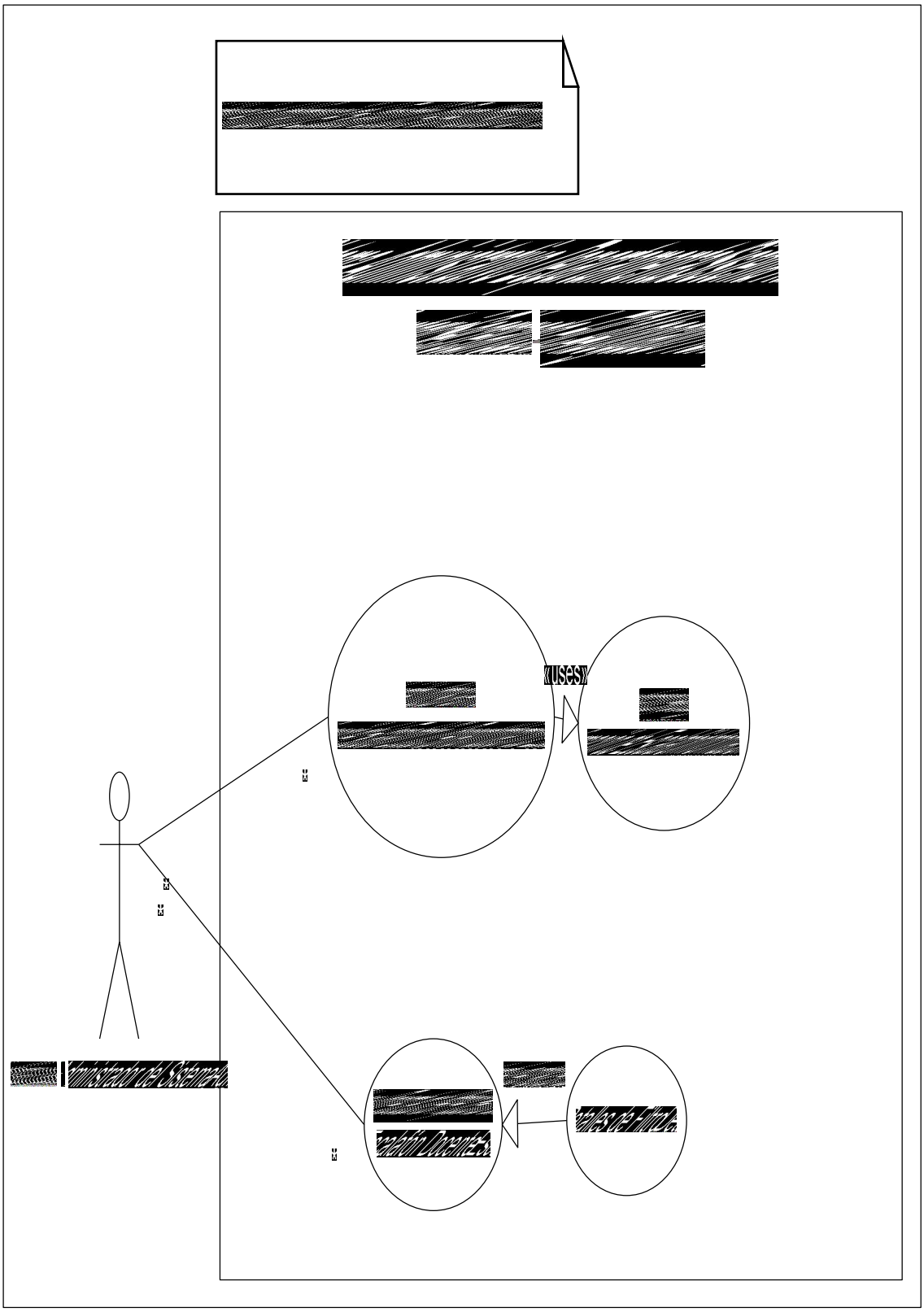


Figura 5.13 Caso de Uso Administración de Información de Escalafón Docente

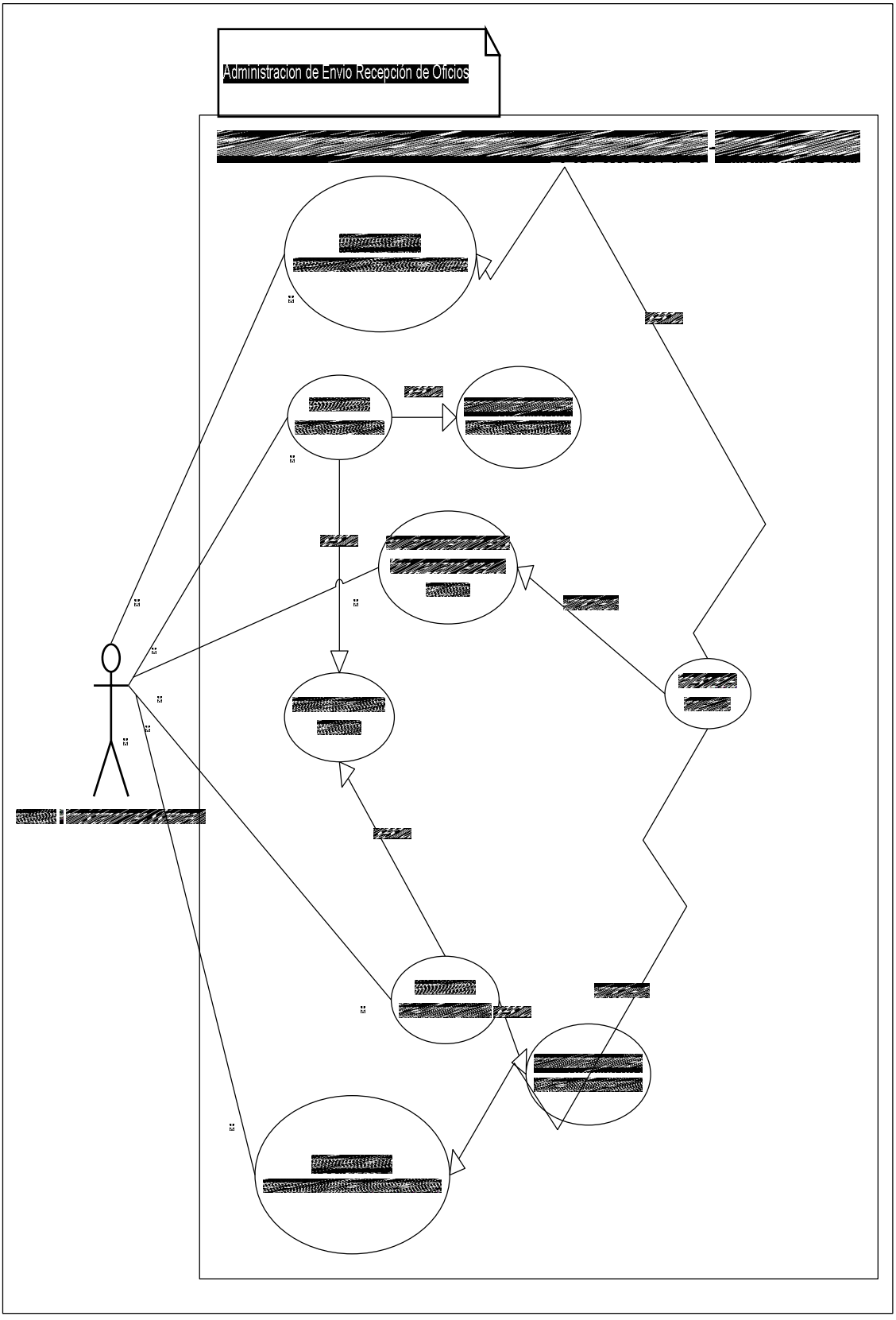


Figura 5.14 Caso de Uso Administración de Envío-Recepción de Oficios

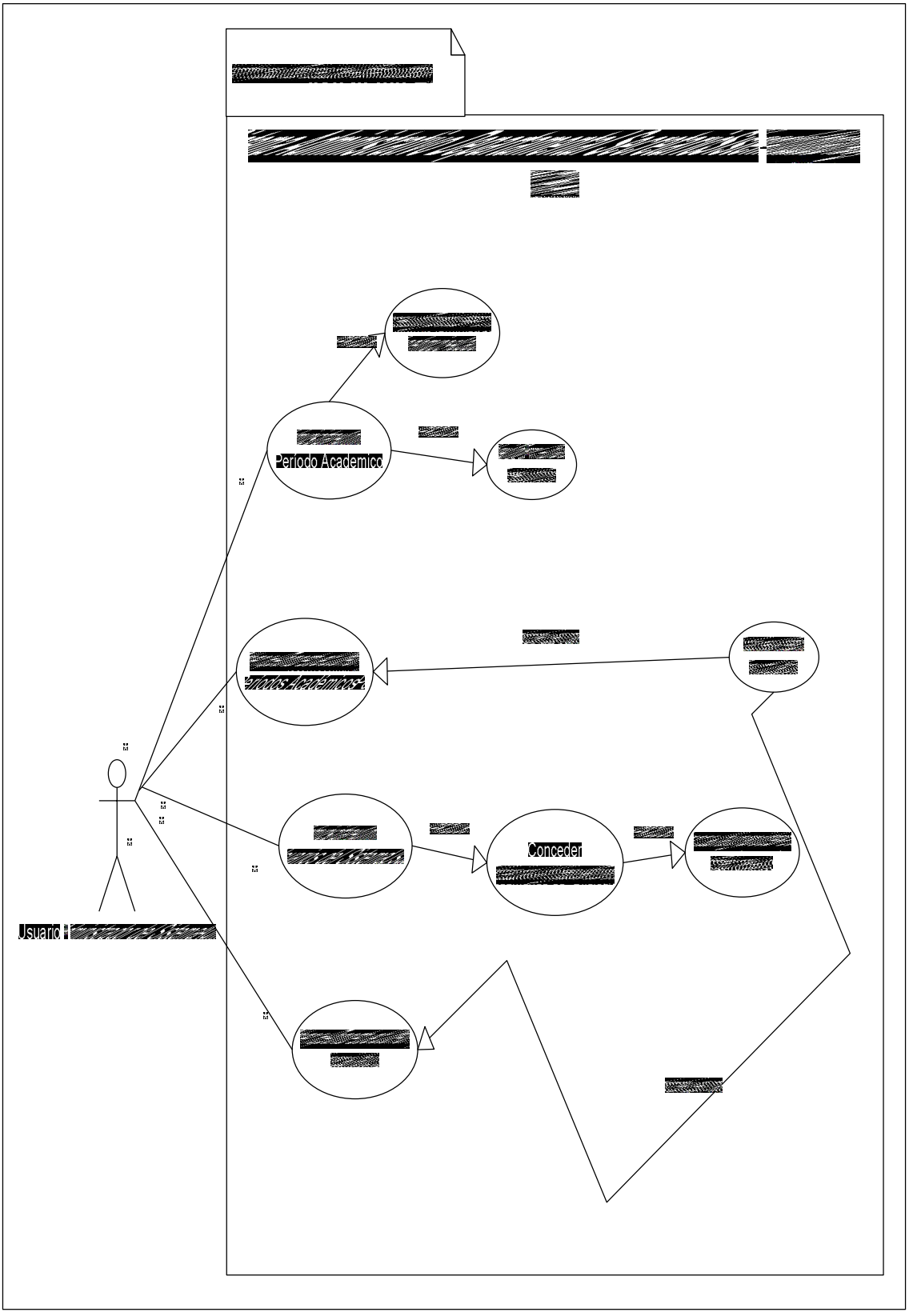


Figura 5.15 Caso de Uso Administración de Sistema

5.5.3. Diagrama de Clases del Sistema Administrativo Departamental Secretario Abogado

El Diagrama de Clases es el diagrama principal para realizar el análisis y diseño de un sistema. Un diagrama de clases presenta las clases del sistema con sus relaciones estructurales y de herencia.

A continuación se detallan los Diagramas de Clases del Sistema Administrativo Secretario Abogado. Se encuentran divididos por módulos de aplicación para una mayor organización y comprensión.

Figura 5.16 Diagrama de Clase Administración Sitio Web Informativo

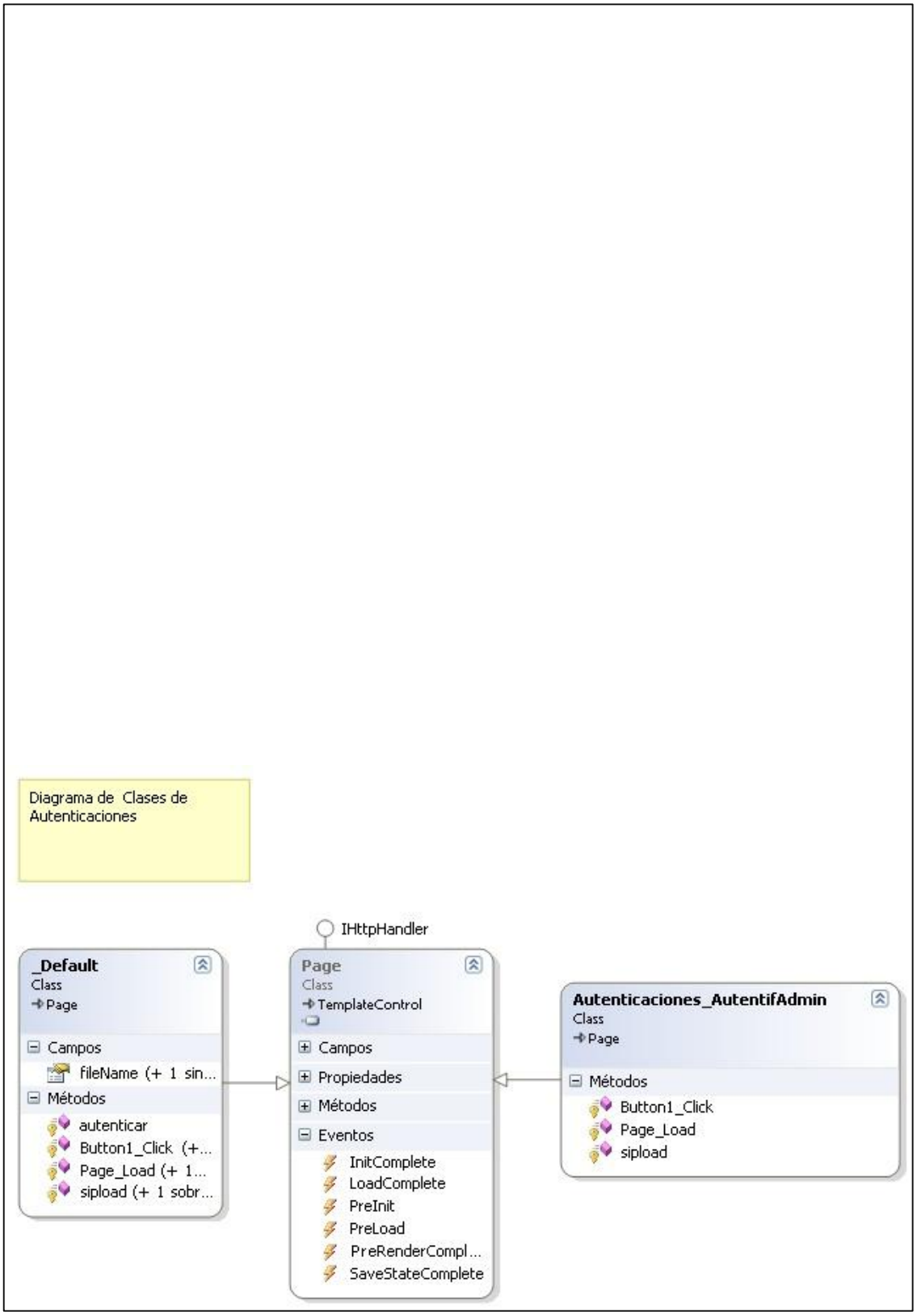


Figura 5.17 Diagrama de Clases Autenticación Central Sistema

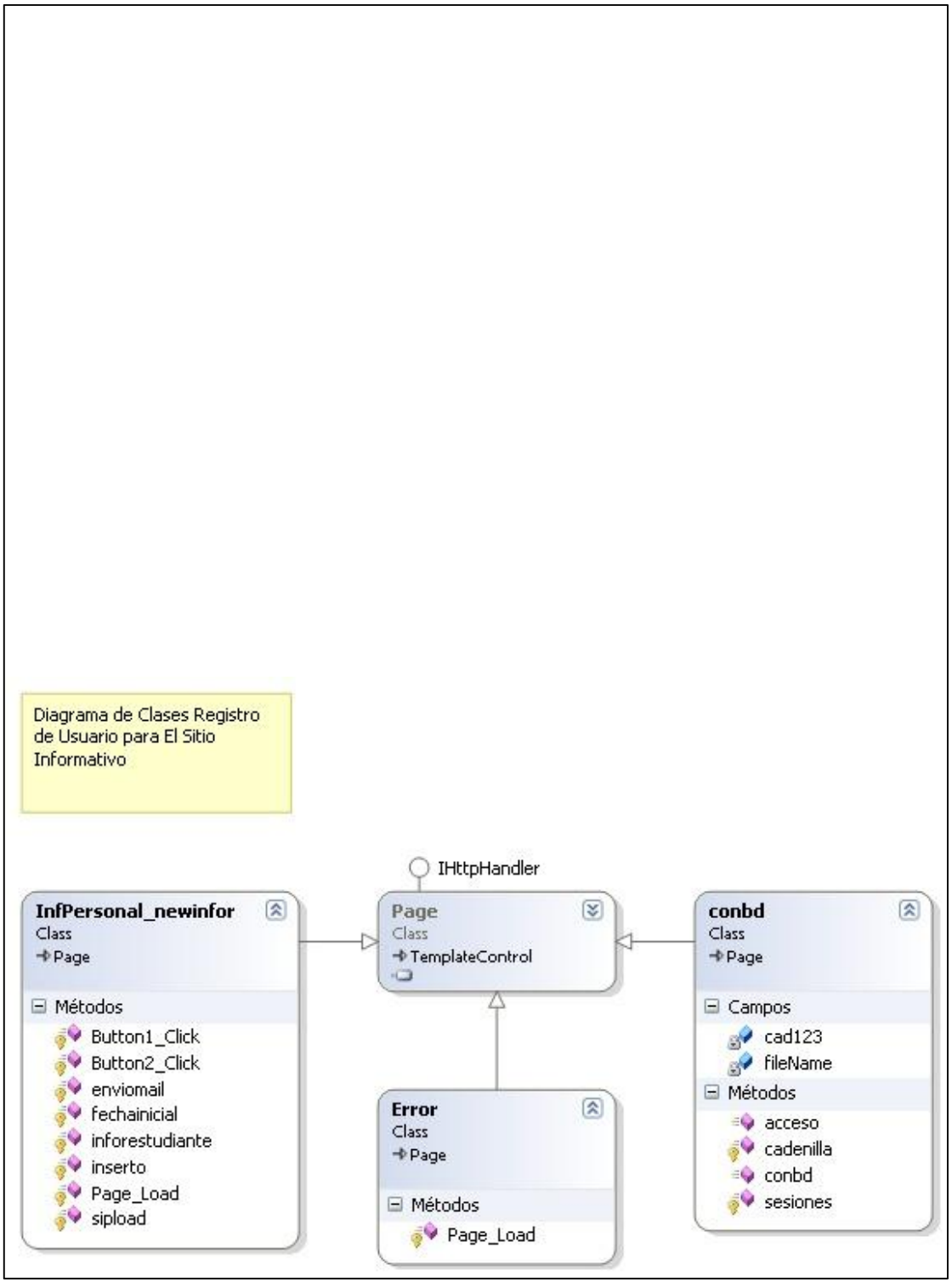


Figura 5.18 Diagrama de Clases Registro Sitio

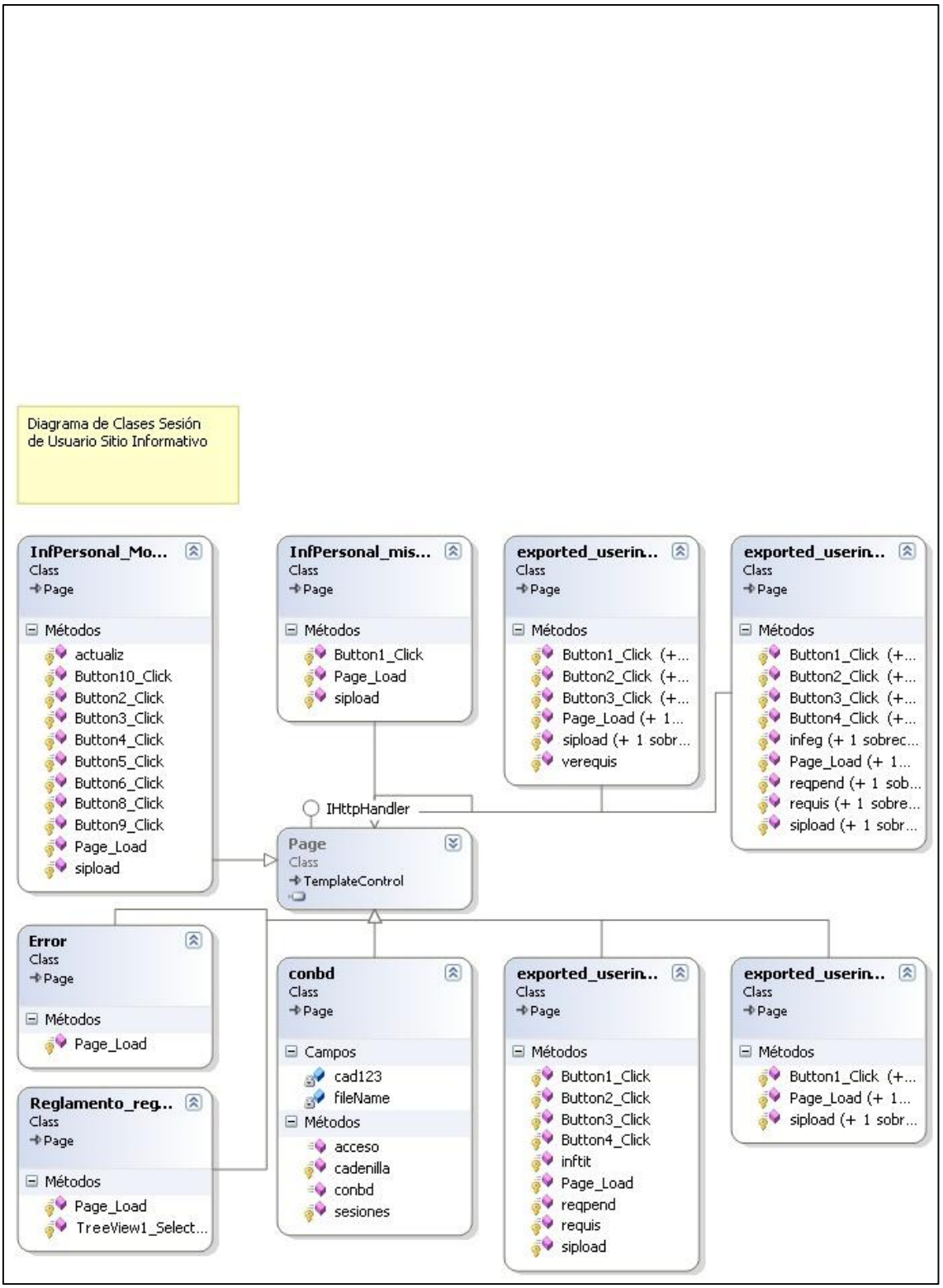


Figura 5.19 Diagrama de Clases Sesión de Usuario Registrado al Sitio

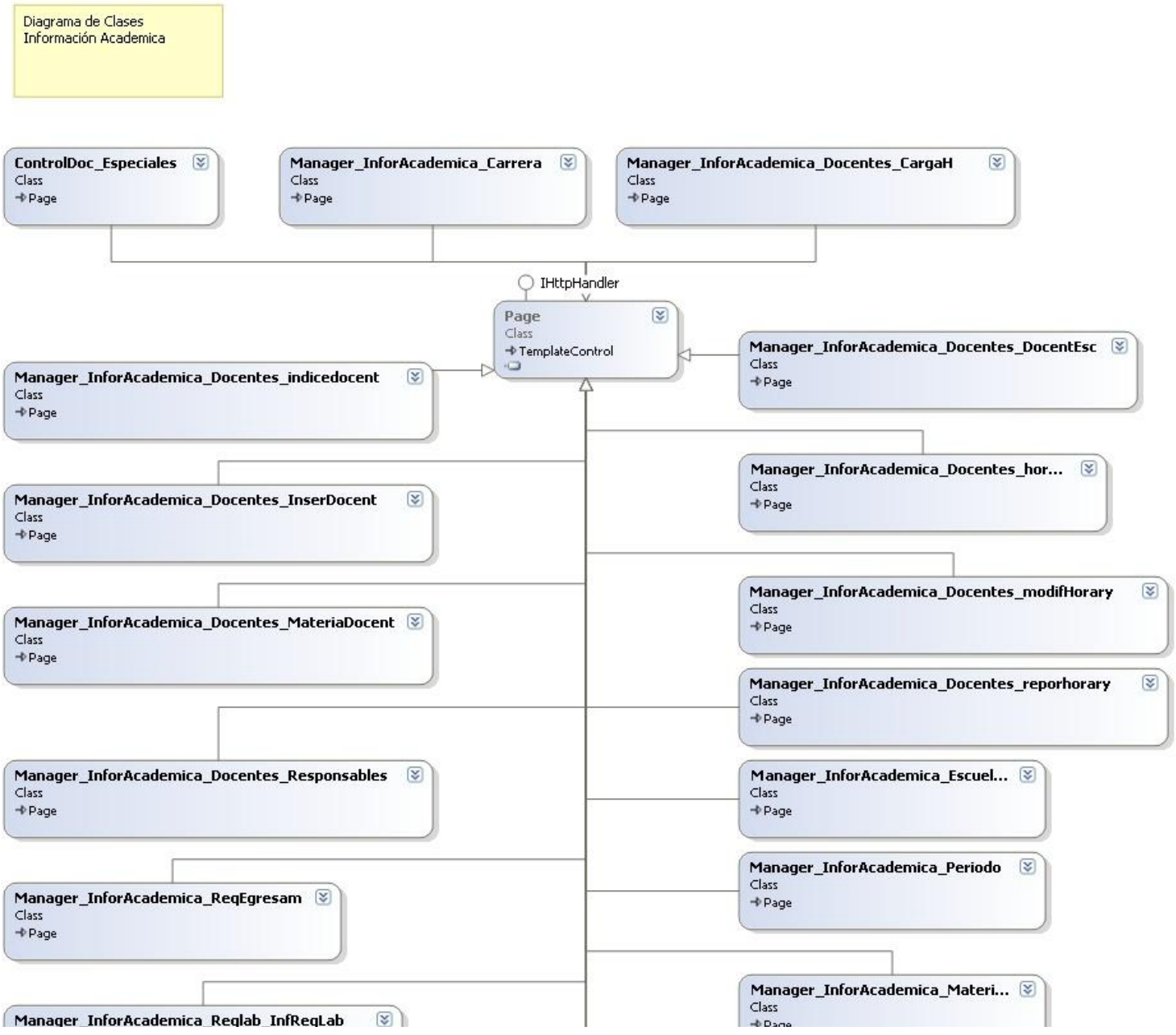


Diagrama de Clases
Información Academica
ControlDoc_Especiales
Class
Page
Manager_InforAcademica_Carrera
Class
Page
Manager_InforAcademica_Docentes_CargaH
Class
Page
IHttpHandler
Page
Class
TemplateControl
Manager_InforAcademica_Docentes_DocentEsc
Class
Page
Manager_InforAcademica_Docentes_indicedocent
Class
Page
Manager_InforAcademica_Docentes_hor...
Class
Page
Manager_InforAcademica_Docentes_InserDocent
Class
Page
Manager_InforAcademica_Docentes_modifHorary
Class
Page
Manager_InforAcademica_Docentes_MateriaDocent
Class
Page
Manager_InforAcademica_Docentes_reporhorary
Class
Page
Manager_InforAcademica_Docentes_Responsables
Class
Page
Manager_InforAcademica_Escuel...
Class
Page
Manager_InforAcademica_ReqEgresam
Class
Page
Manager_InforAcademica_Periodo
Class
Page
Manager_InforAcademica_Materi...
Class
Page
Manager_InforAcademica_Reglab_InfReqLab

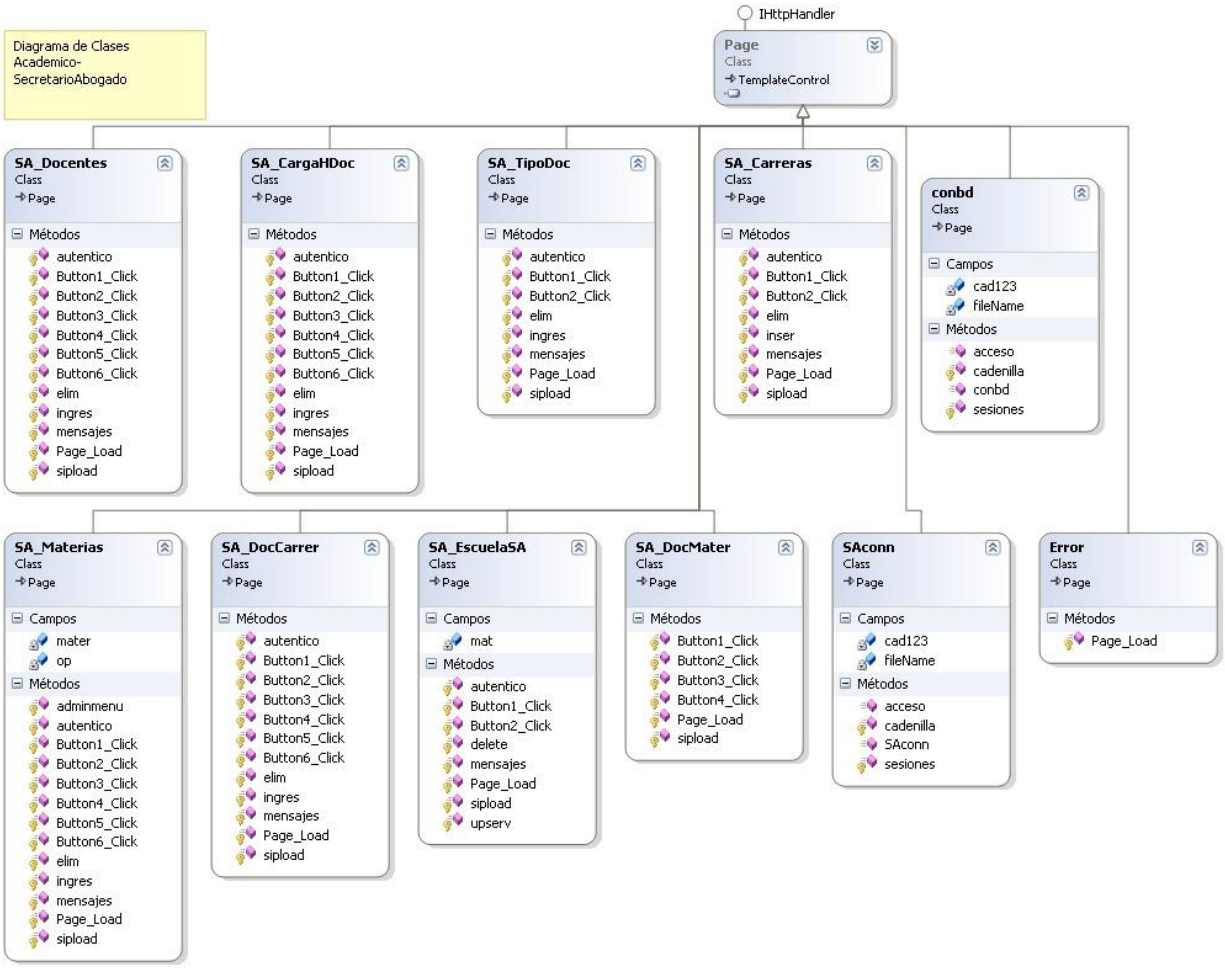
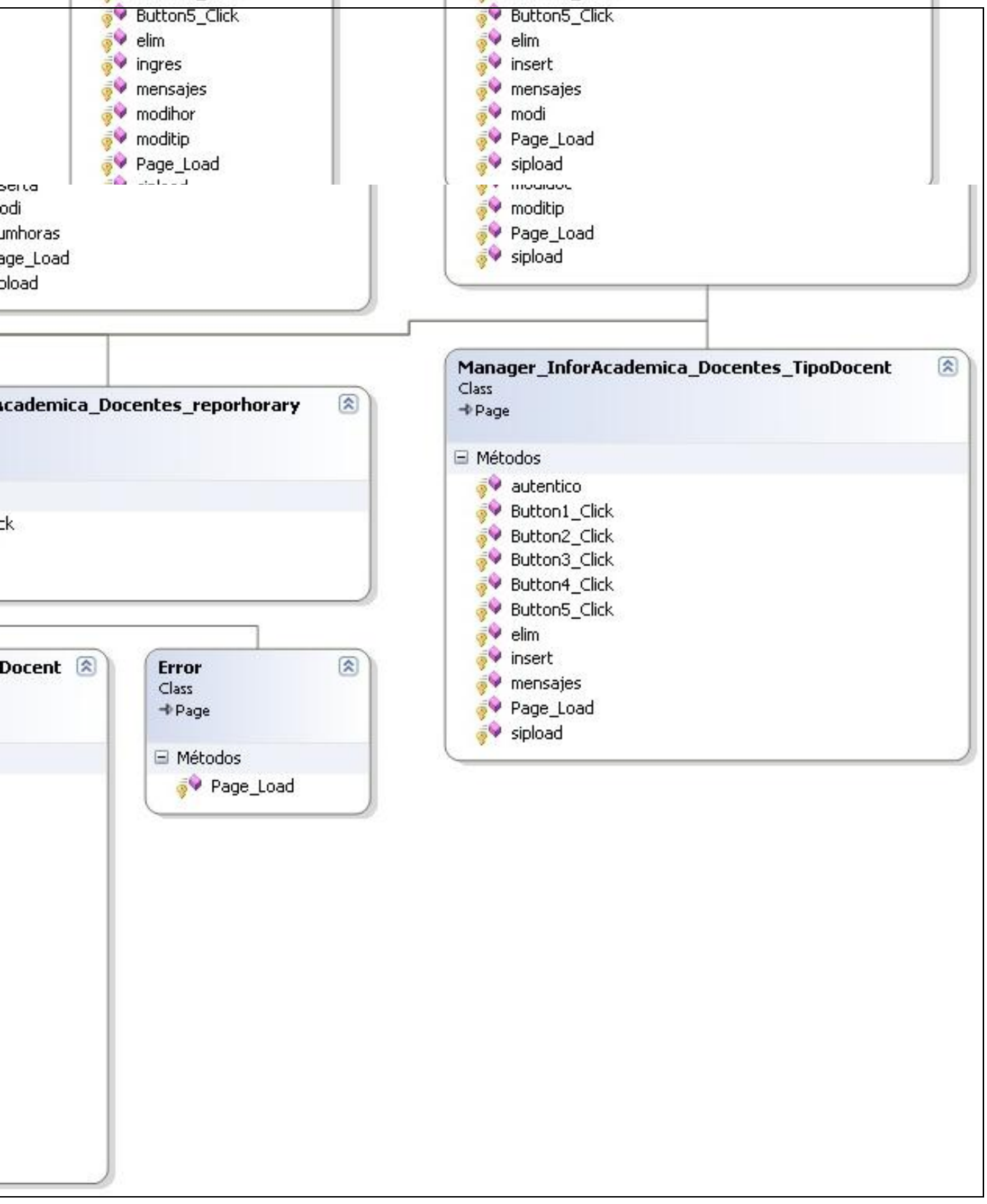


Figura 5.21 Diagrama de Clases Académico-Secretario Abogado



a de Clase Docentes

Figura 5.23 Diagrama de Clases Control Docente

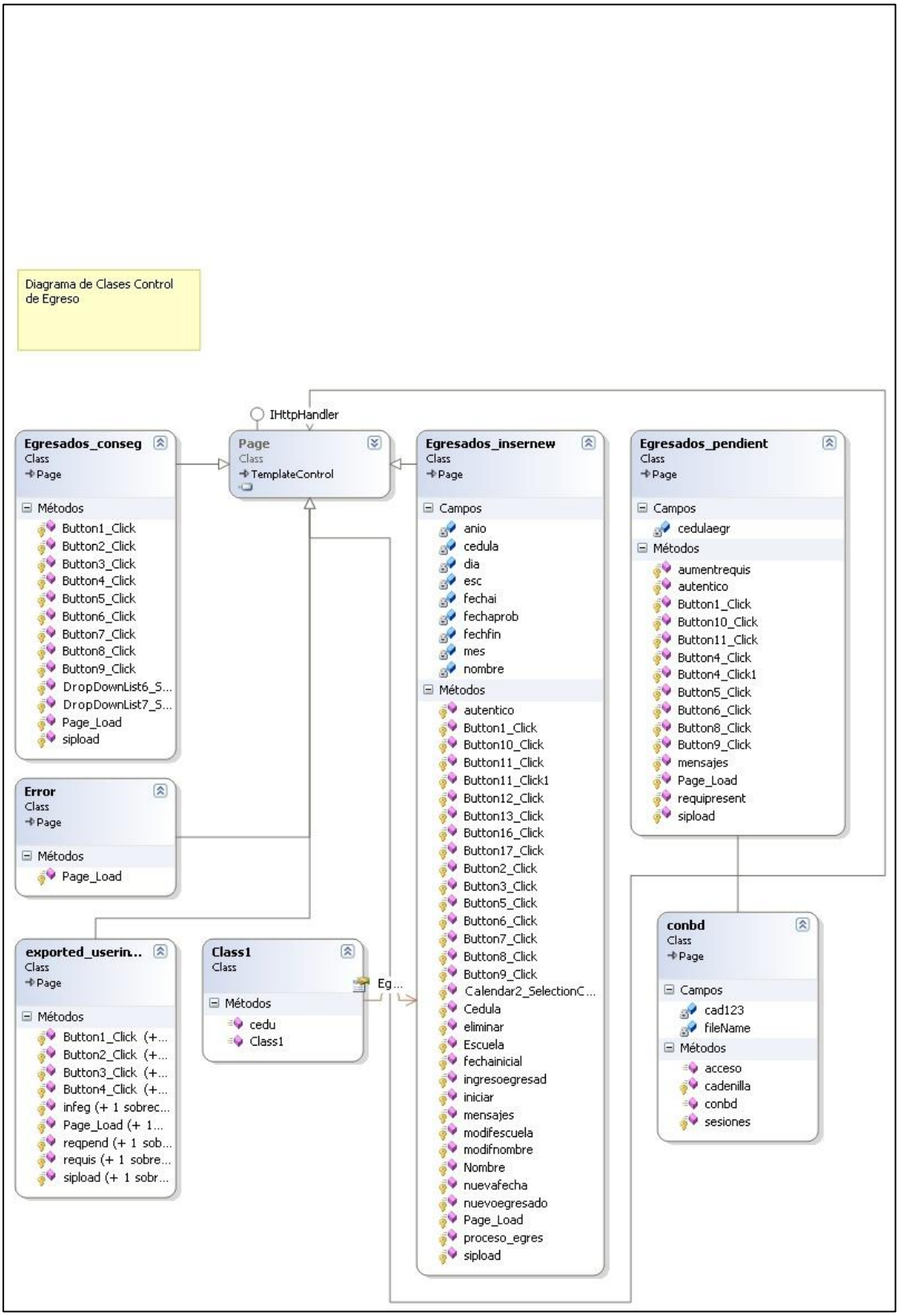


Figura 5.24 Diagrama de Clases Egresos

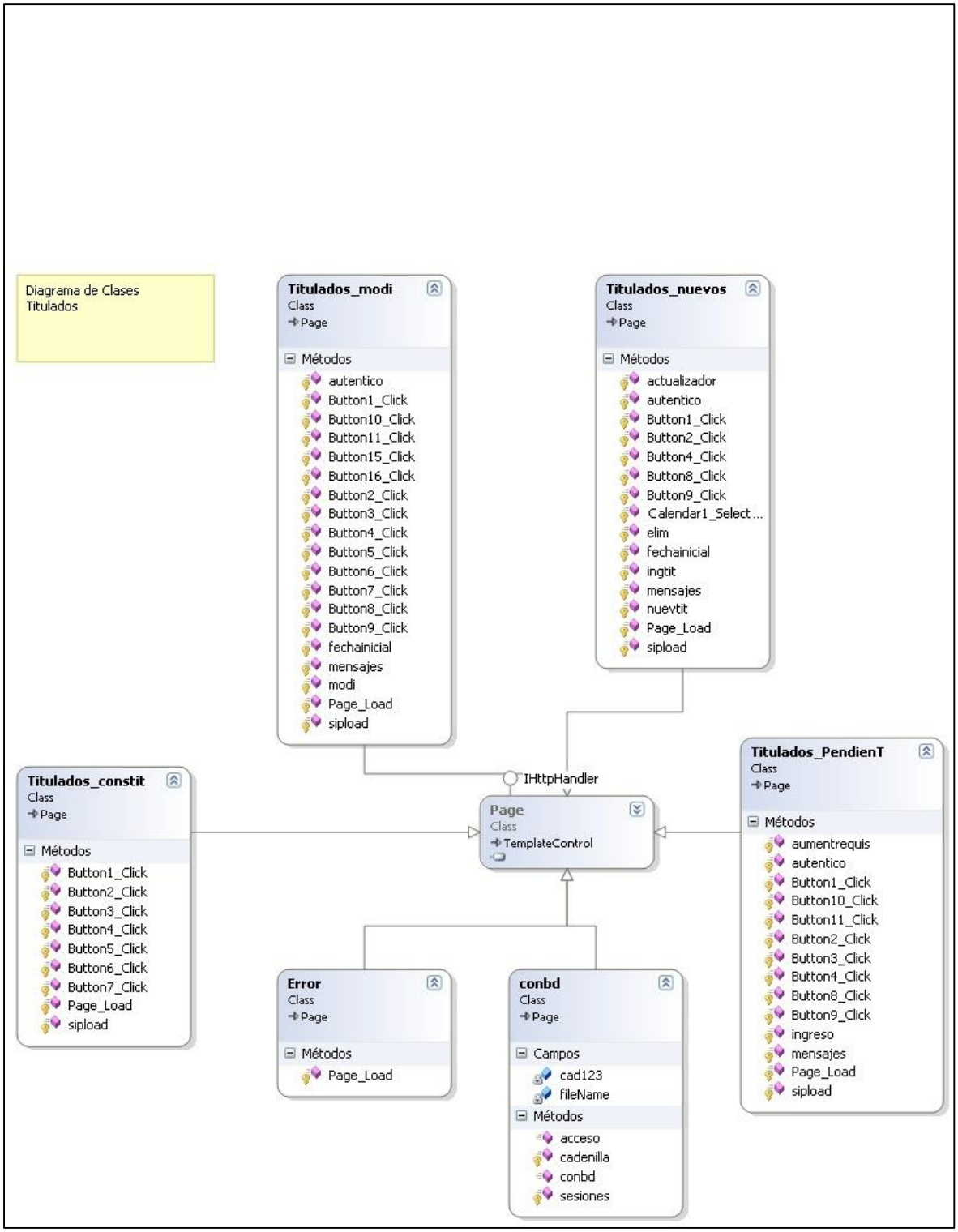


Figura 5.25 Diagrama de clases Titulaciones

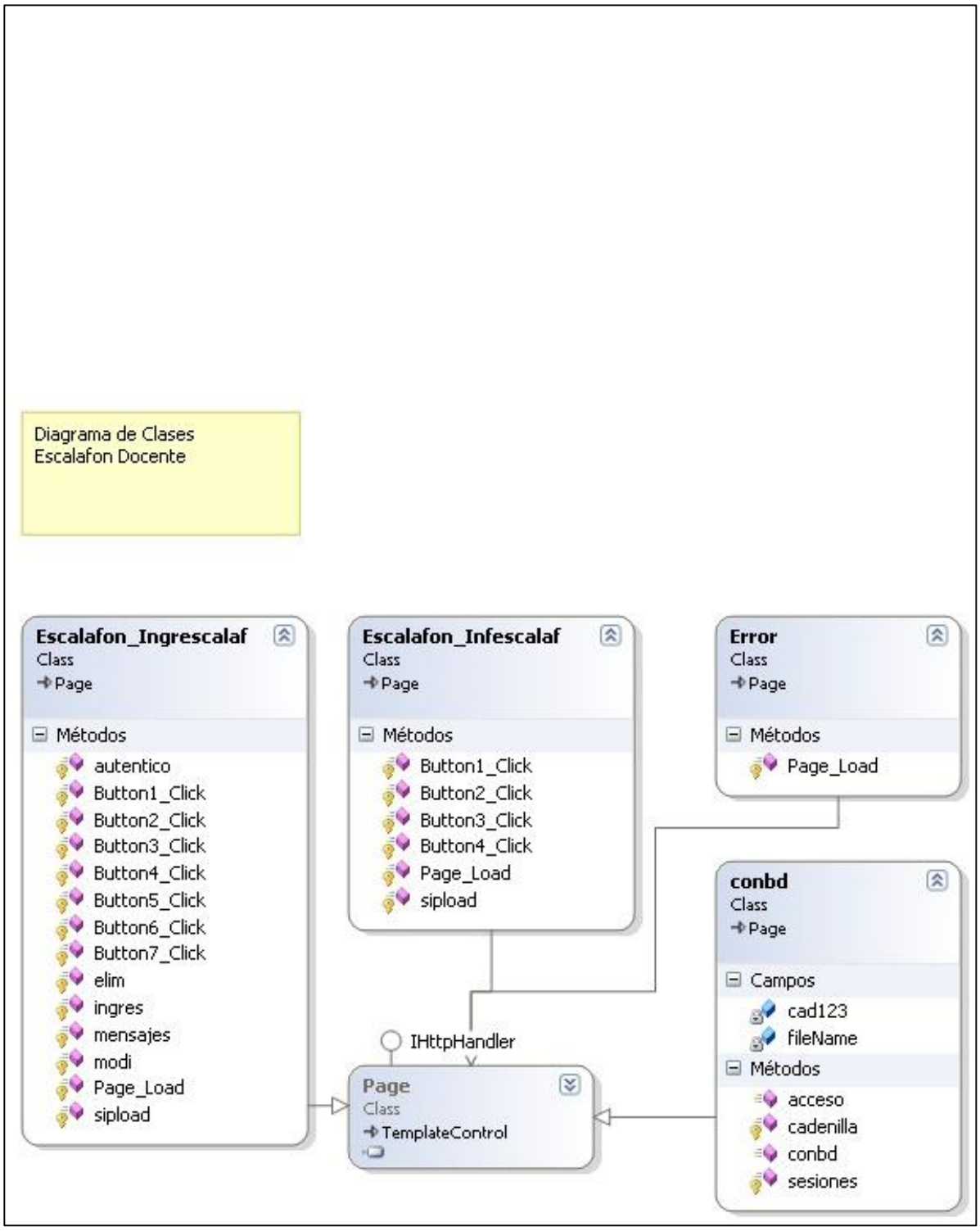


Figura 5.26 Diagrama de Clases Escalafón

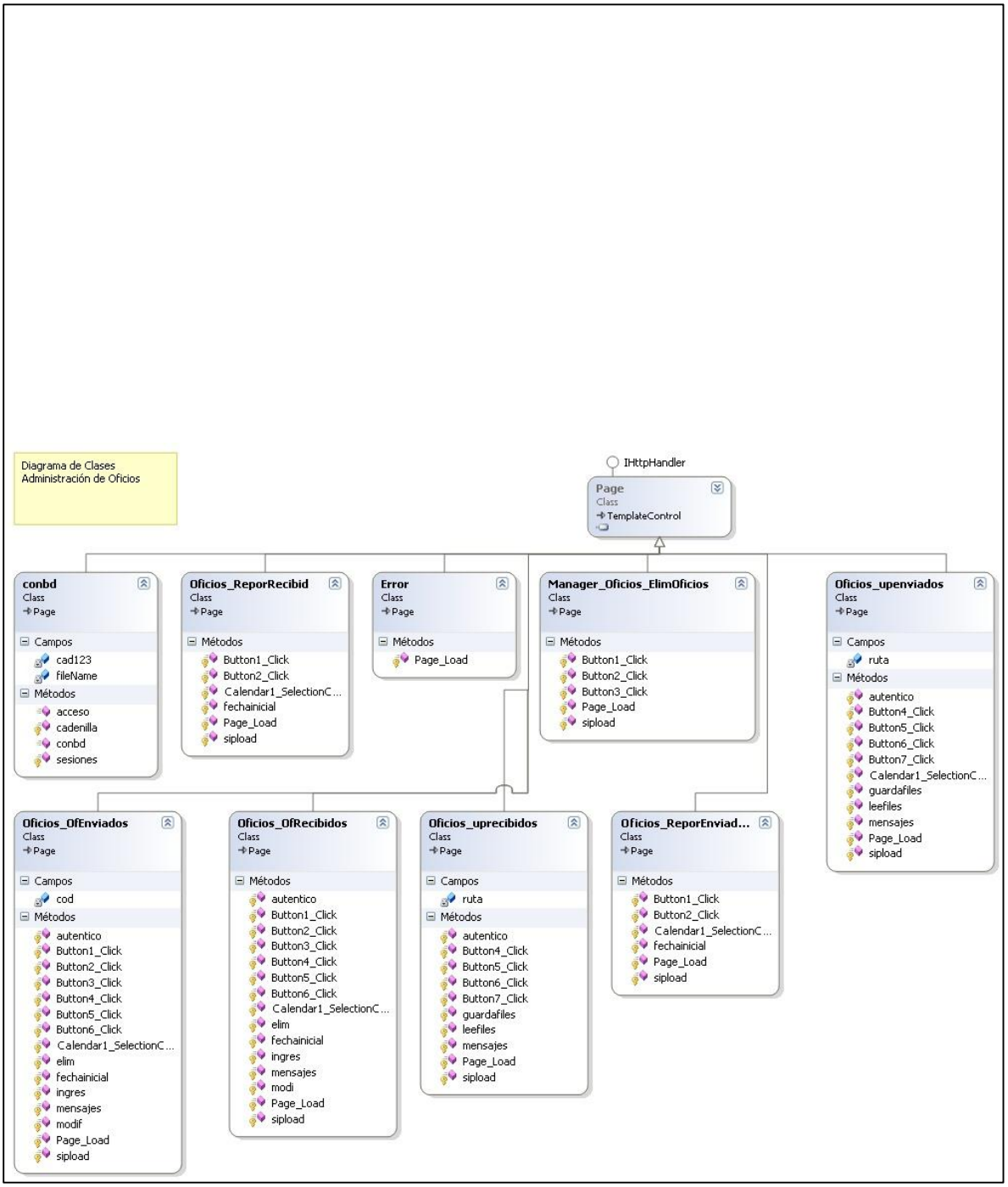


Figura 5.27 Diagrama de Clases Oficios

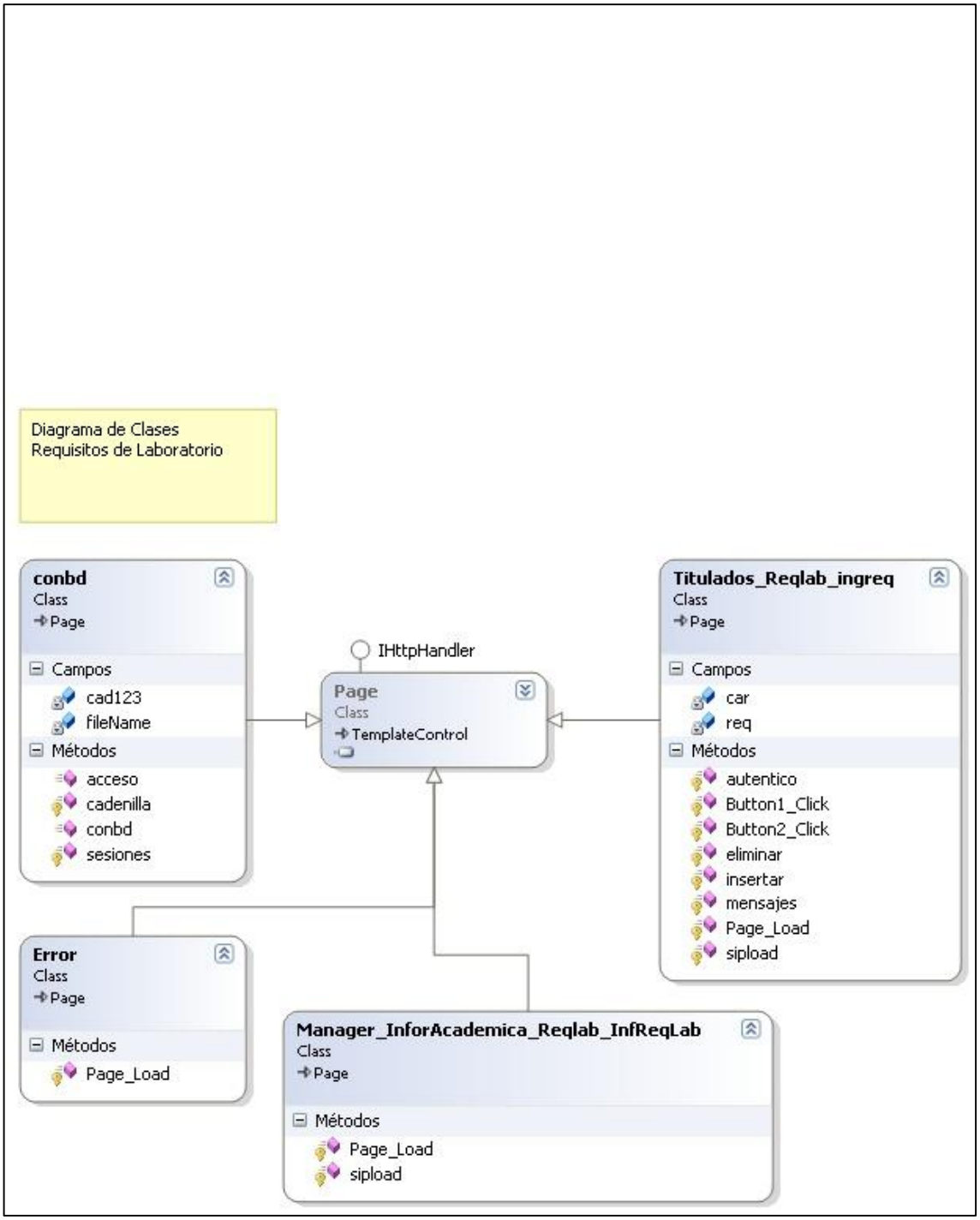


Figura 5.28 Diagrama de Clases Requisitos Laboratorio

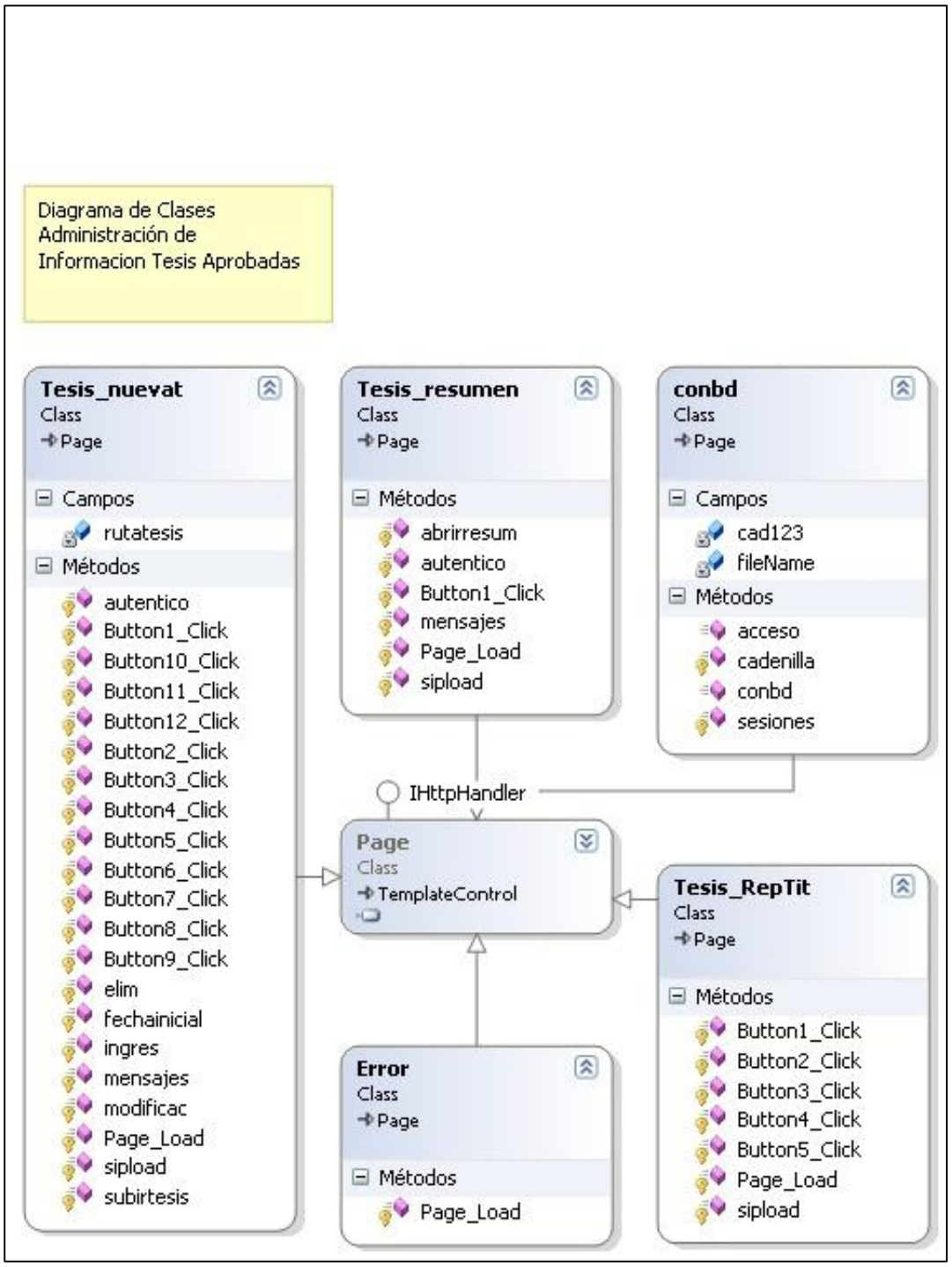


Figura 5.29 Diagrama de Clases Tesis

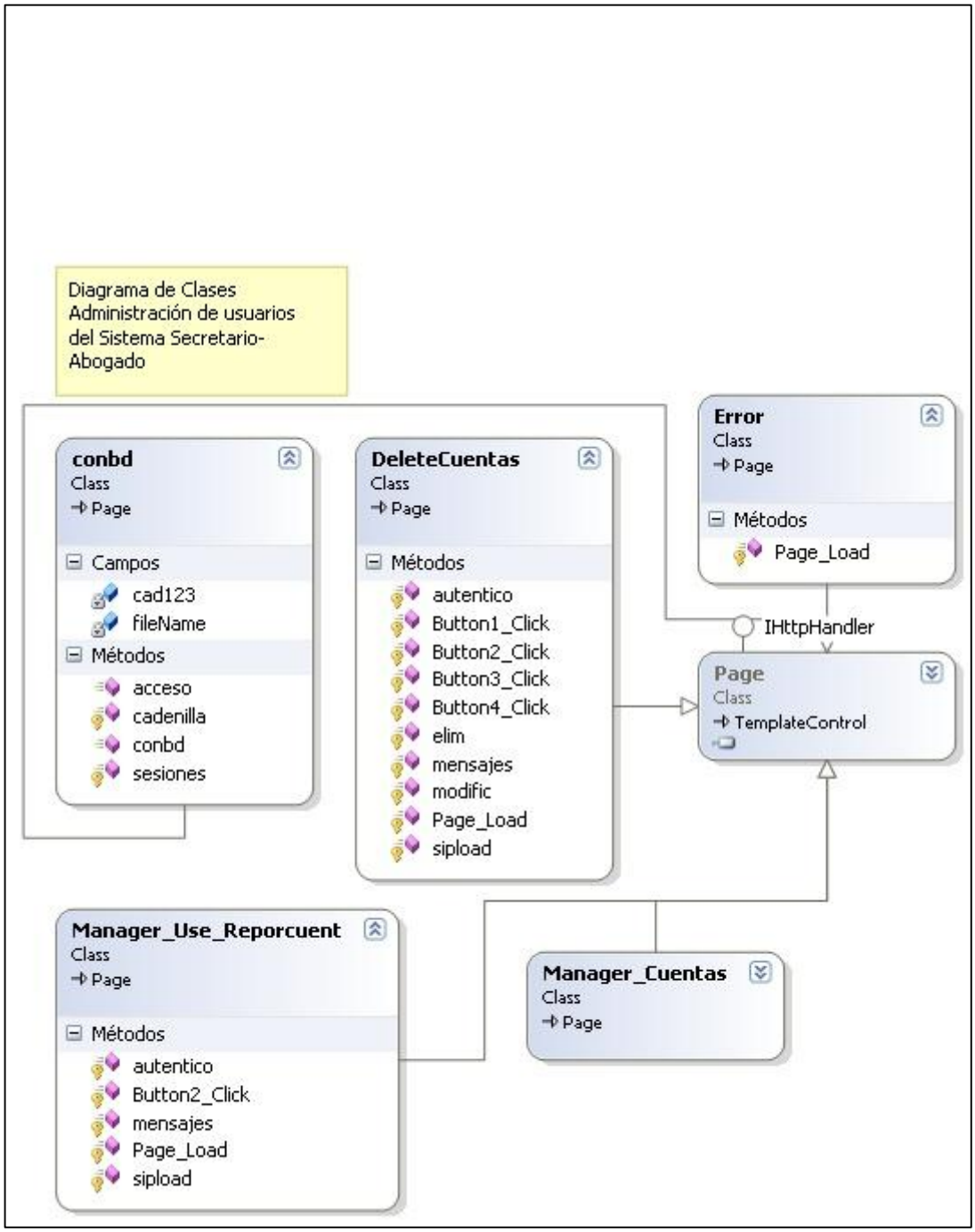
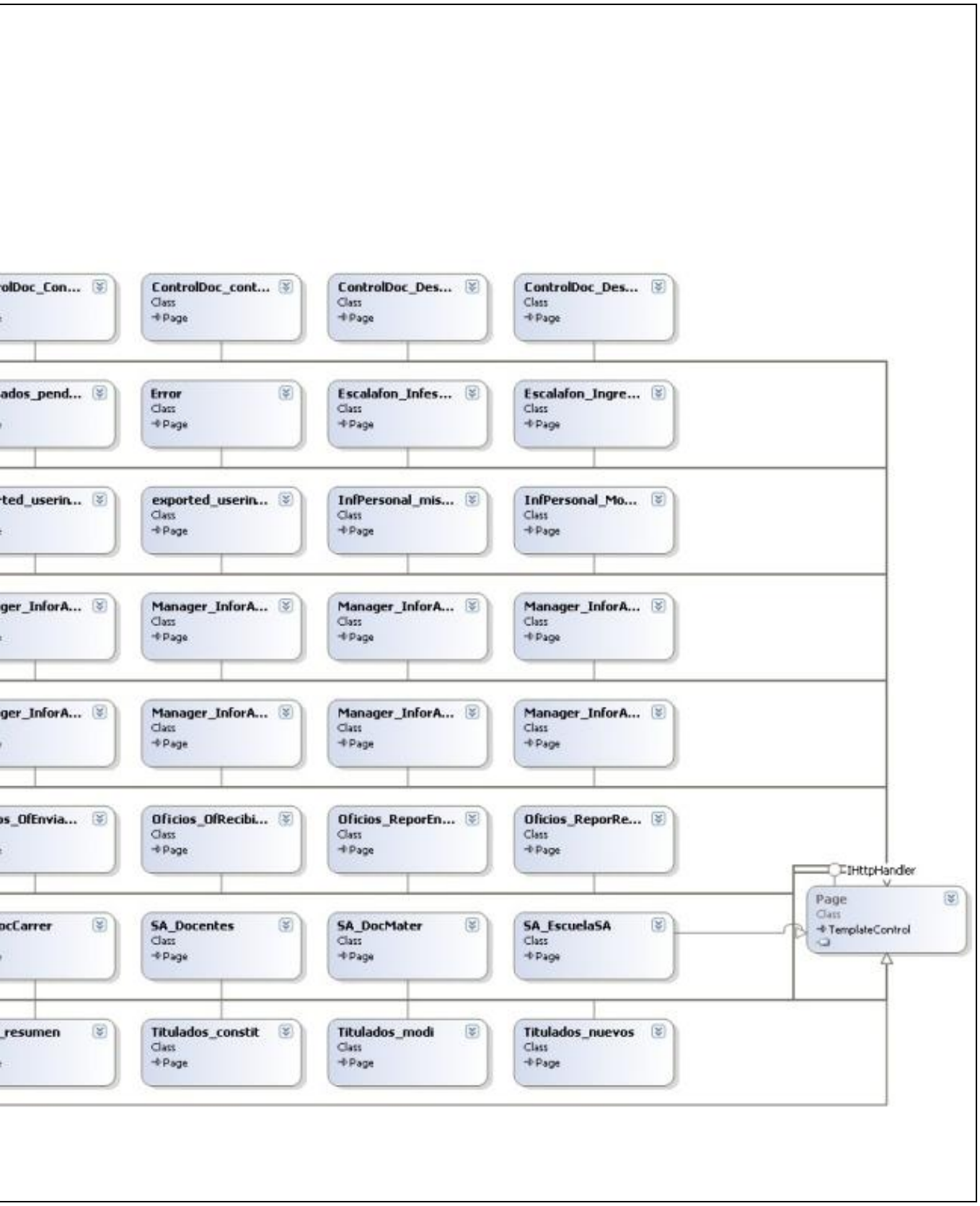


Figura 5.30 Diagrama de clases Usuarios



ministrativo Departamental Secretario-Abogado

5.5.4. Diagramas de Colaboración

A través de los mensajes de interacción colaborativos podremos determinar de forma más precisa el comportamiento de los objetos que interactúan en cada módulo del Sistema Administrativo Secretario Abogado.

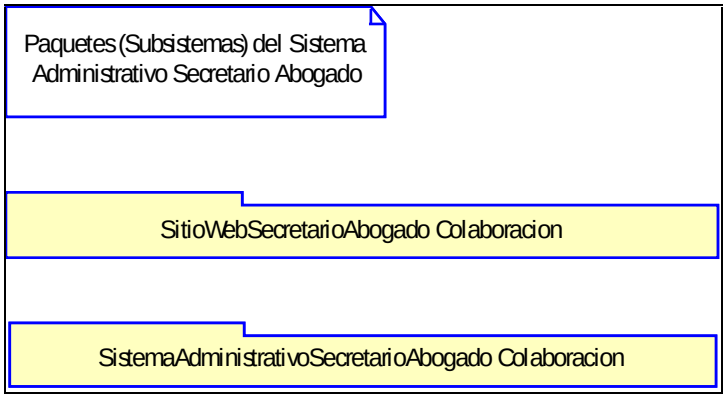
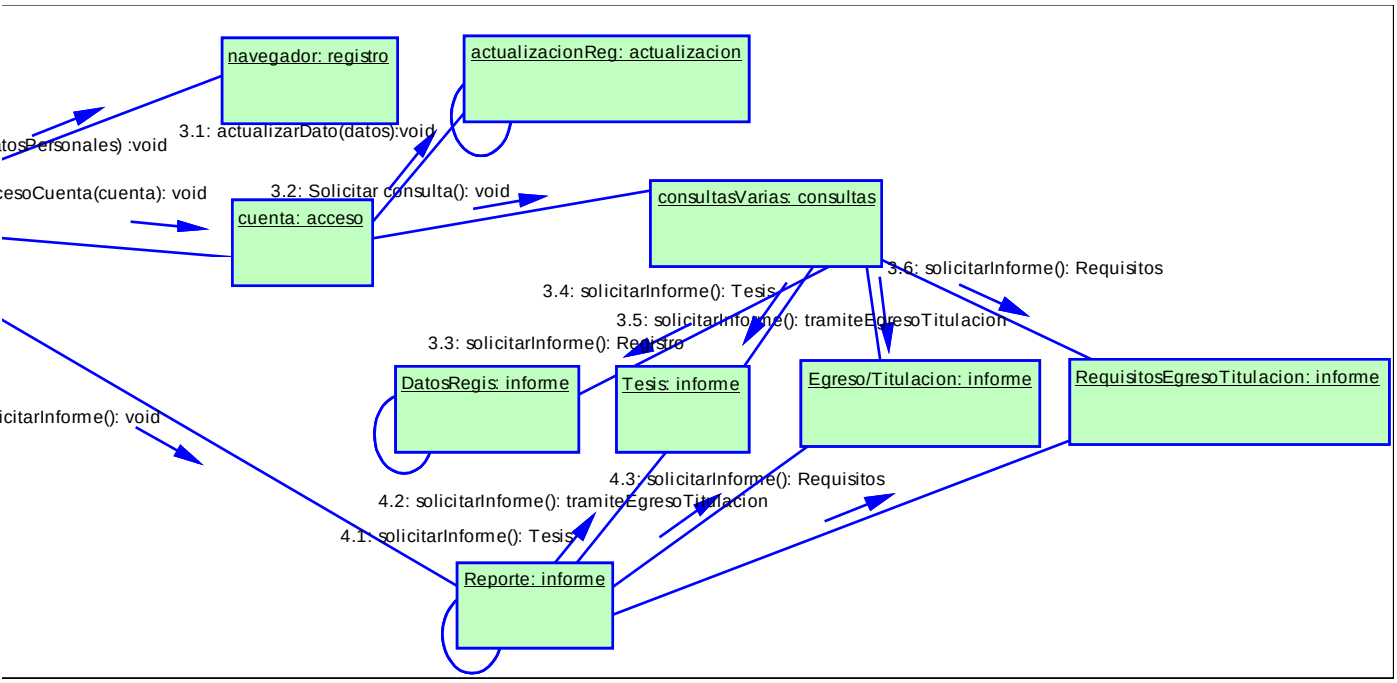


Figura 5.32 Paquetes (Subsistemas) del Sistema Administrativo Secretario Abogado



gura 5.33 Diagrama Colaboración Sitio Web Secretario Abogado

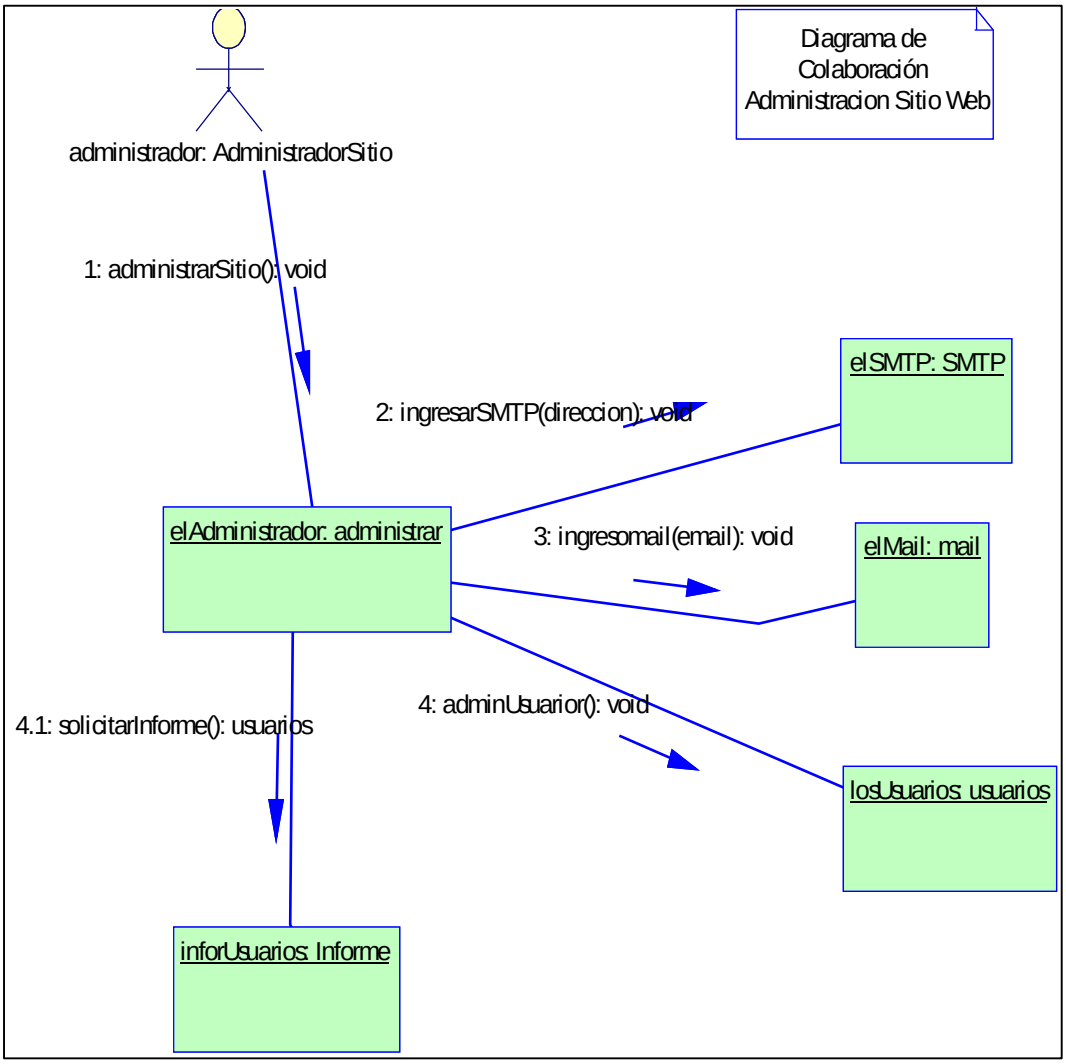


Figura 5.34 Diagrama de Colaboración Administración Sitio Web

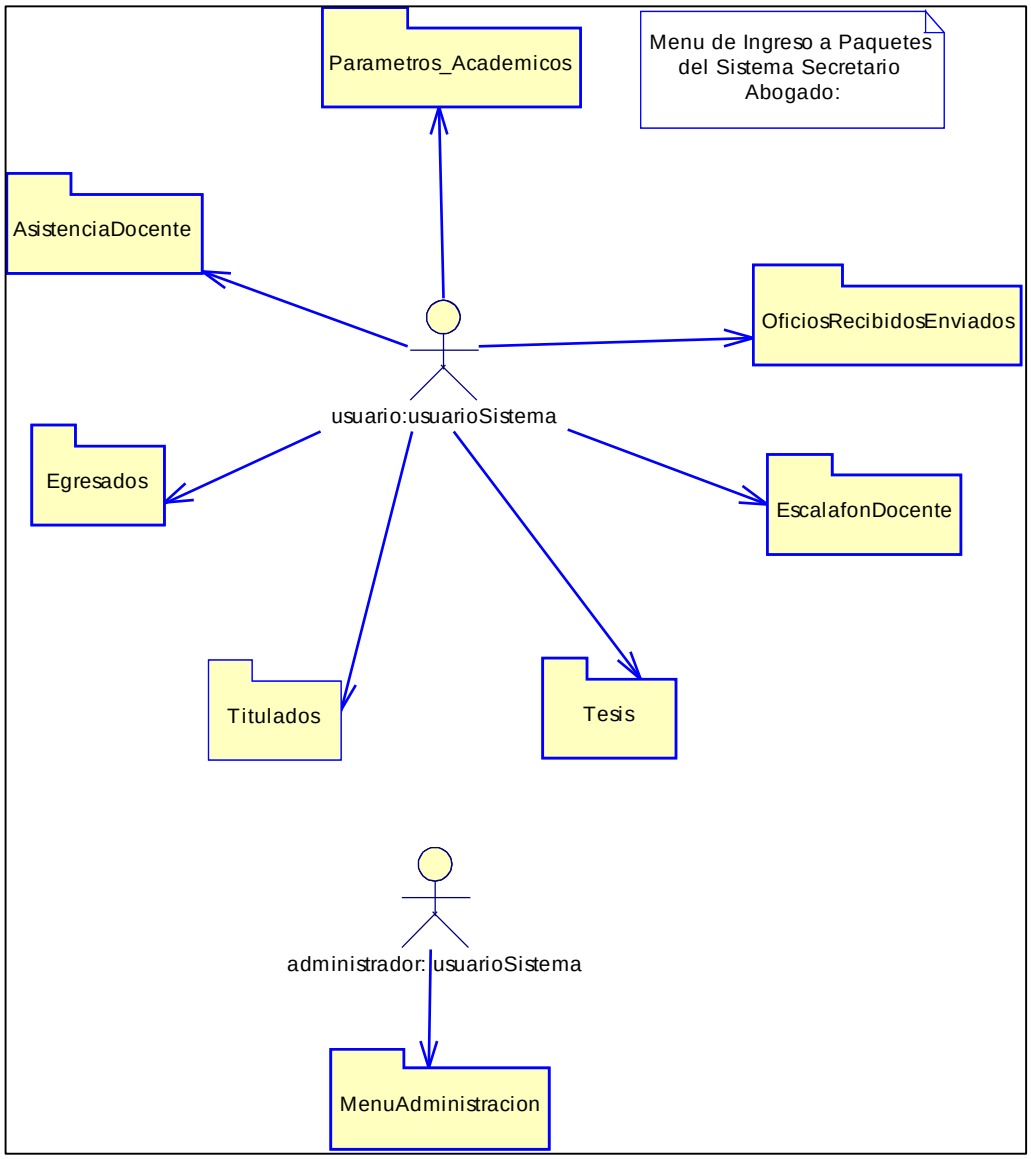


Figura 5.35 Menú de Ingreso a Paquetes del Sistema Secretario Abogado

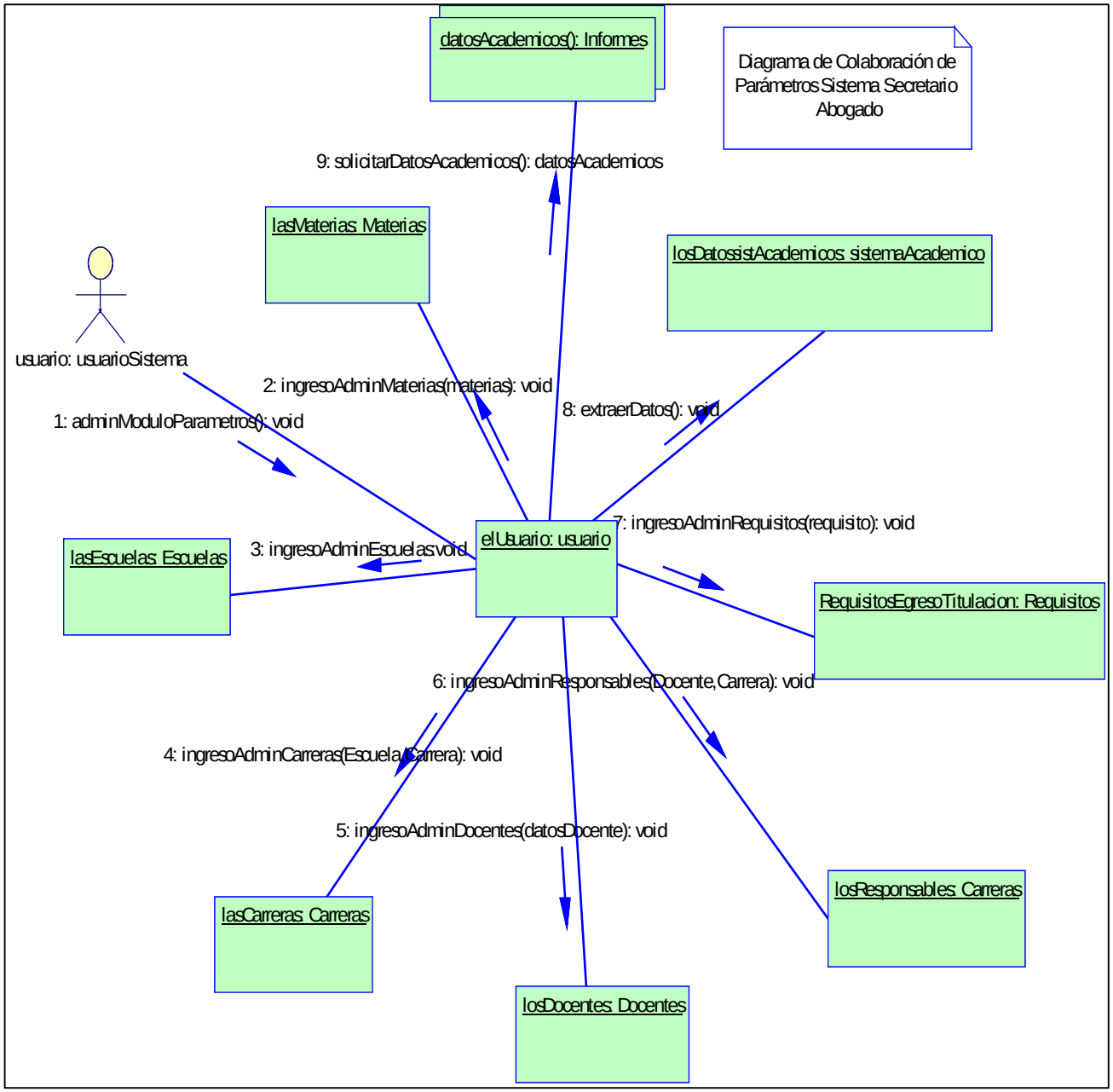


Figura 5.36 Diagrama de Colaboración de Parámetros Sistema Secretario Abogado

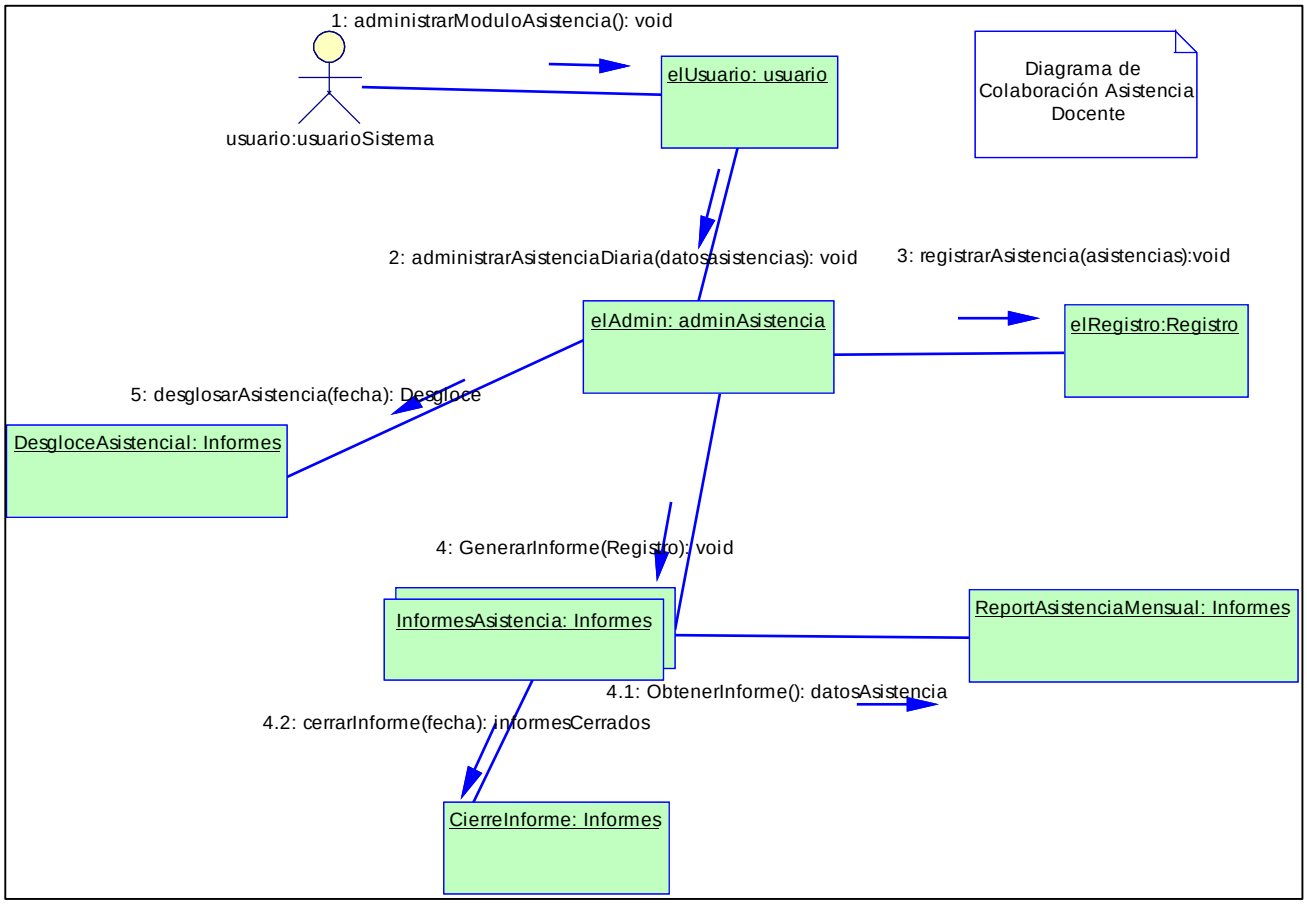


Figura 5.37 Diagrama de Colaboración Asistencia Docente

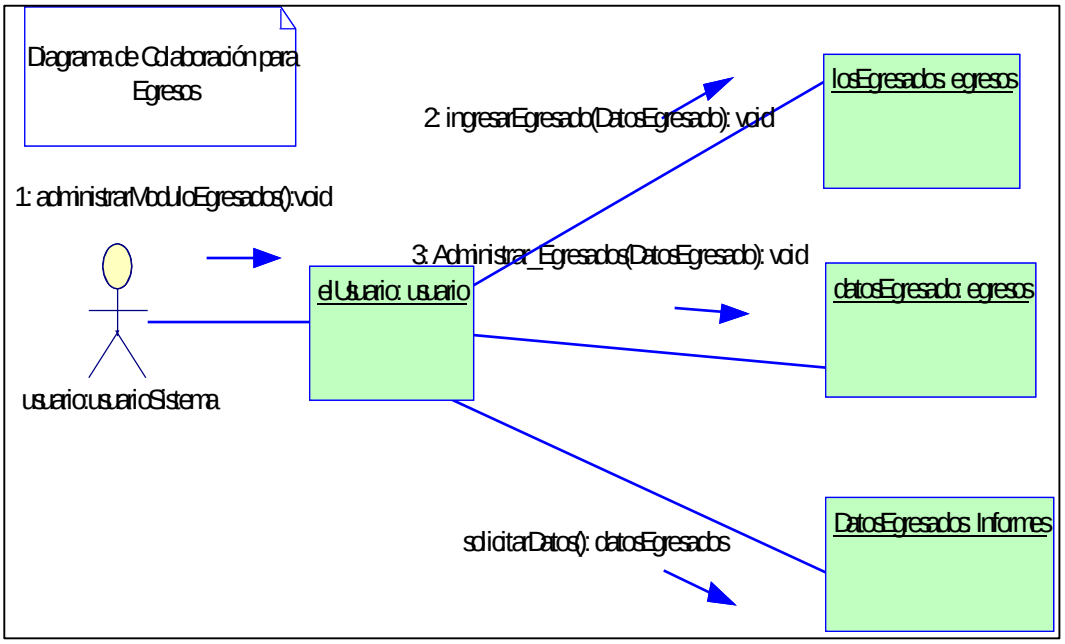


Figura 5.38 Diagrama de Colaboración para Egresos

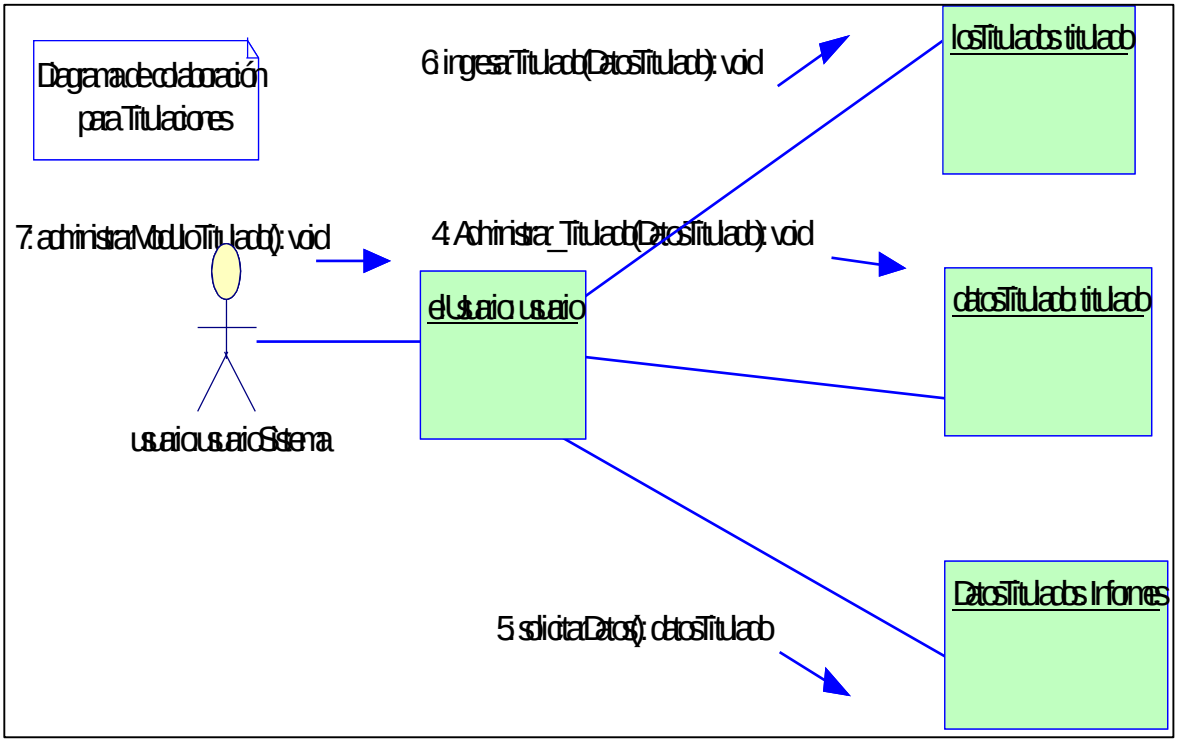


Figura 5.39 Diagrama de Colaboración para Titulaciones

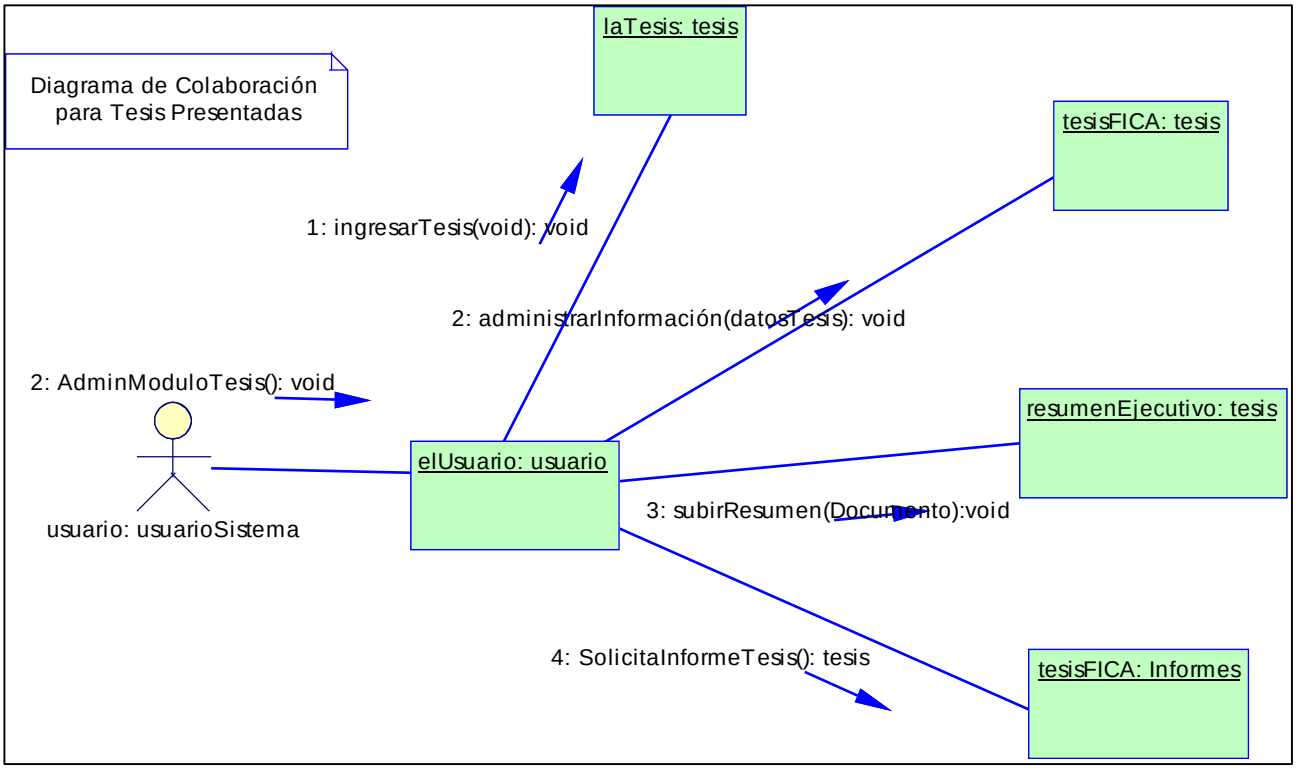


Figura 5.40 Diagrama de Colaboración para Tesis Presentadas

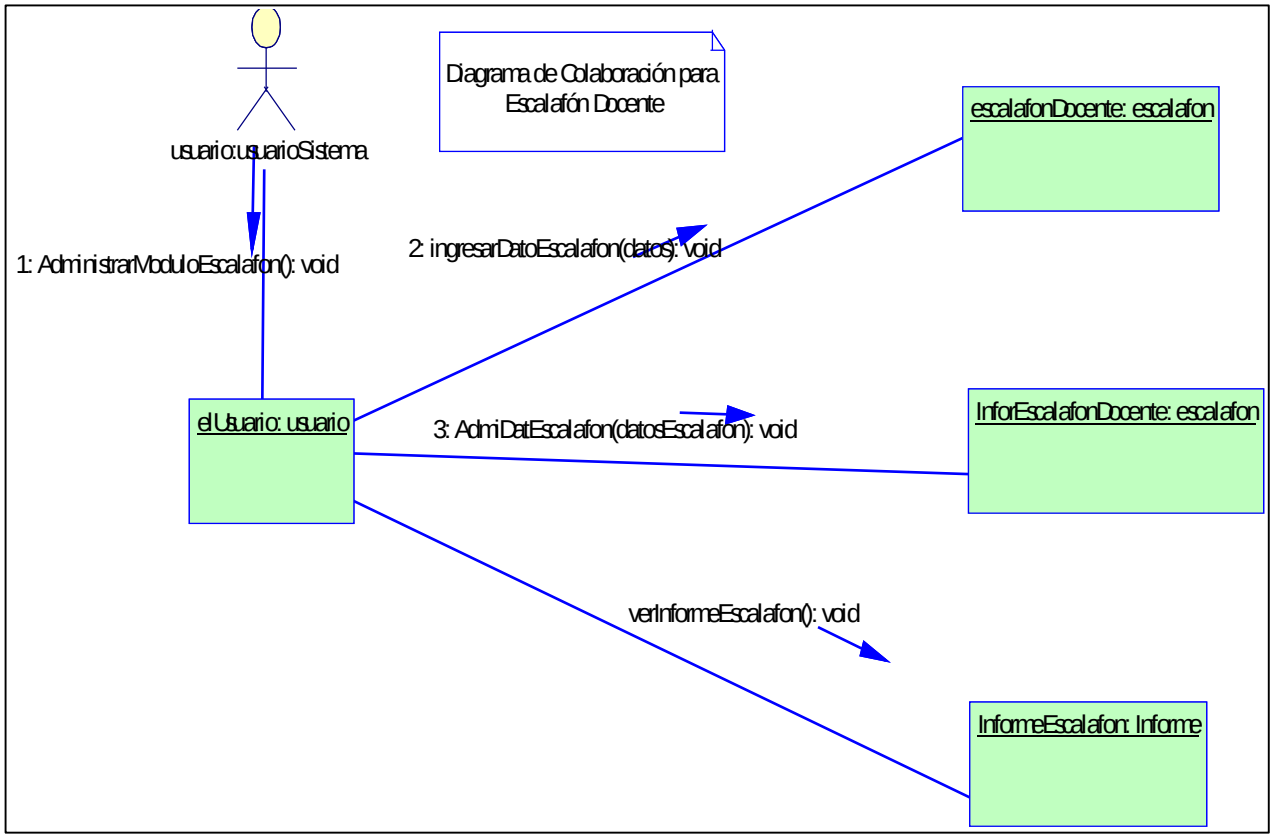


Figura 5.41 Diagrama de Colaboración para Escalafón Docente

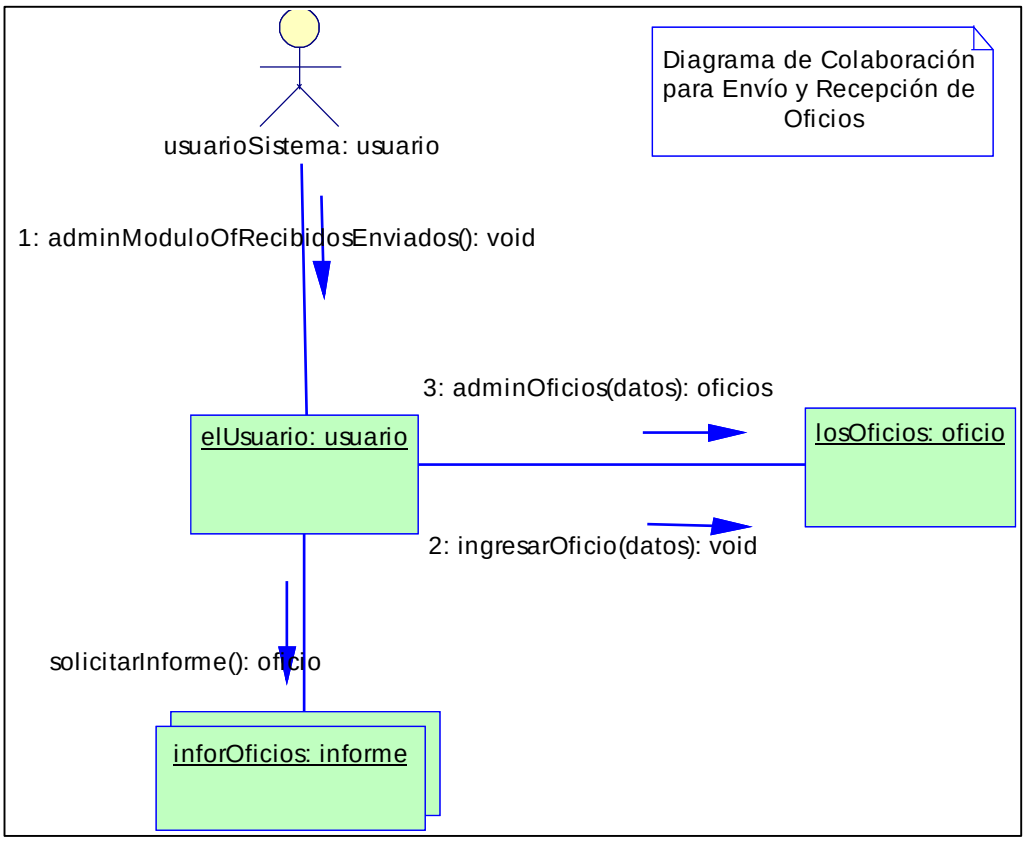
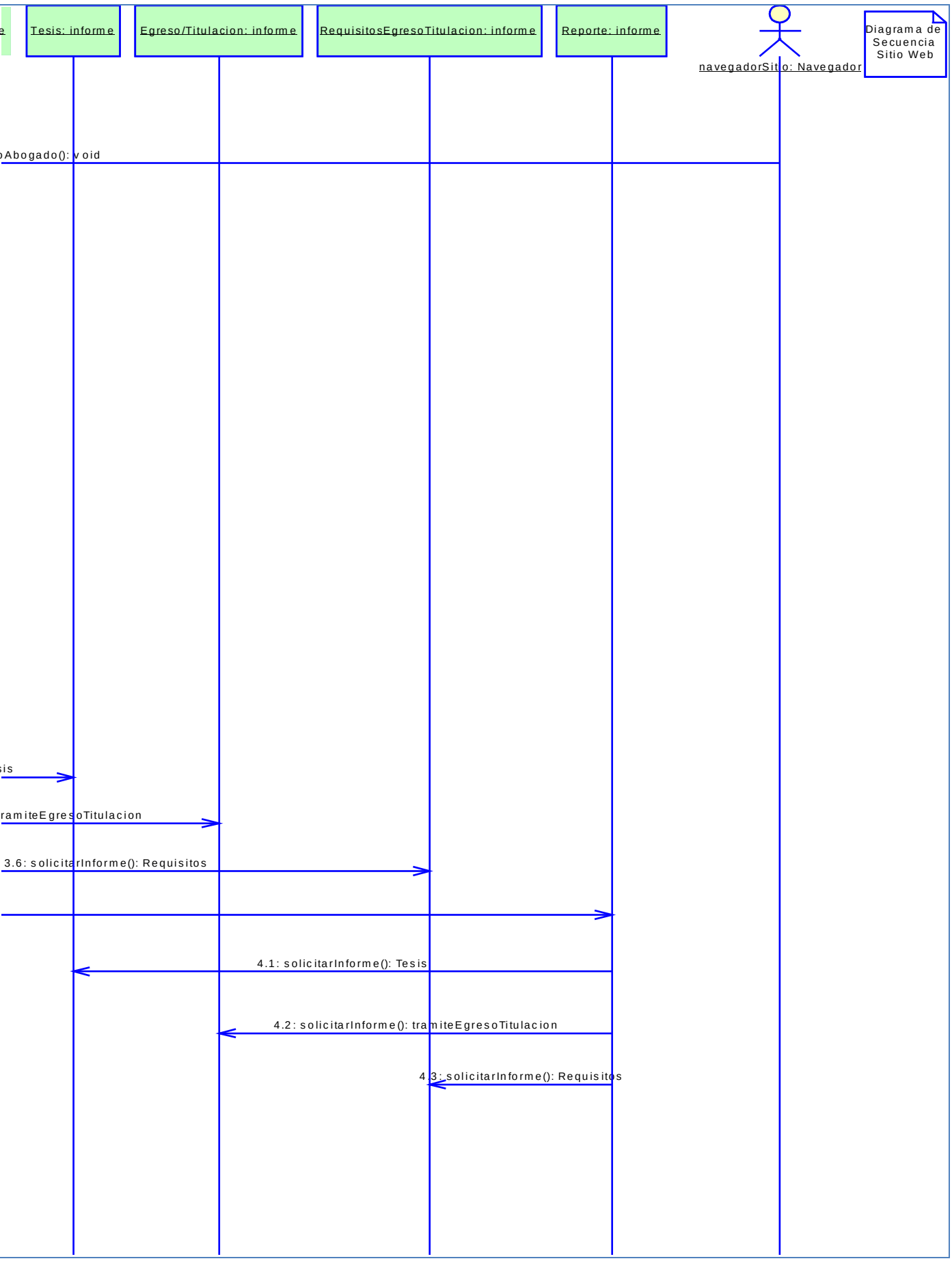


Figura 5.42 Diagrama de Colaboración para Envío y Recepción de Oficios

5.5.5. Diagramas de Secuencia

Los Diagramas de Secuencia nos permitirán conocer más detalladamente la manera en que interactuarán los objetos que componen el Sistema Administrativo Secretario Abogado, en un orden cronológico y de forma jerárquica.



e Secuencia Sitio Web

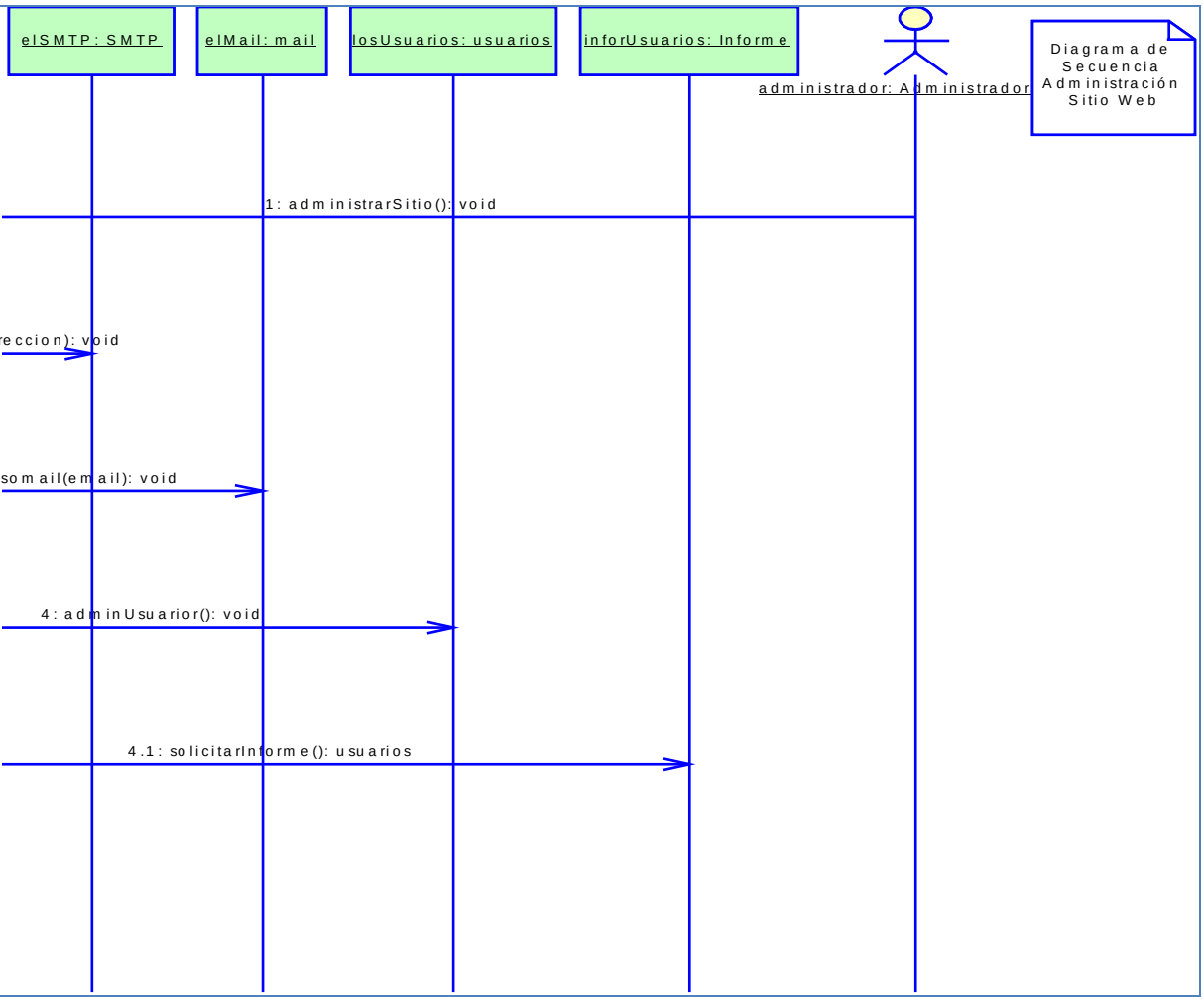


Figura 5.44 Diagrama de Secuencia Administración Sitio Web

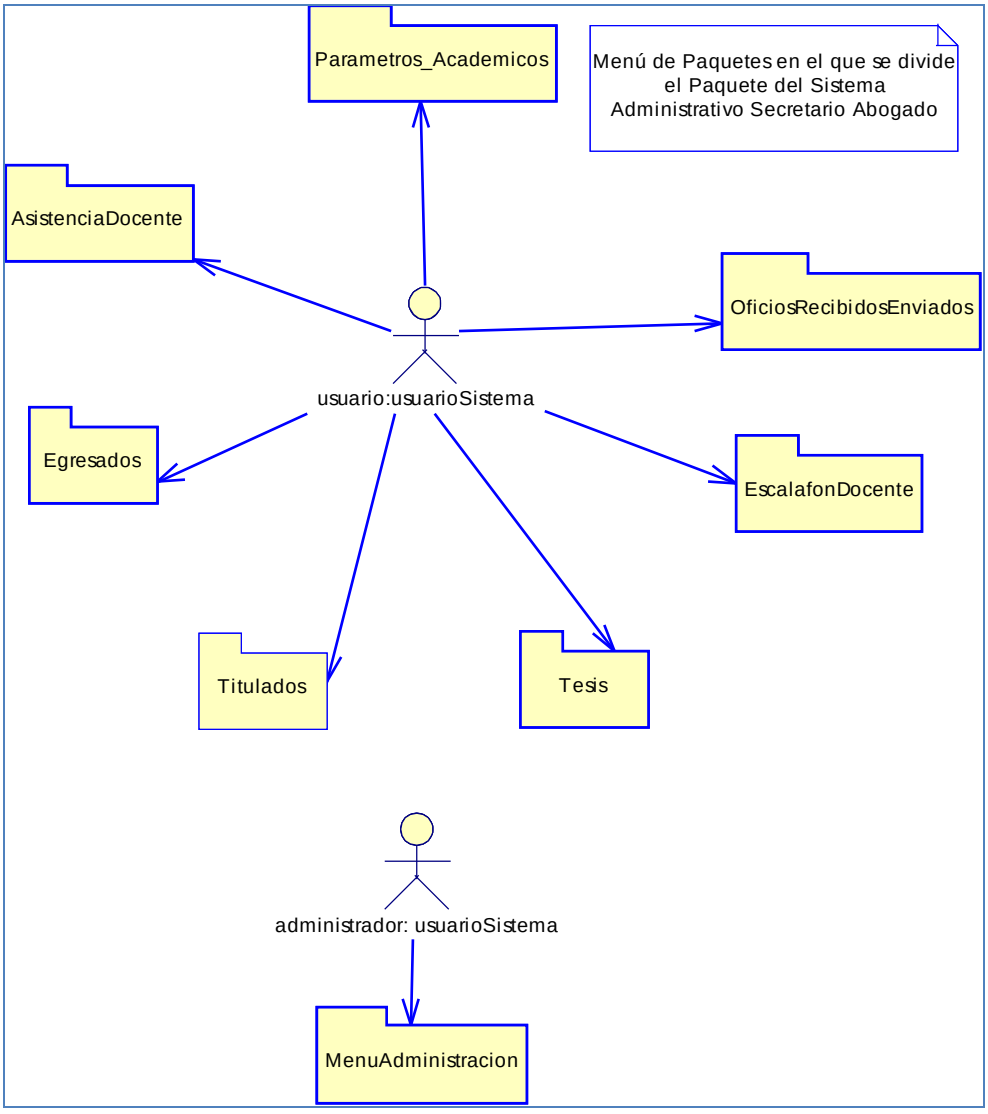
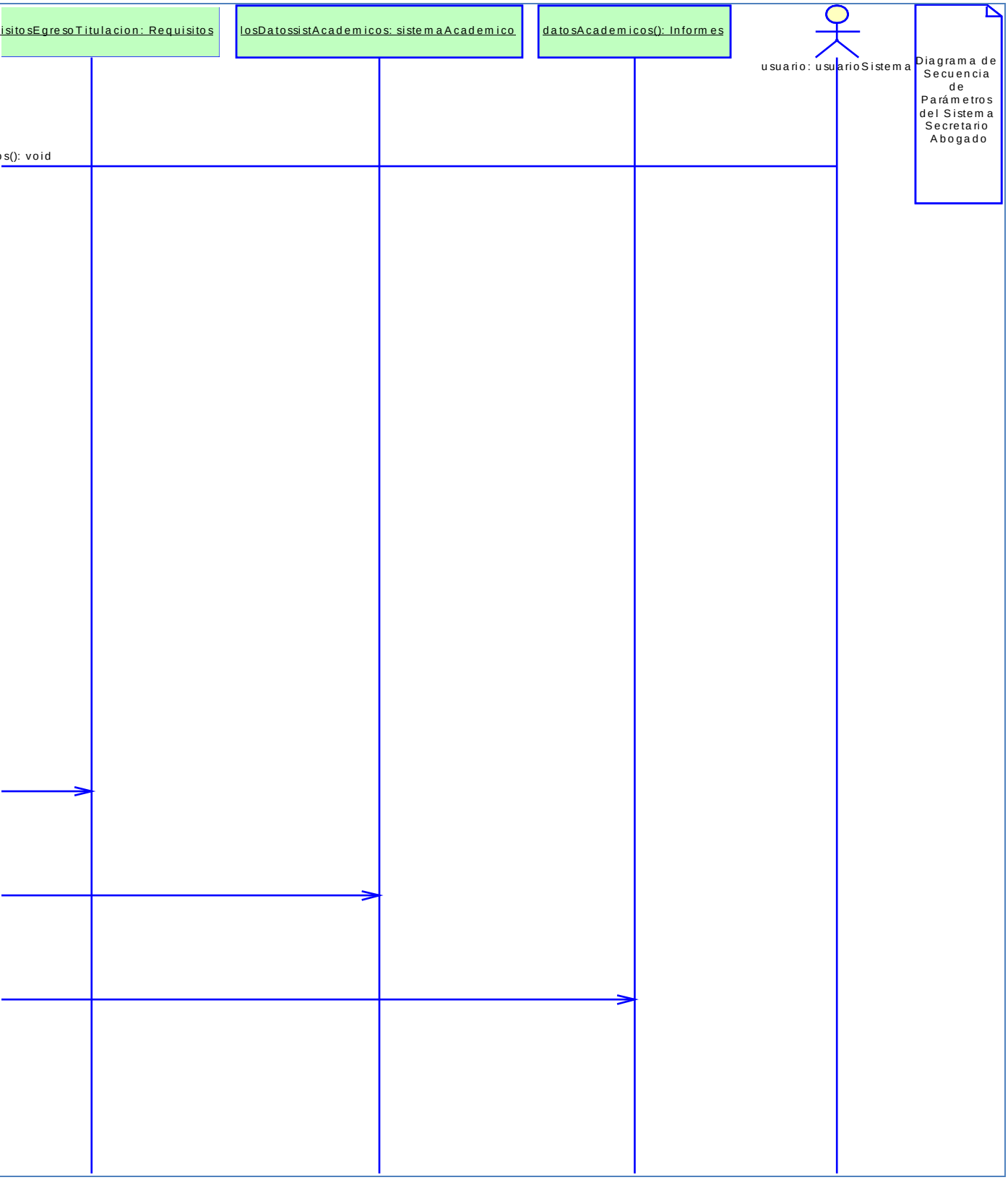
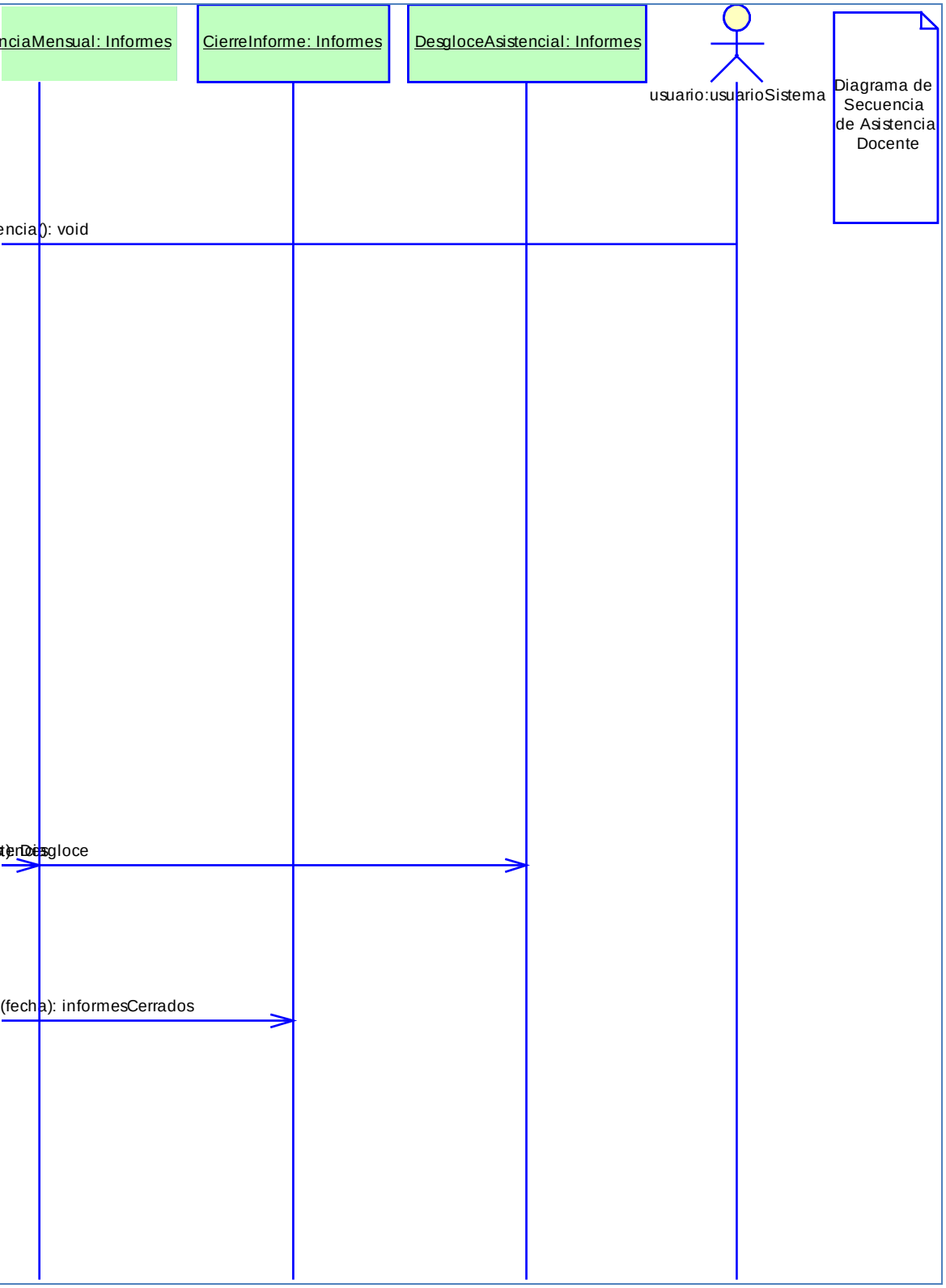


Figura 5.45 Menú de Paquetes en el que se divide el Paquete del Sistema Administrativo Secretario Abogado



rámetros del Sistema Secretario Abogado



uencia de Asistencia Docente

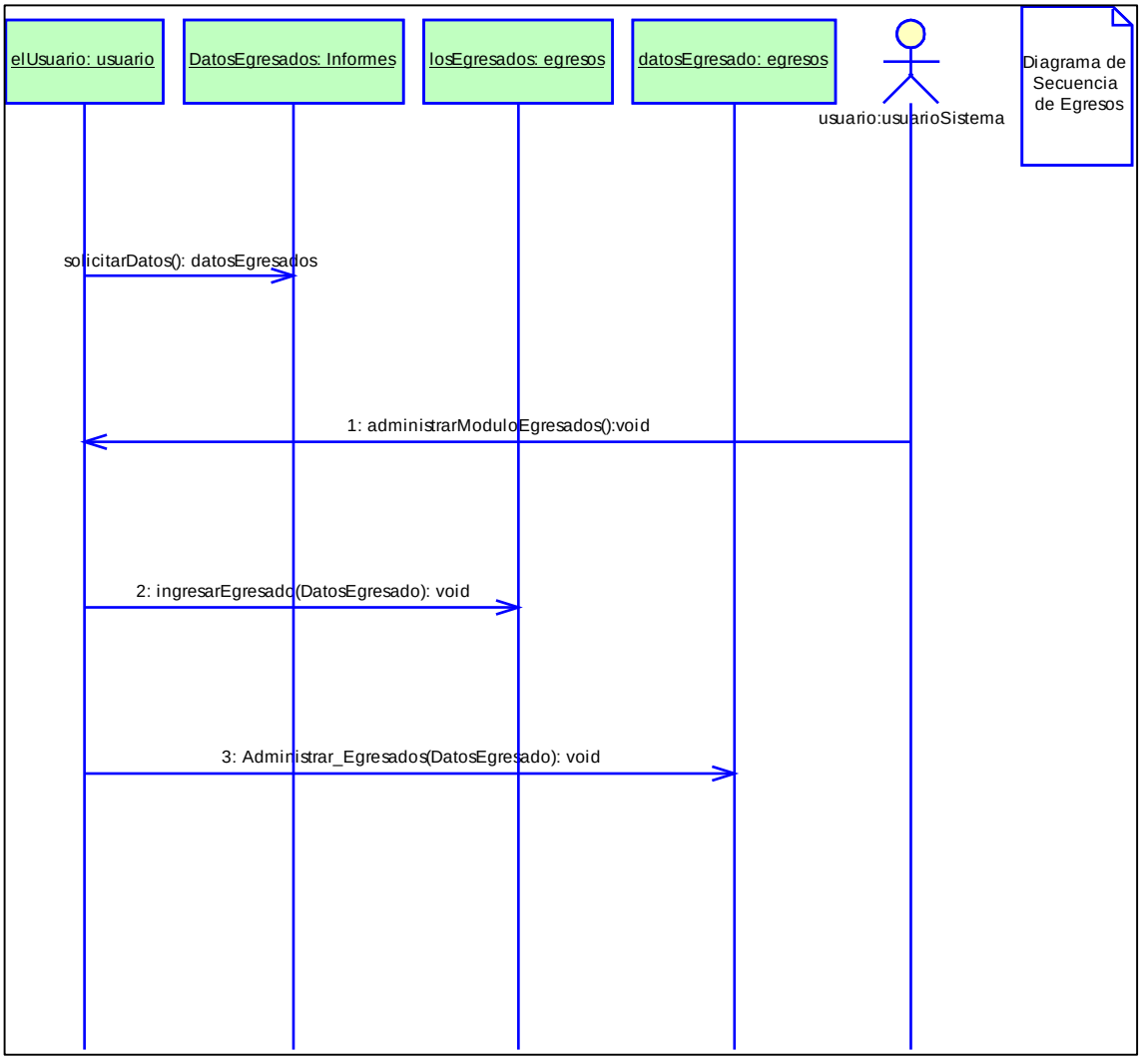


Figura 5.48 Diagrama de Secuencia de Egresos

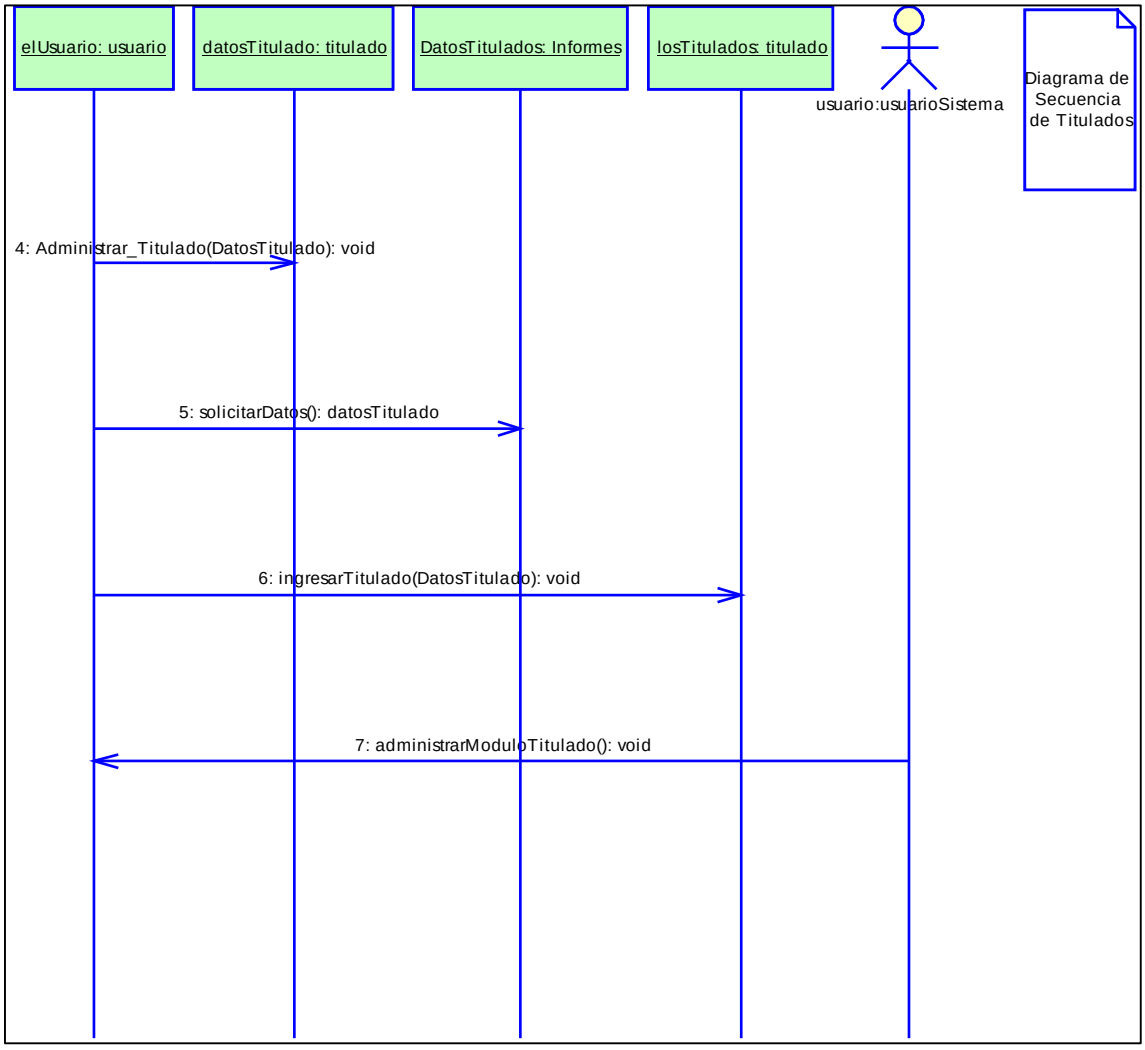


Figura 5.49 Diagrama de Secuencia de Titulados

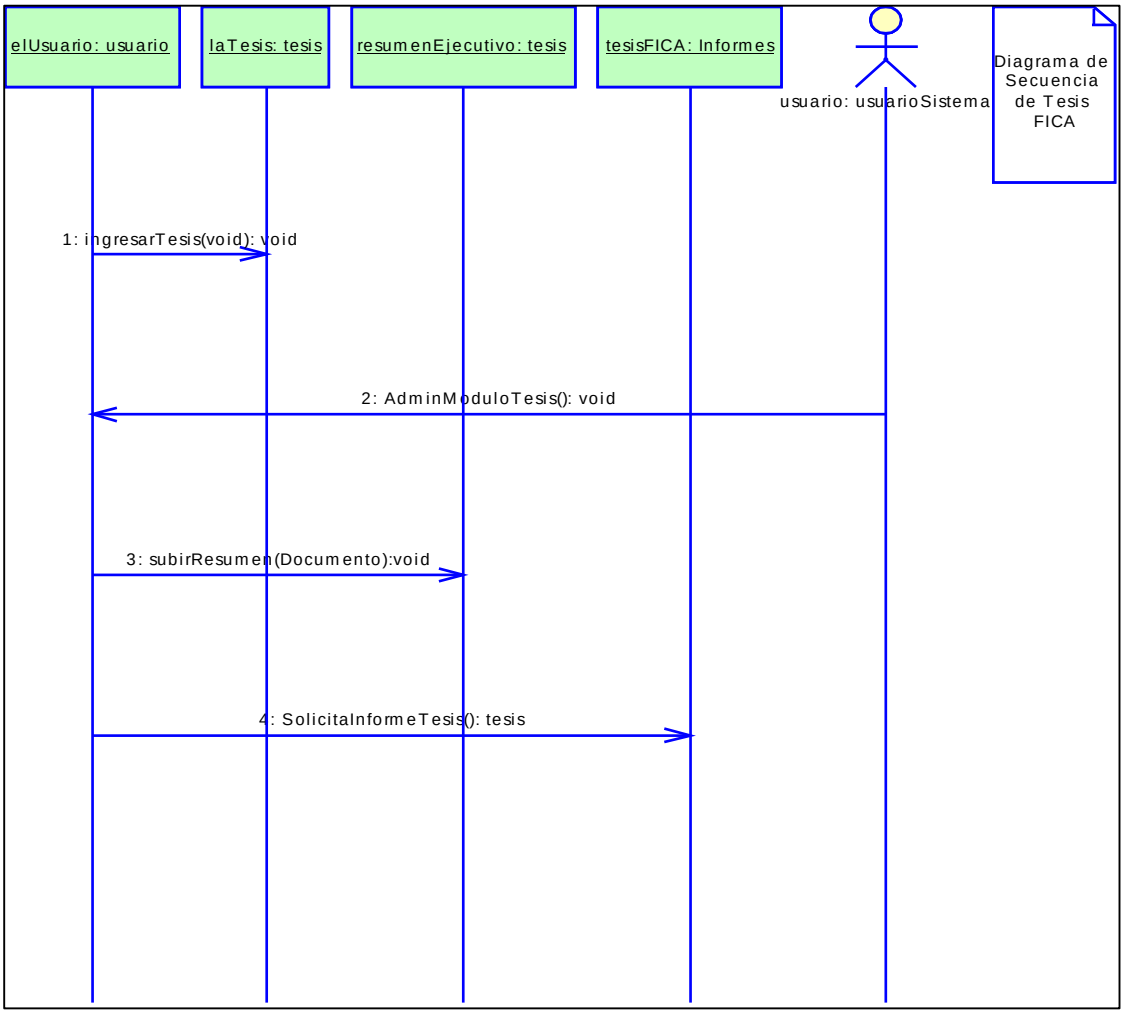


Figura 5.50 Diagrama de Secuencia de Tesis FICA

ATIVO WEB

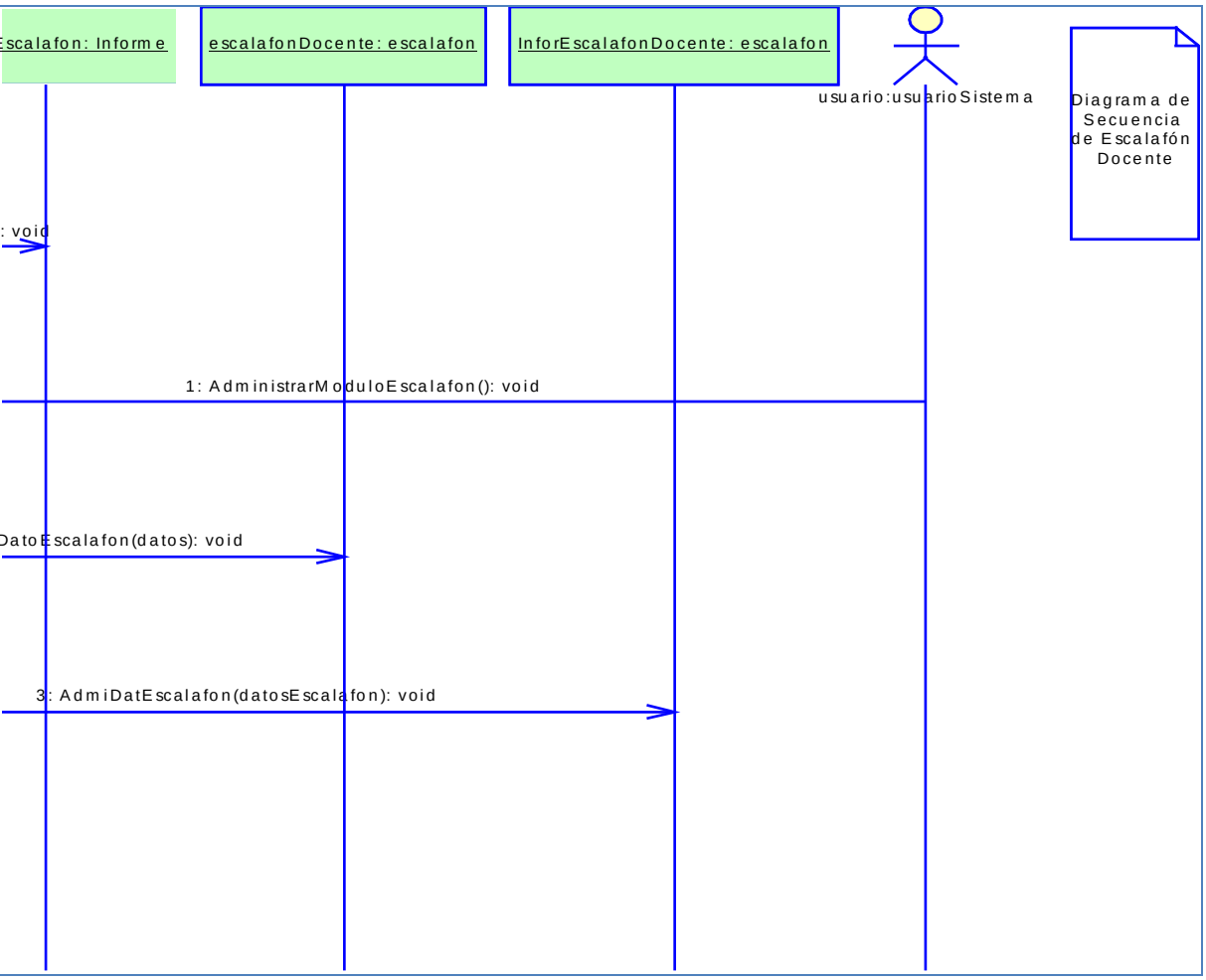


Figura 5.51 Diagrama de Secuencia de Escalafón Docente

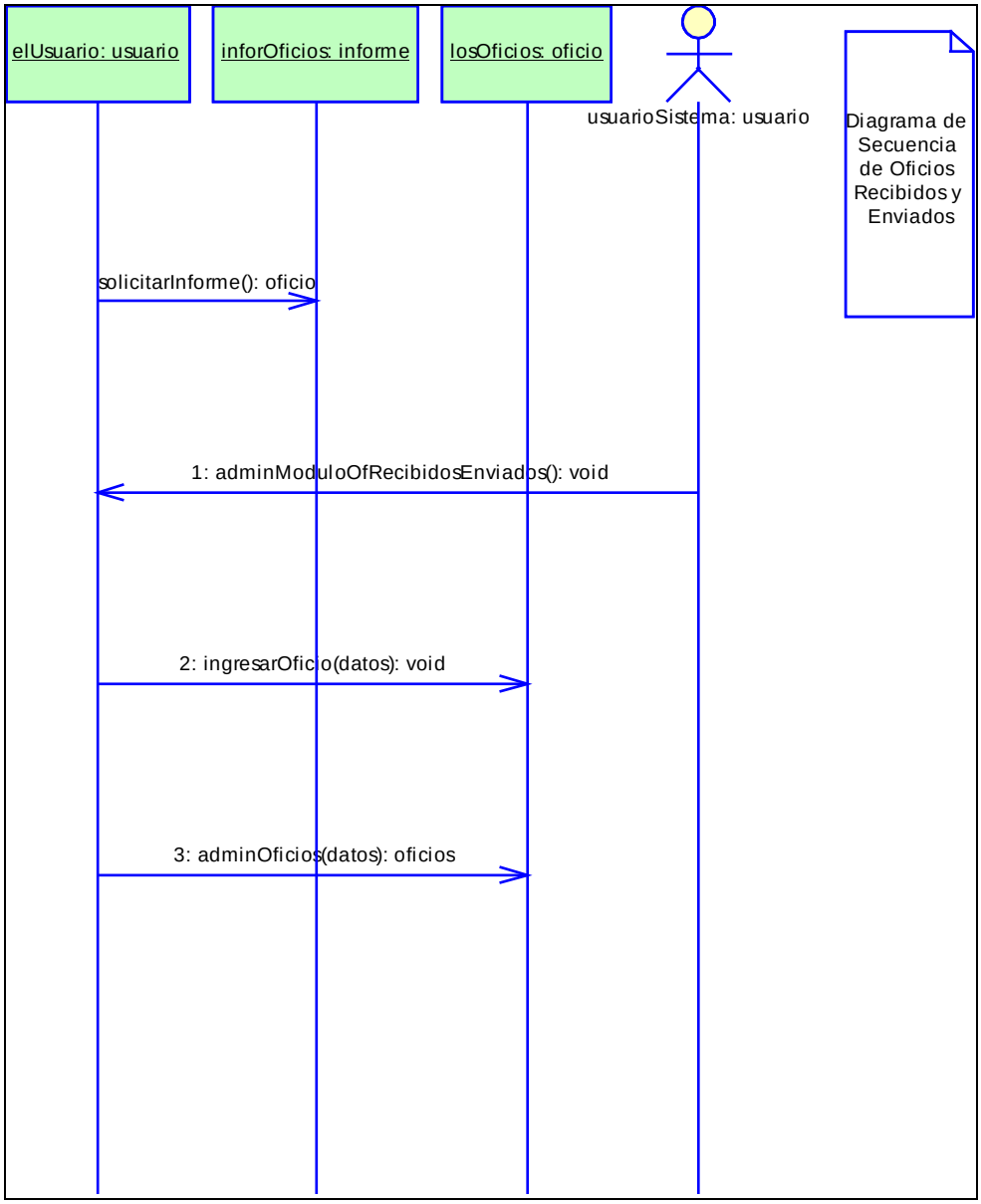
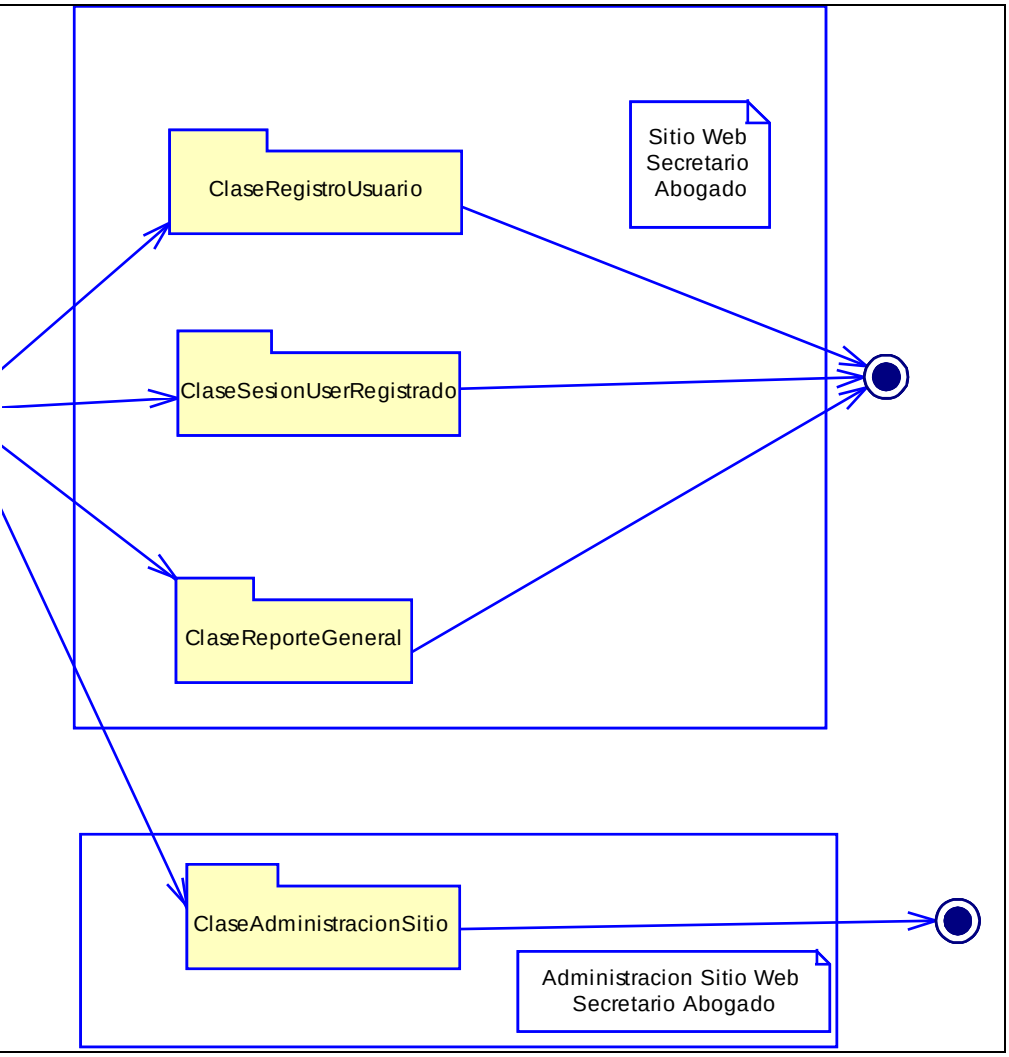


Figura 5.52 Diagrama de Secuencia de Oficios Recibidos y Enviados

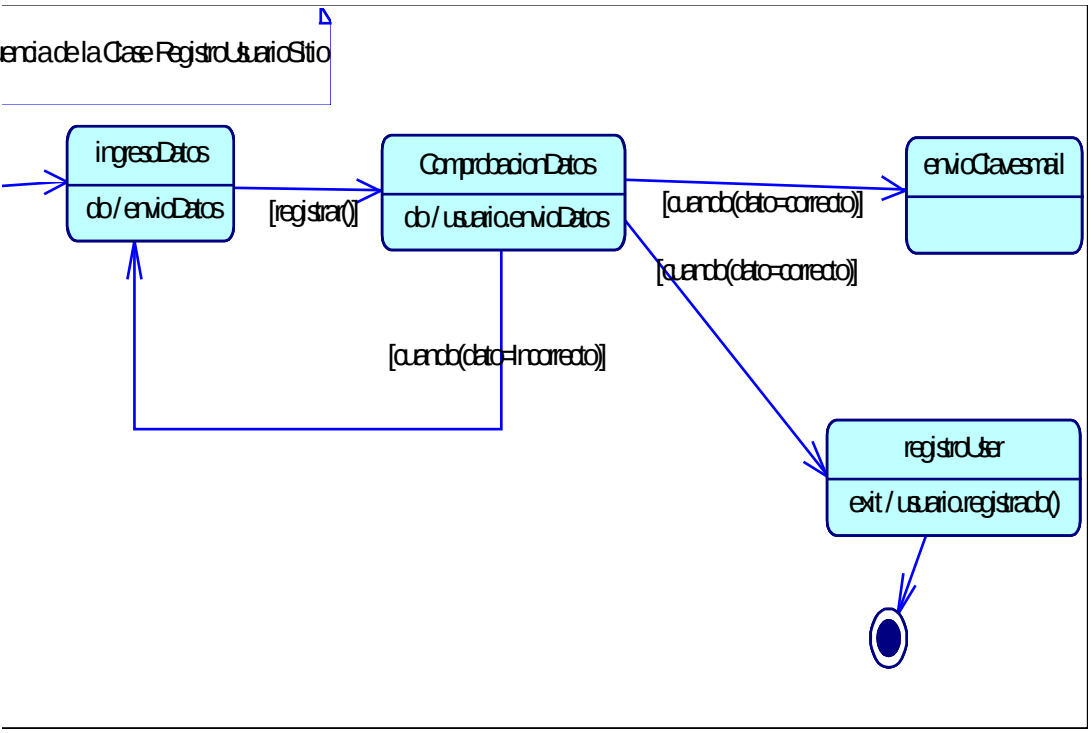
5.5.6. Diagramas de Estado

Mediante los Diagramas de Estado modelaremos las diferentes acciones que tomarán los objetos de nuestro Sistema al recibir mensajes determinados. Estos diagramas mejoran la comprensión de funcionamiento interno de los módulos del sistema.

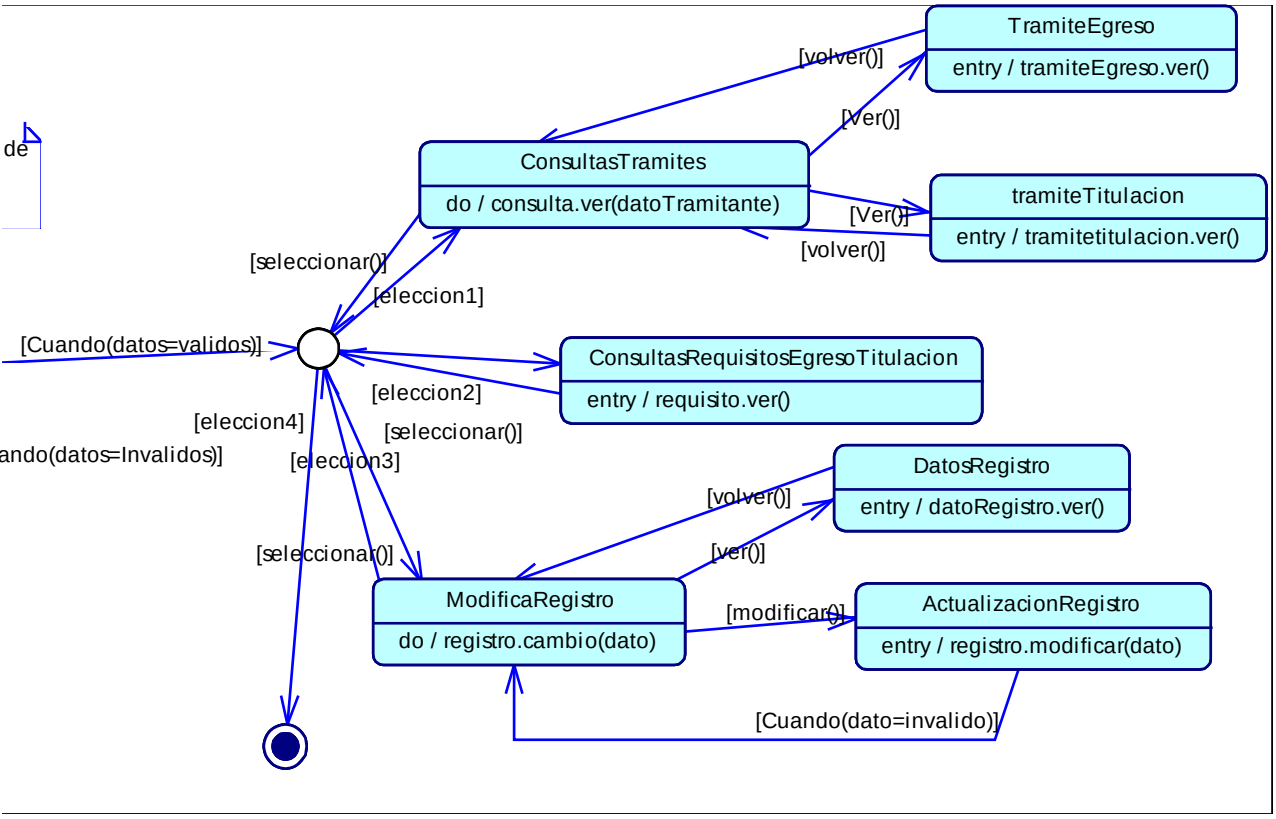
ATIVO WEB



3 Paquetes en los que se divide el paquete Sitio Web Secretario Abogado



ra 5.54 Diagramas de Secuencia de la Clase RegistroUsuarioSitio



ra 5.55 Diagramas de Secuencia de la Clase SesionUserRegistrado

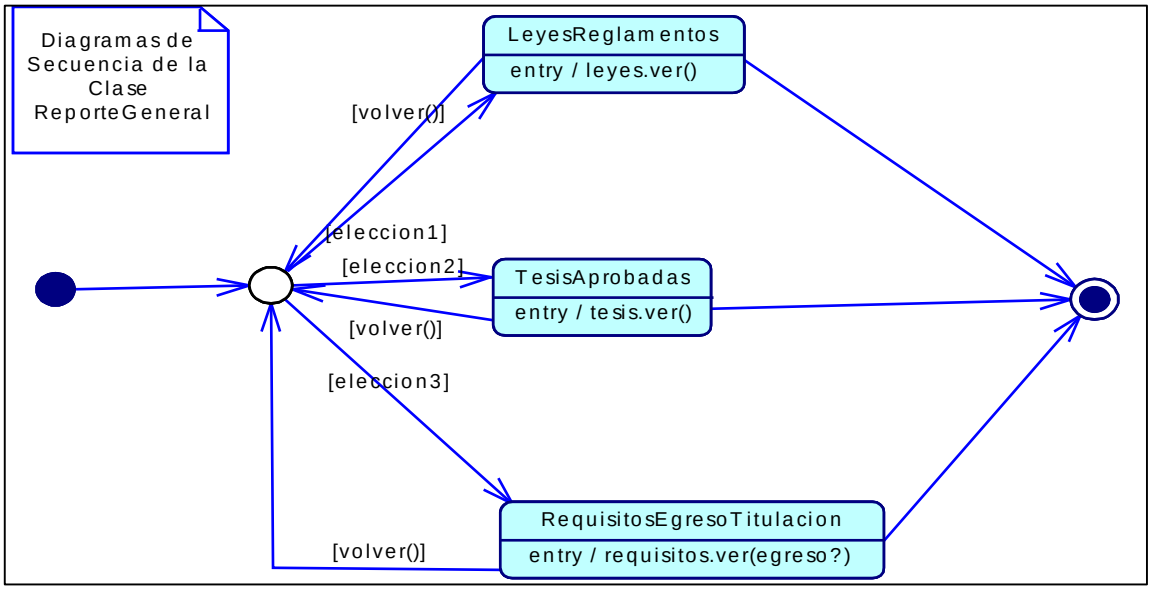
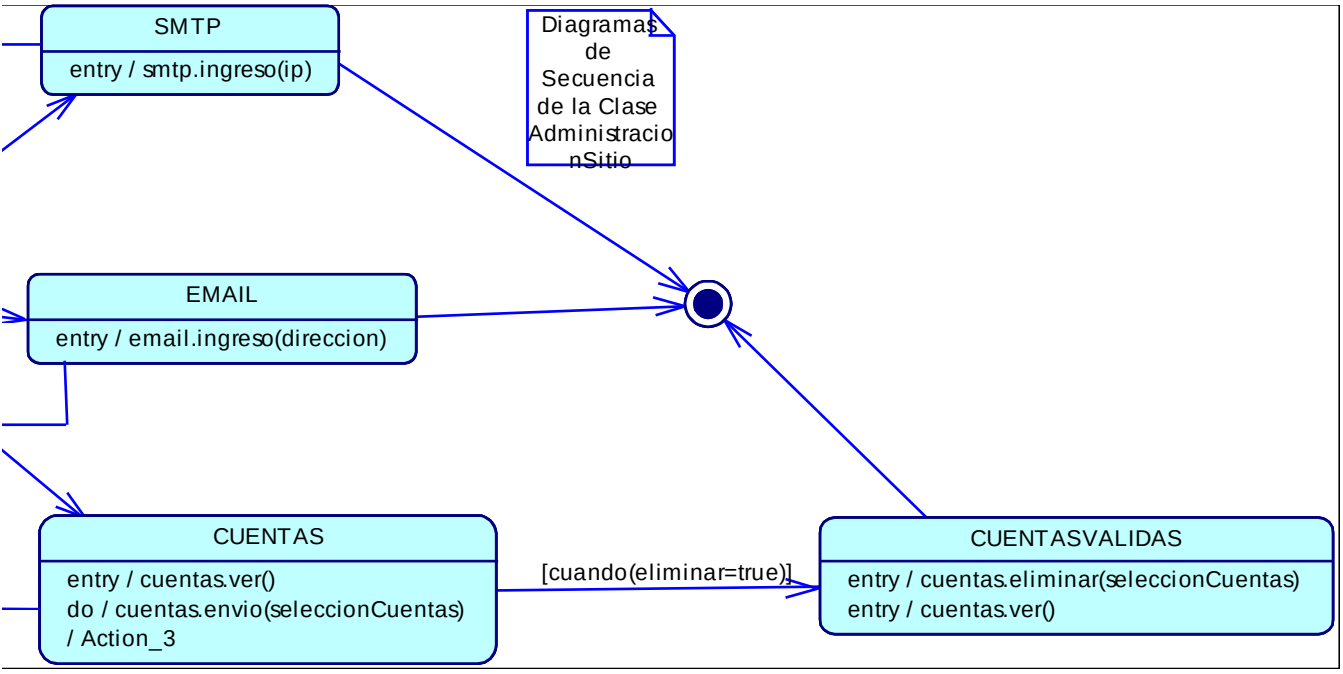


Figura 5.56 Diagramas de Secuencia de la Clase ReporteGeneral



ura 5.57 Diagramas de Secuencia de la Clase AdministracionSitio

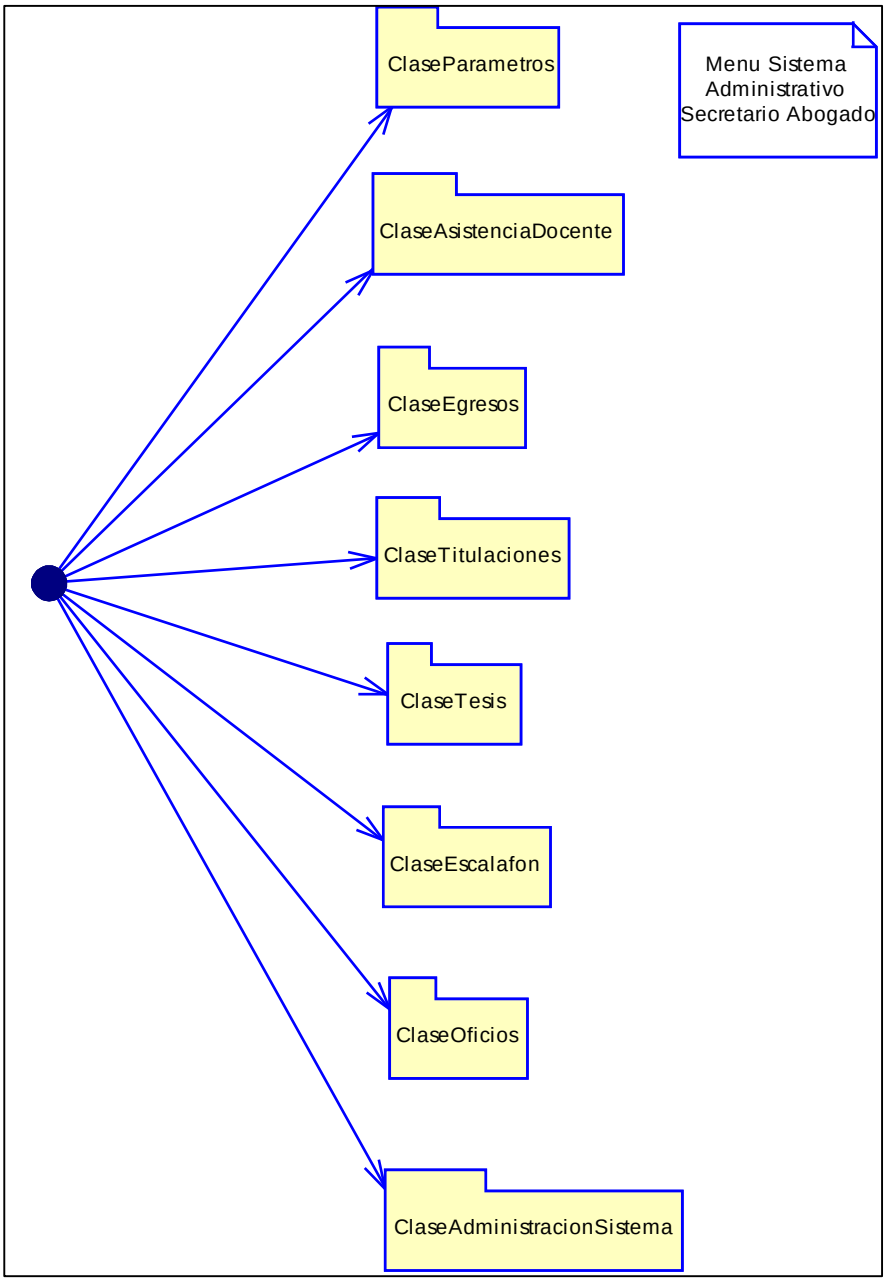


Figura 5.58 División en Paquetes del Paquete SistemaAdministrativoSecretarioAbogado

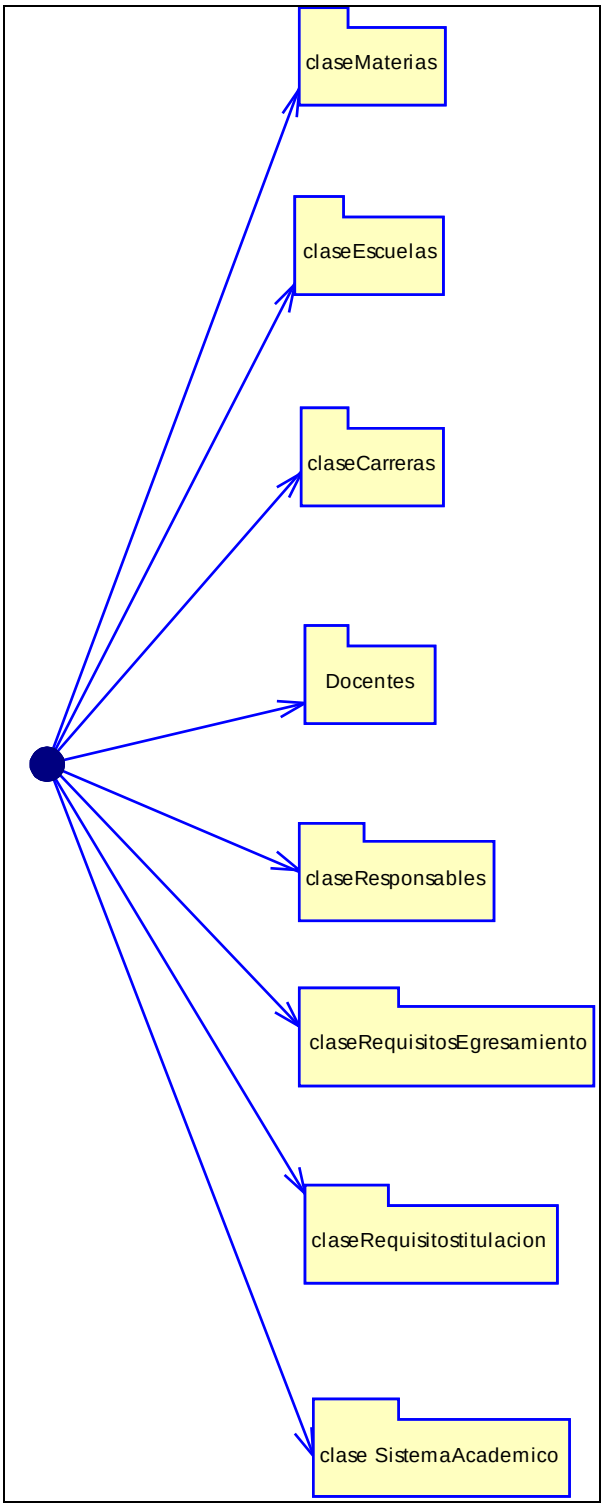


Figura 5.59 División del Paquete ClaseParametros

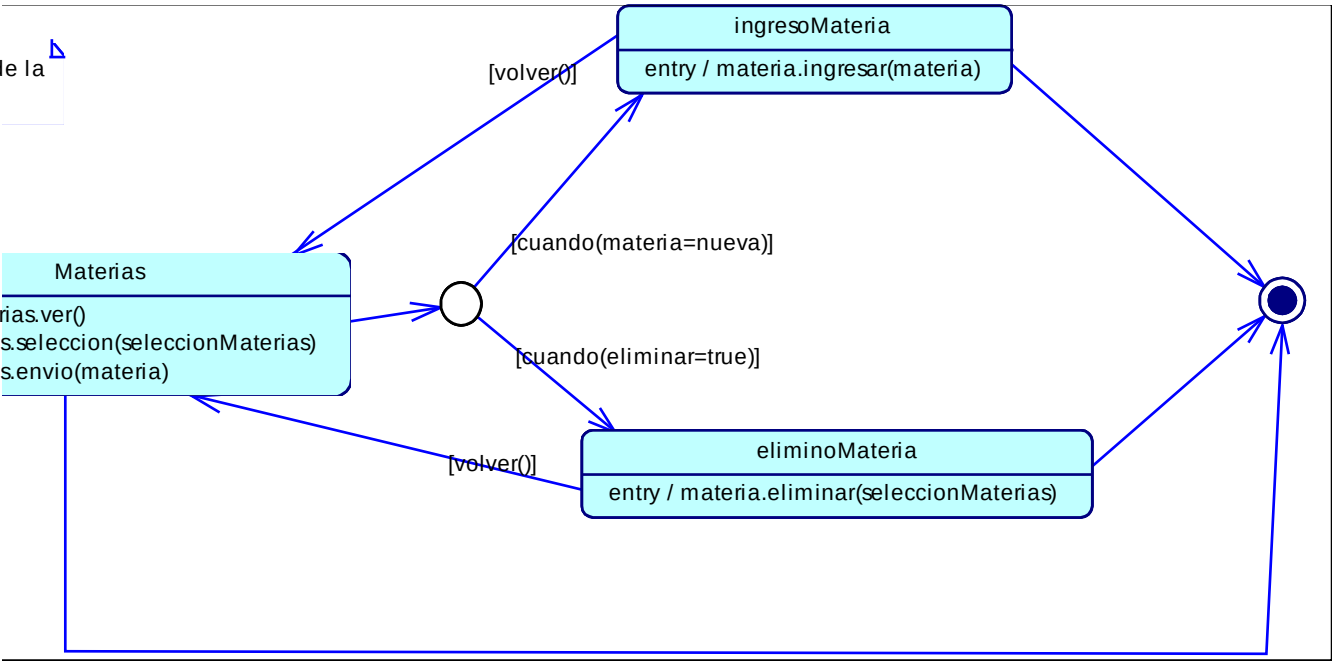


Figura 5.60 Diagrama de Estado de la Clase Materias

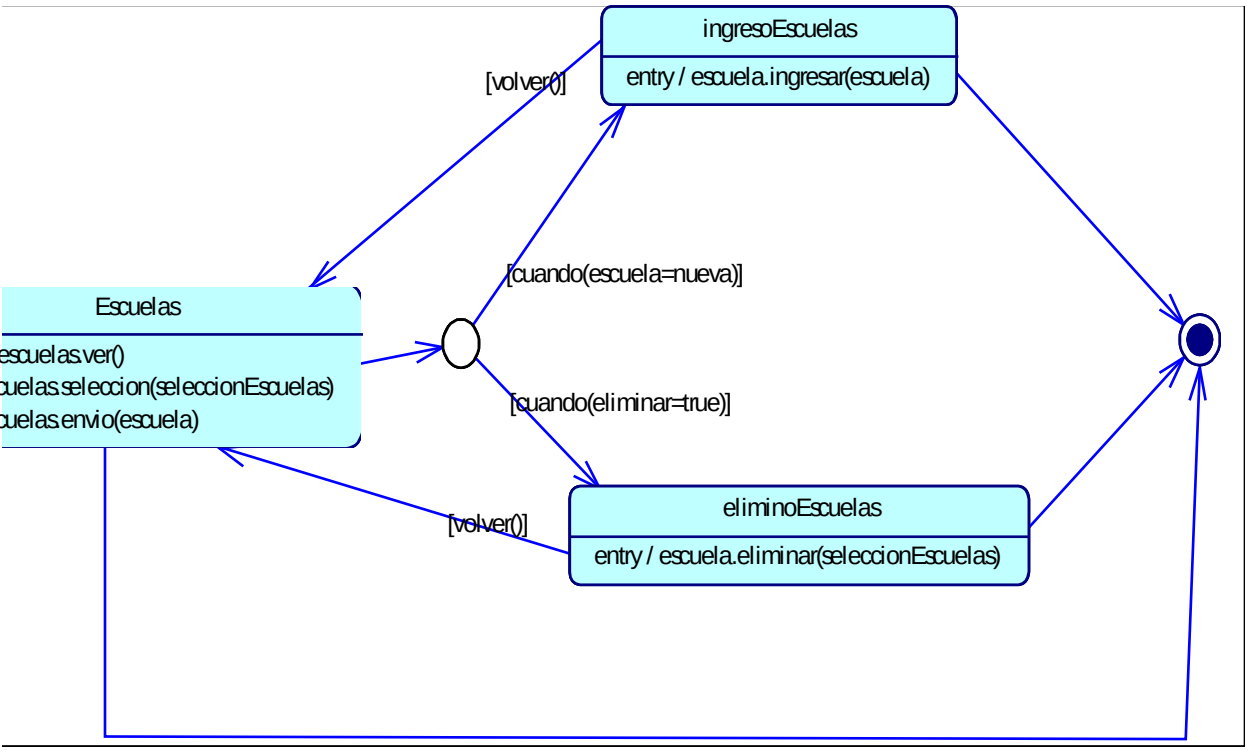


Figura 5.61 Diagrama de Estado de la Clase Escuelas

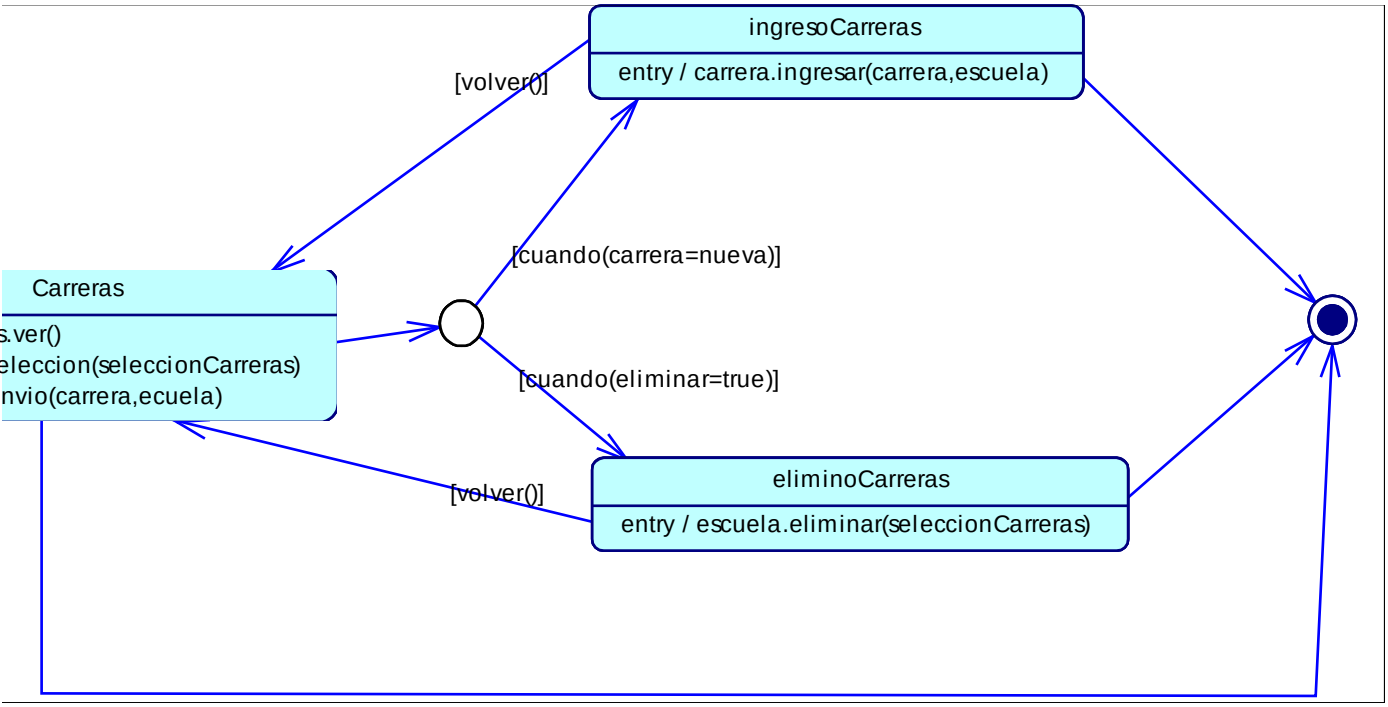


Figura 5.62 Diagrama de Estado de la Clase Carreras

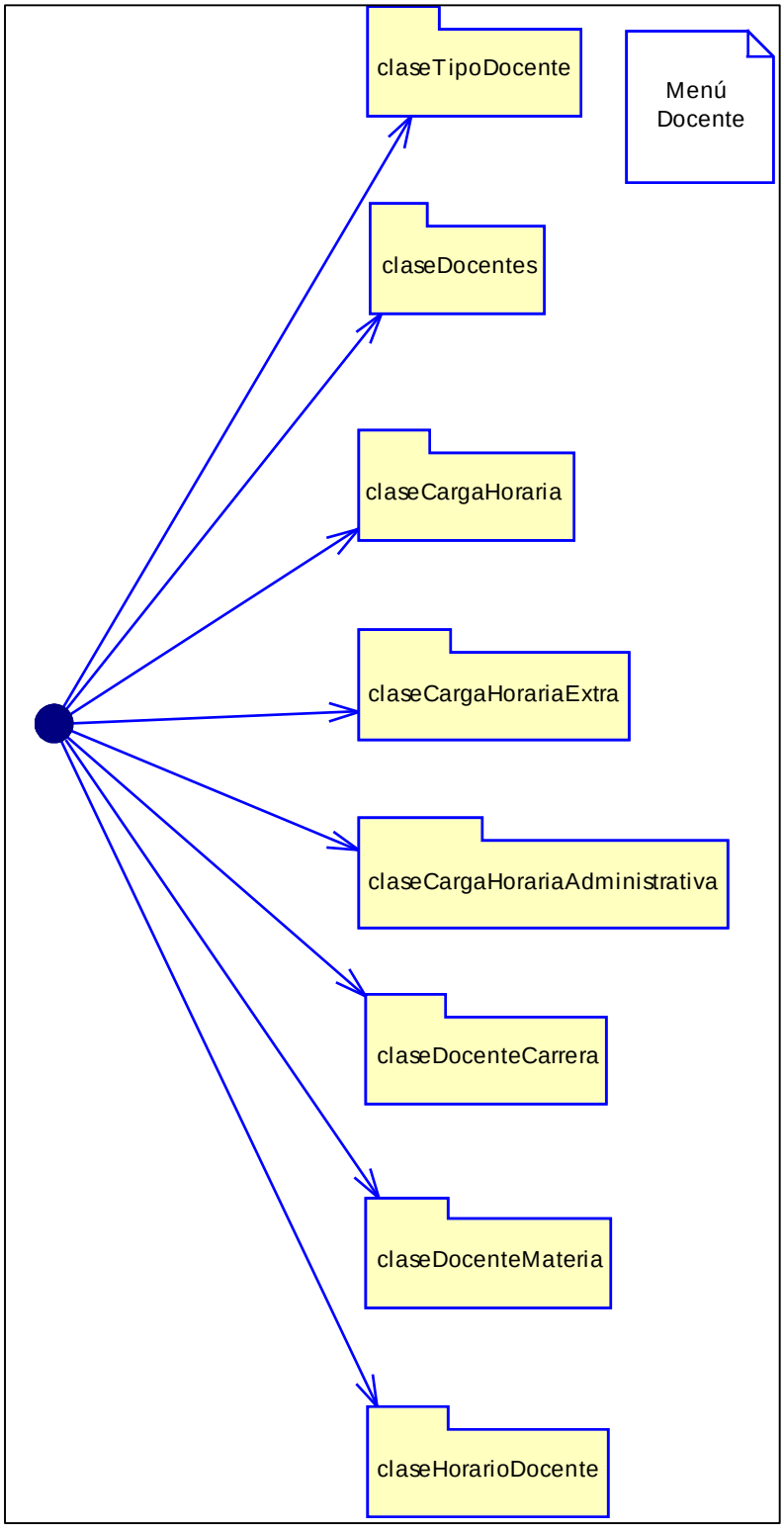


Figura 5.63 División en Paquetes del Paquete Docentes

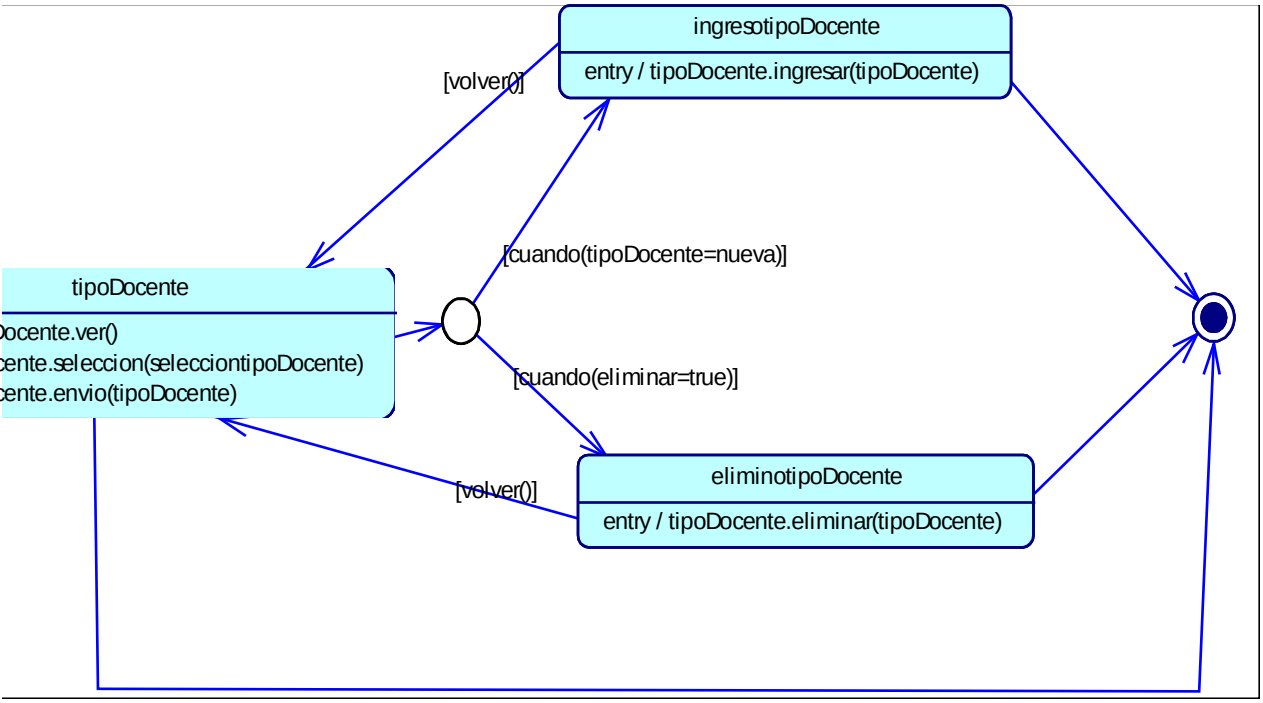


Figura 5.64 Diagrama de Estado de la Clase tipoDocente

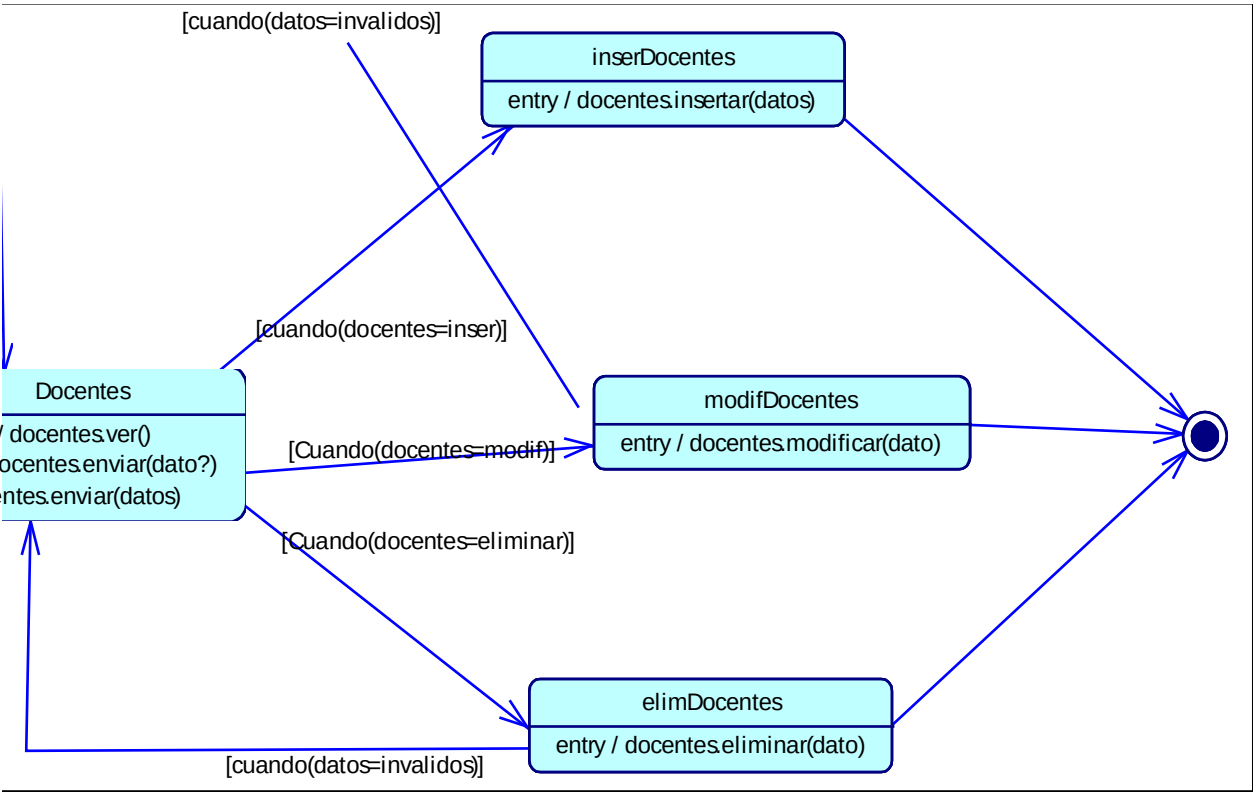


Figura 5.65 Diagrama de Estado de la Clase Docentes

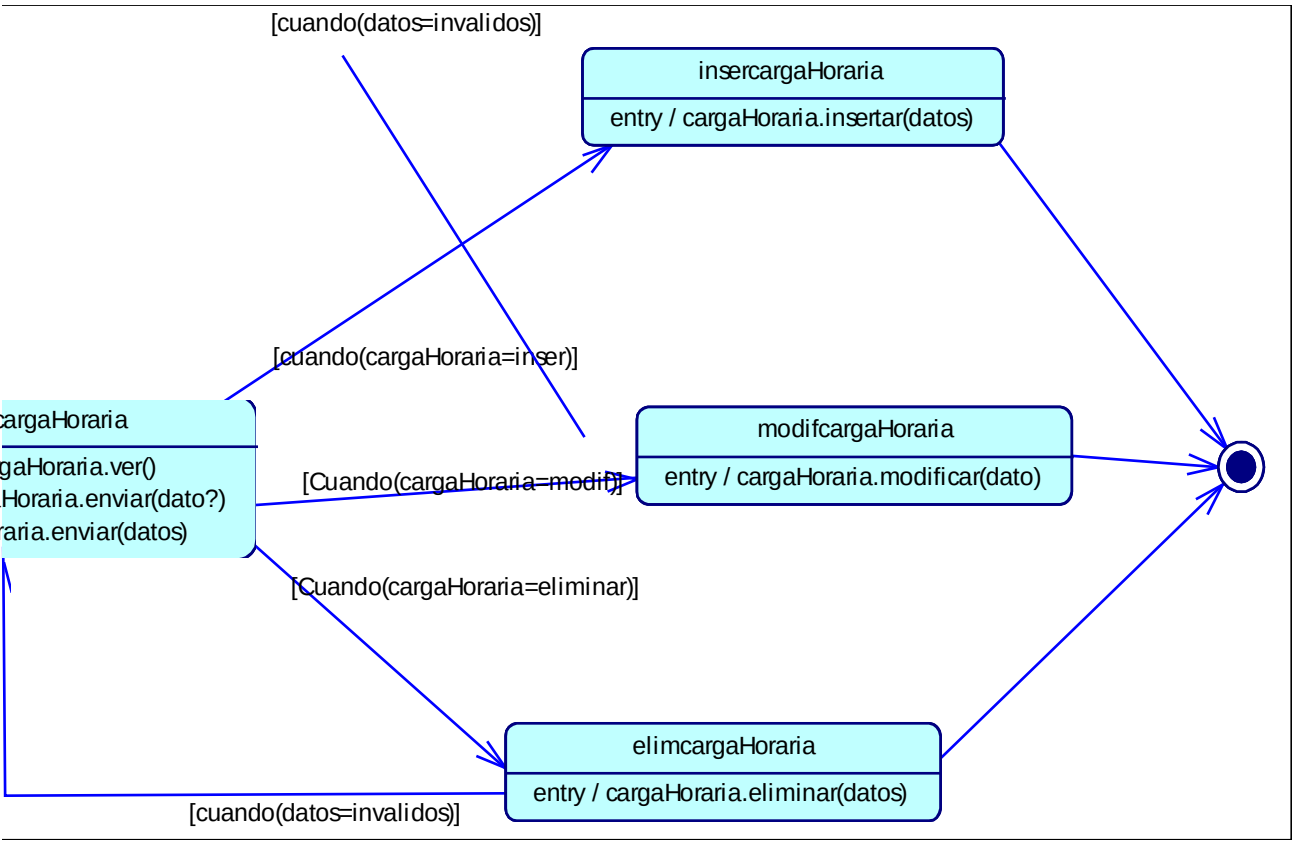


Figura 5.66 Diagrama de Estado de la Clase cargaHoraria

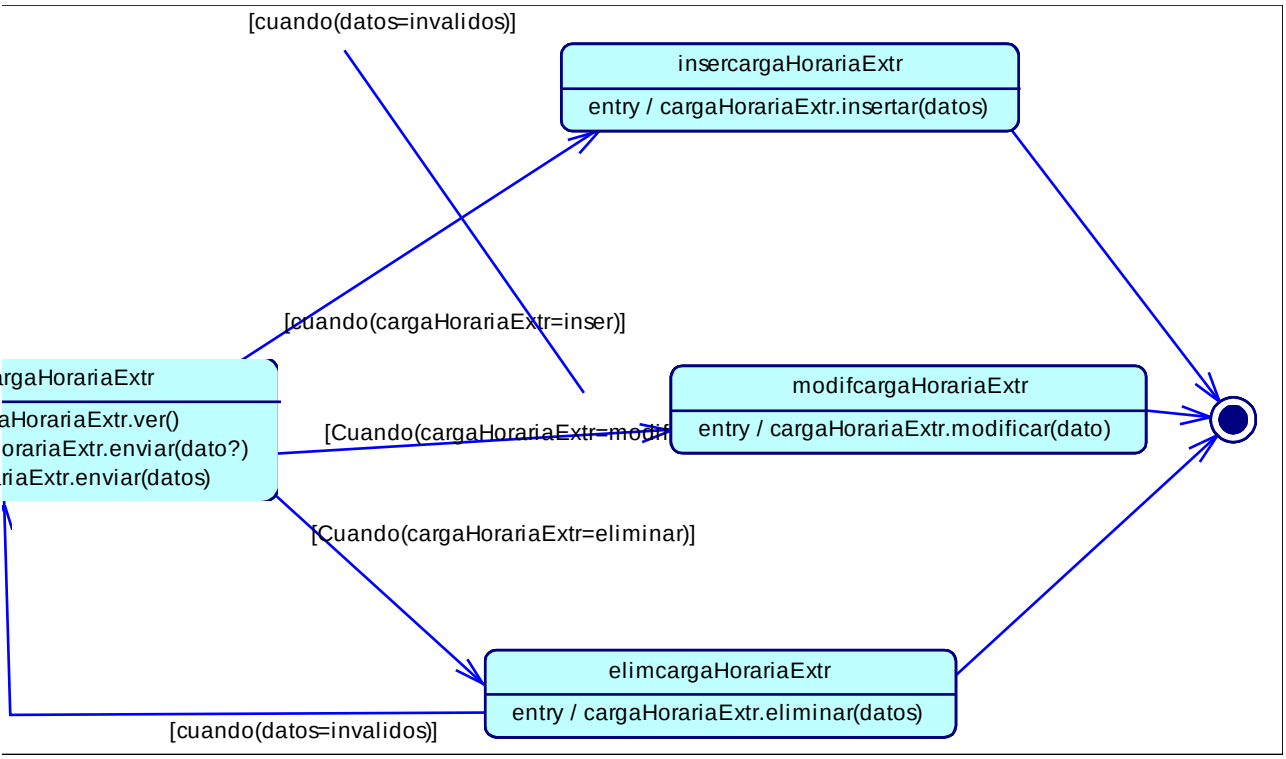
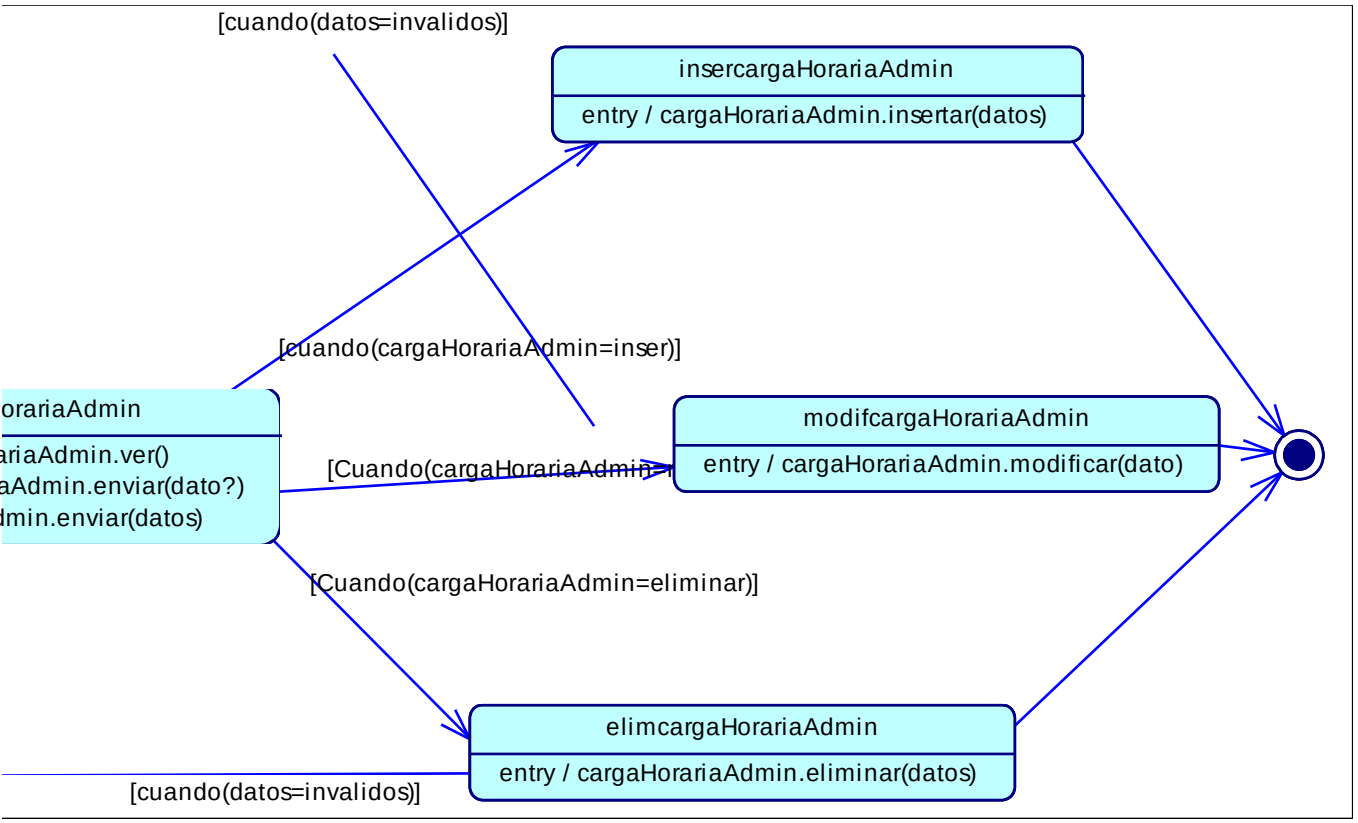


Figura 5.67 Diagrama de Estado de la Clase cargaHorariaExtr



igura 5.68 Diagrama de Estado de la Clase cargaHorariaAdmin

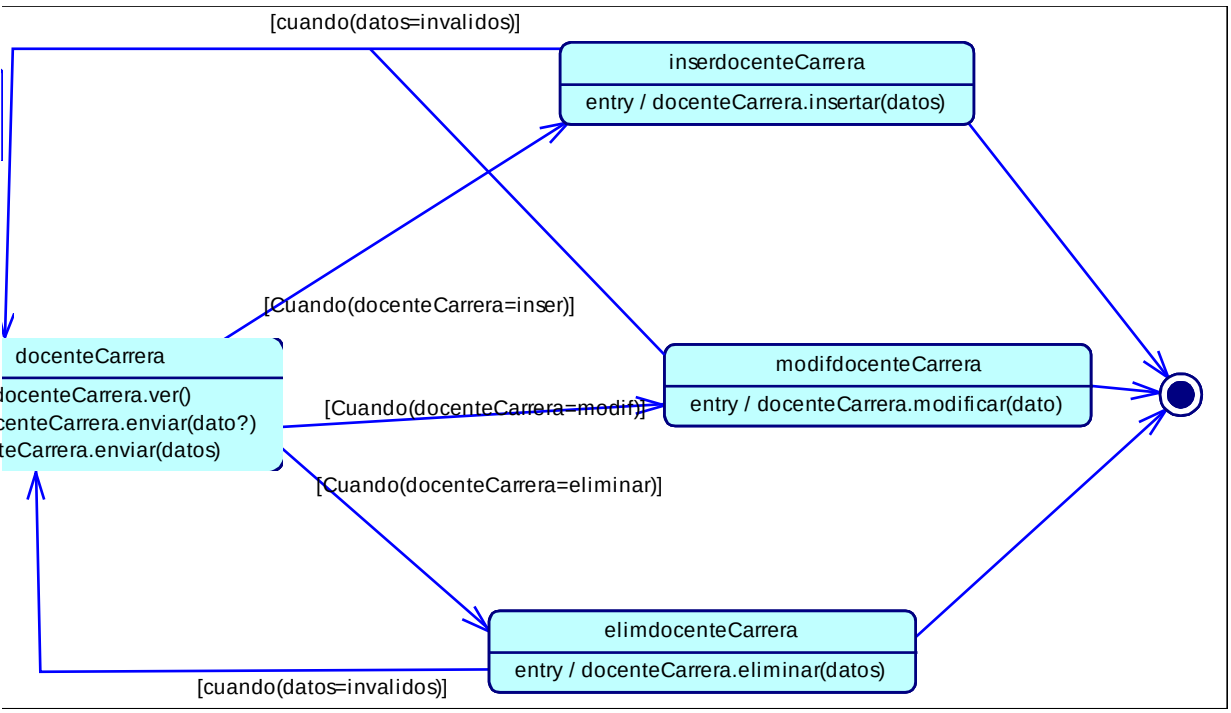


Figura 5.69 Diagrama de Estado de la Clase docenteCarrera

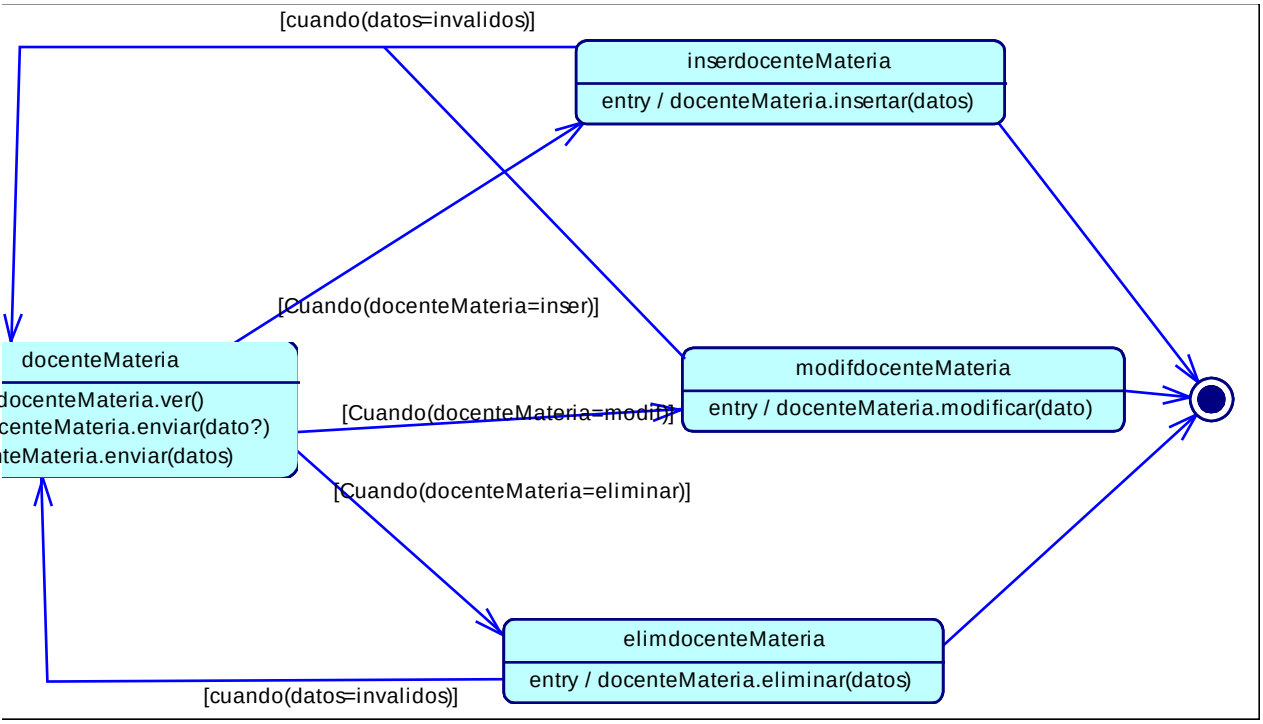


Figura 5.70 Diagrama de Estado de la Clase docenteMateria

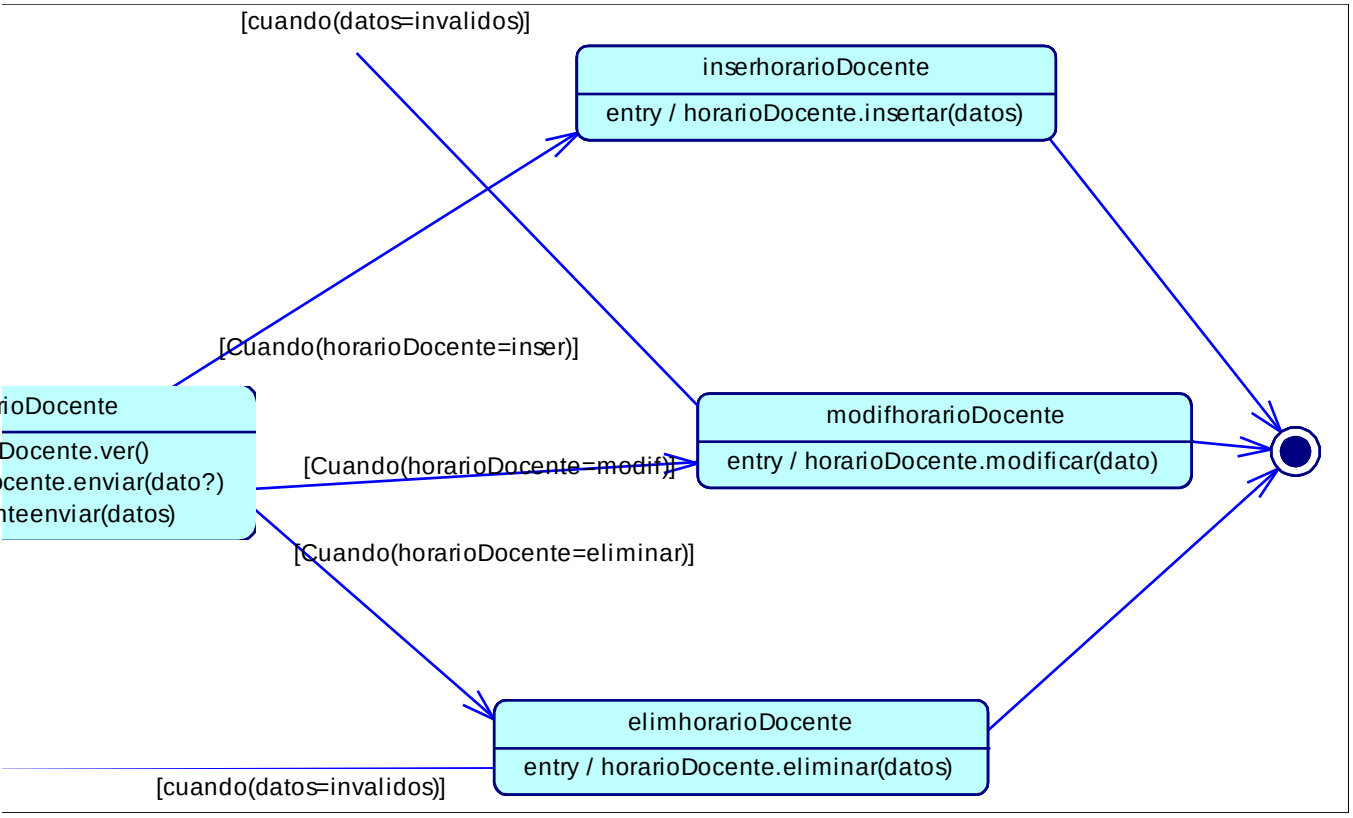


Figura 5.71 Diagrama de Estado de la Clase horarioDocente

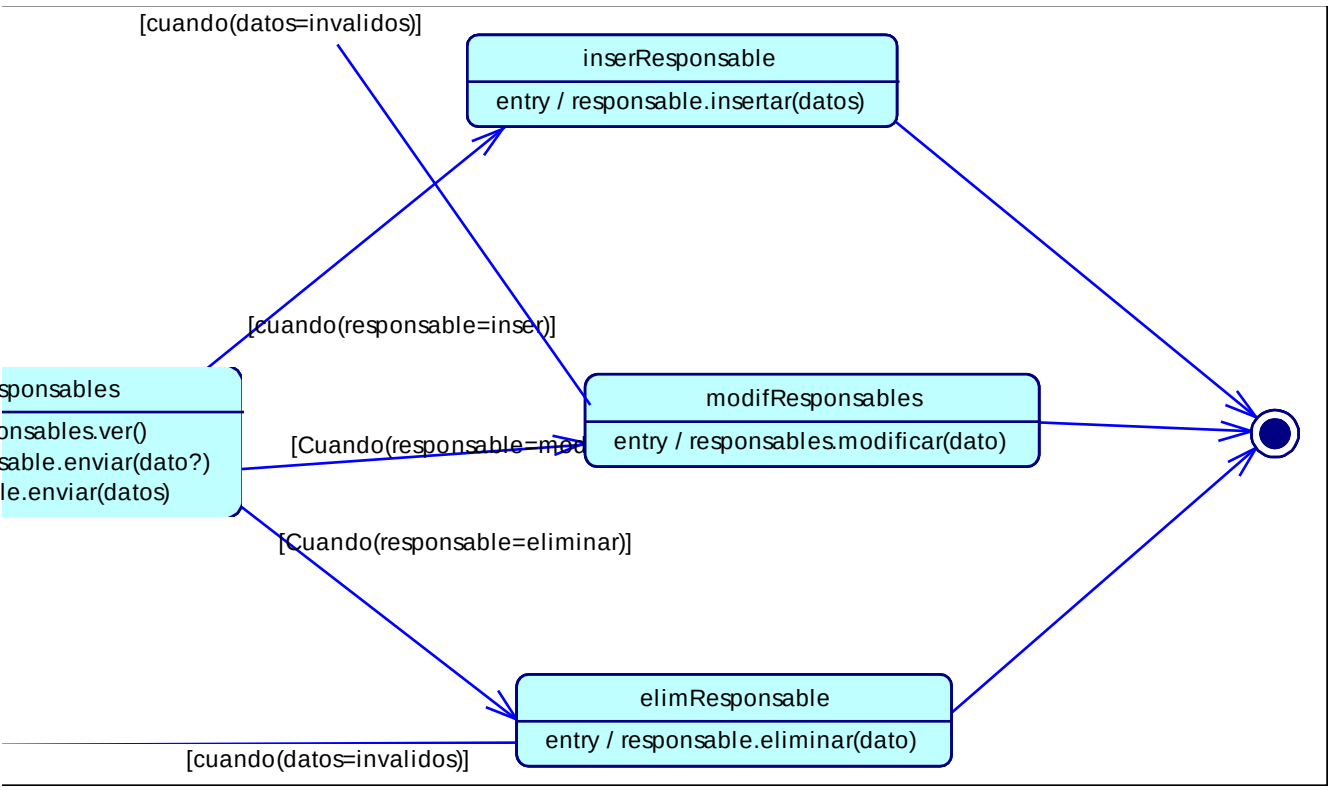
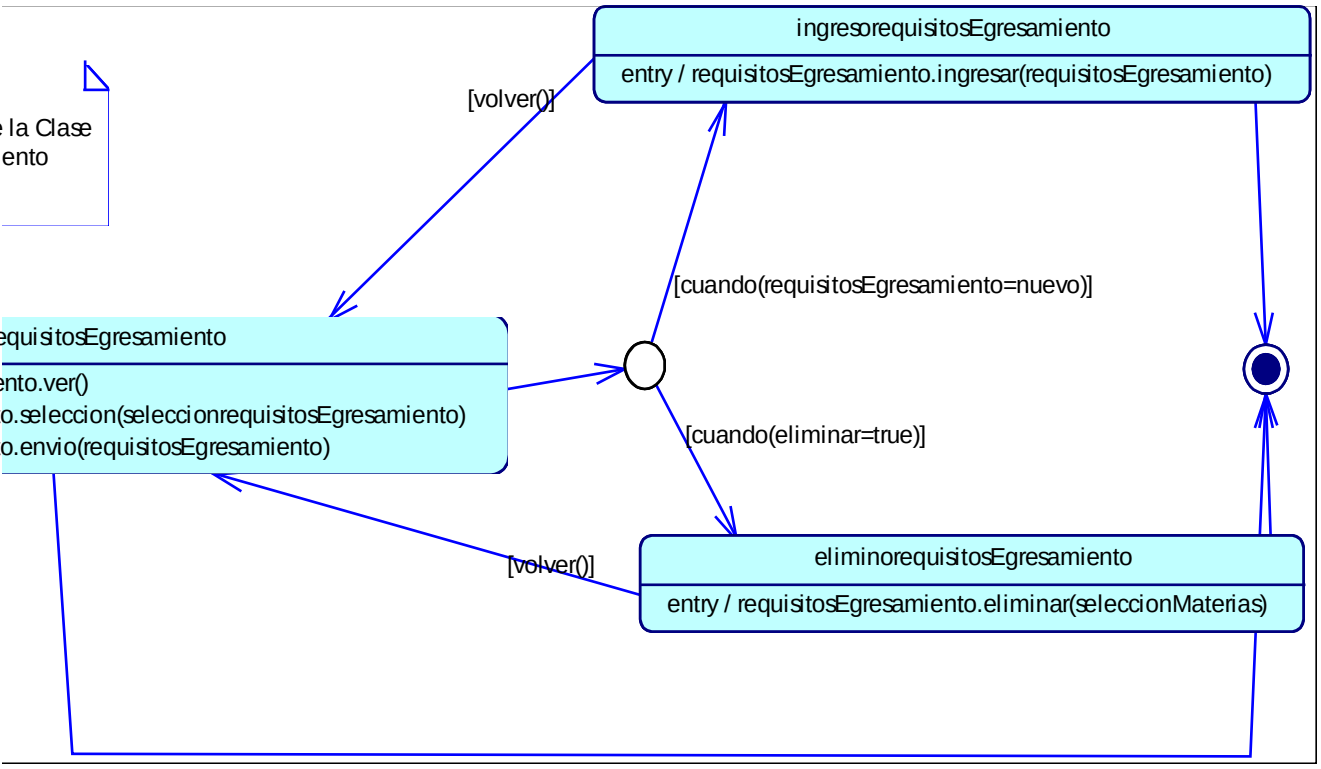
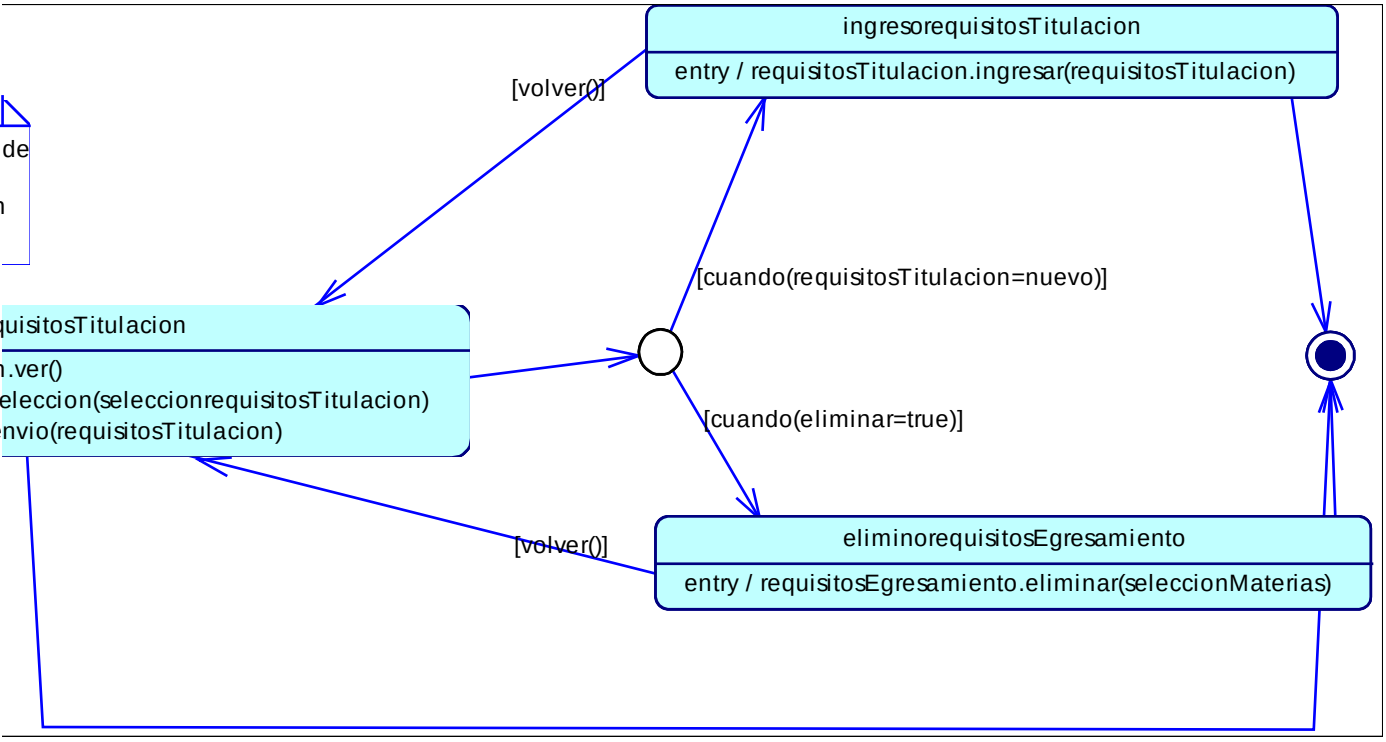


Figura 5.72 Diagrama de Estado de la Clase Responsables



ura 5.73 Diagrama de Estado de la Clase requisitosEgresamiento



igura 5.74 Diagrama de Estado de la Clase requisitosTitulacion

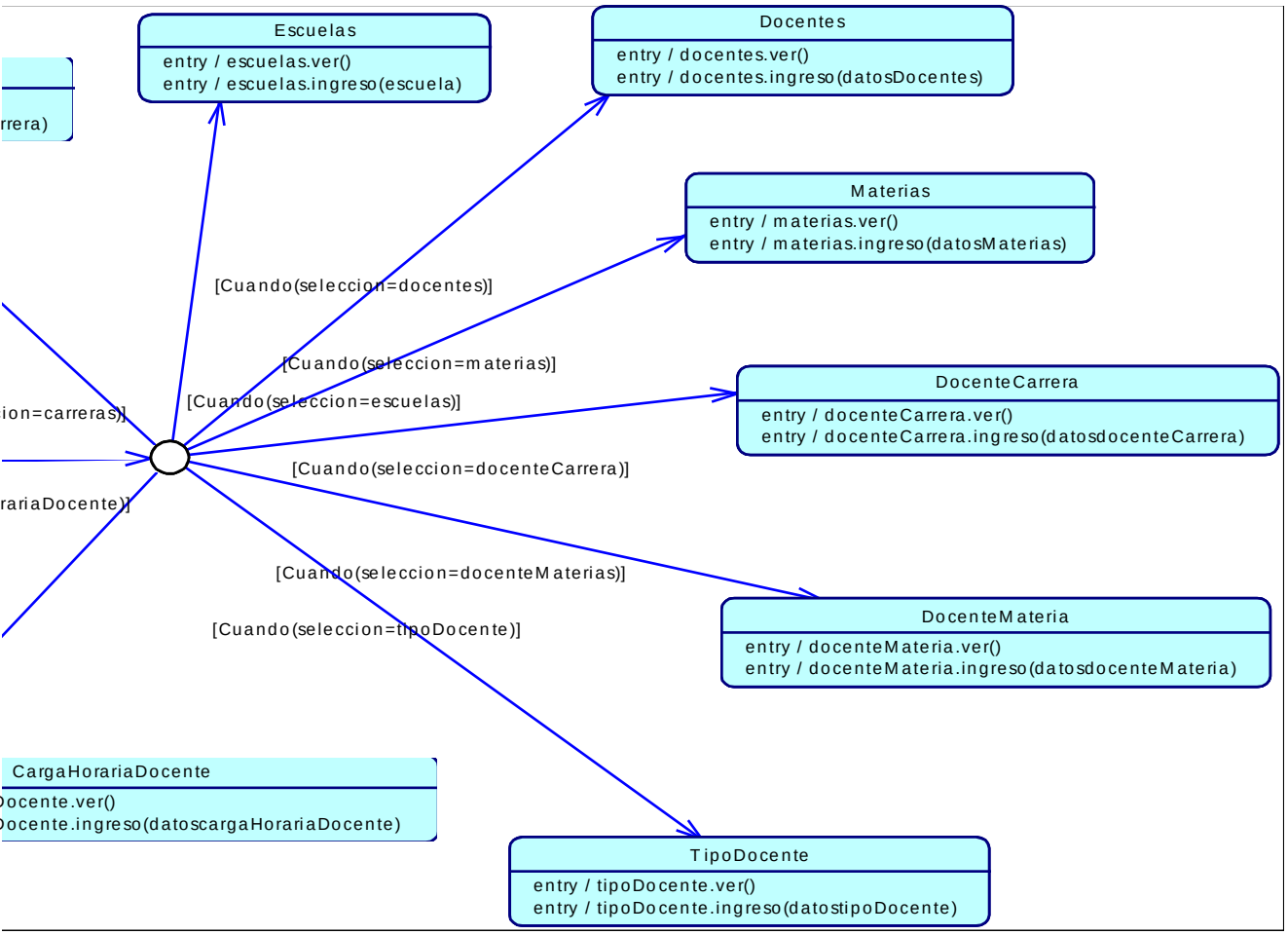


Figura 5.75 Diagrama de Estado de la Clase sistemaAcademico

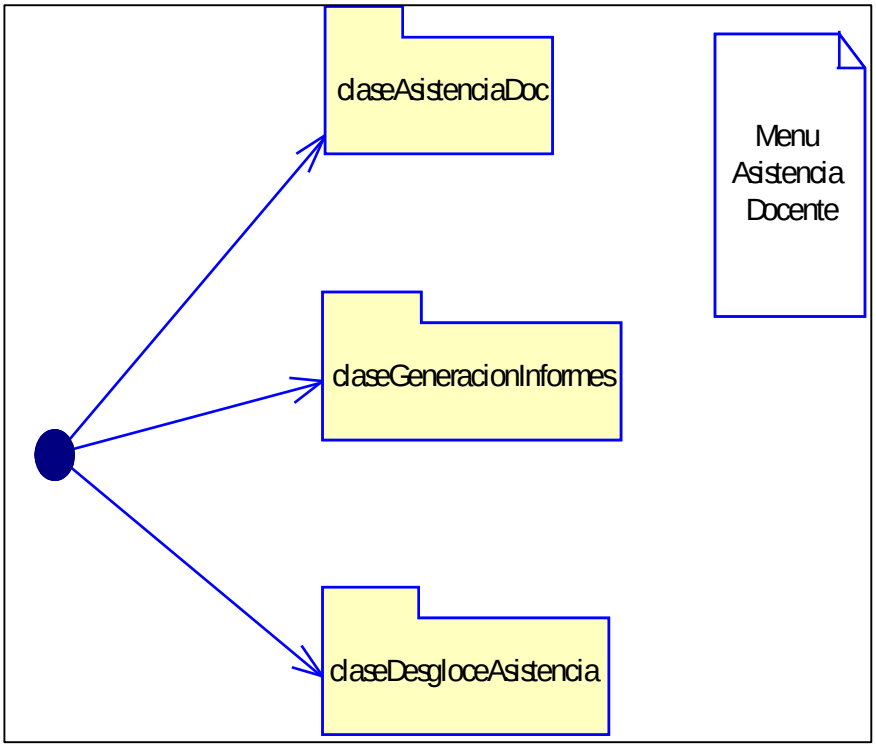


Figura 5.76 División del Paquete ClaseAsistenciaDocente

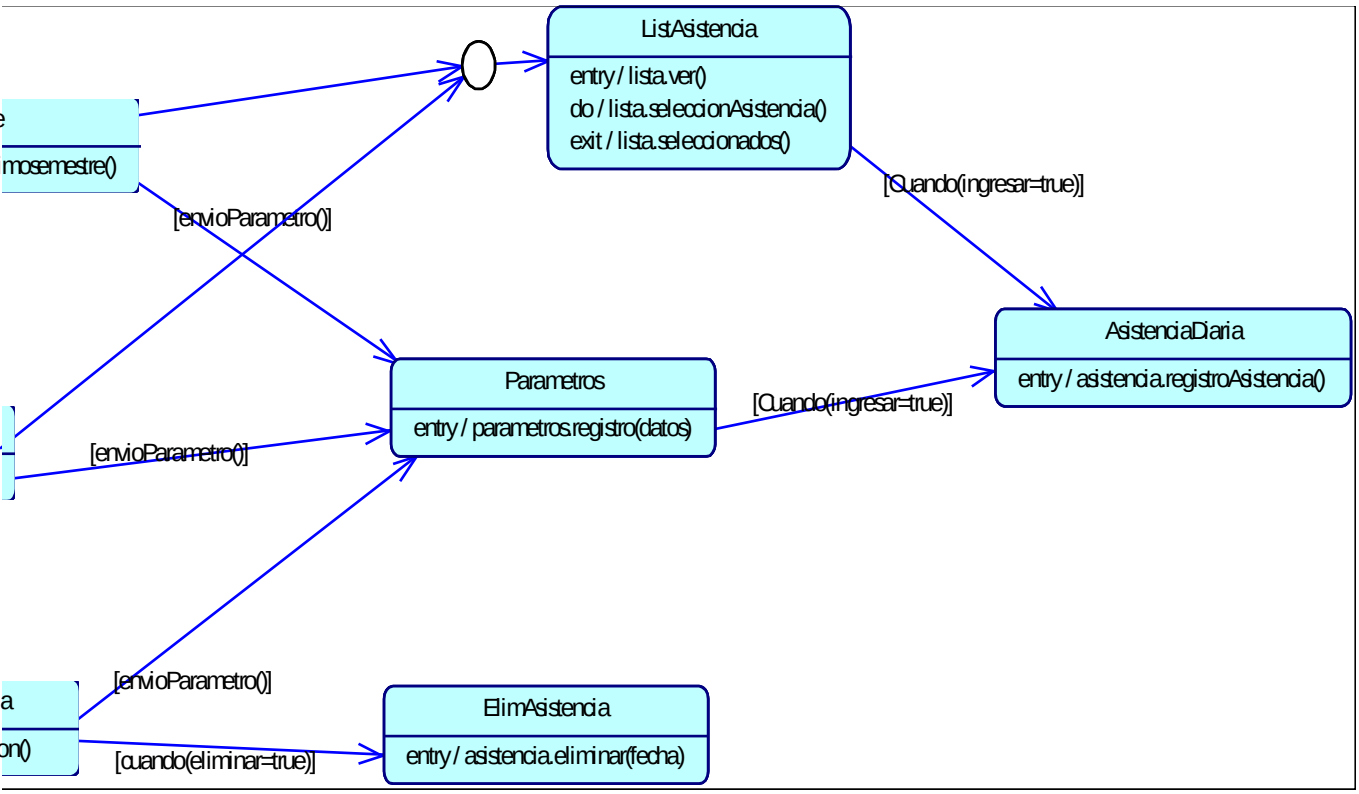
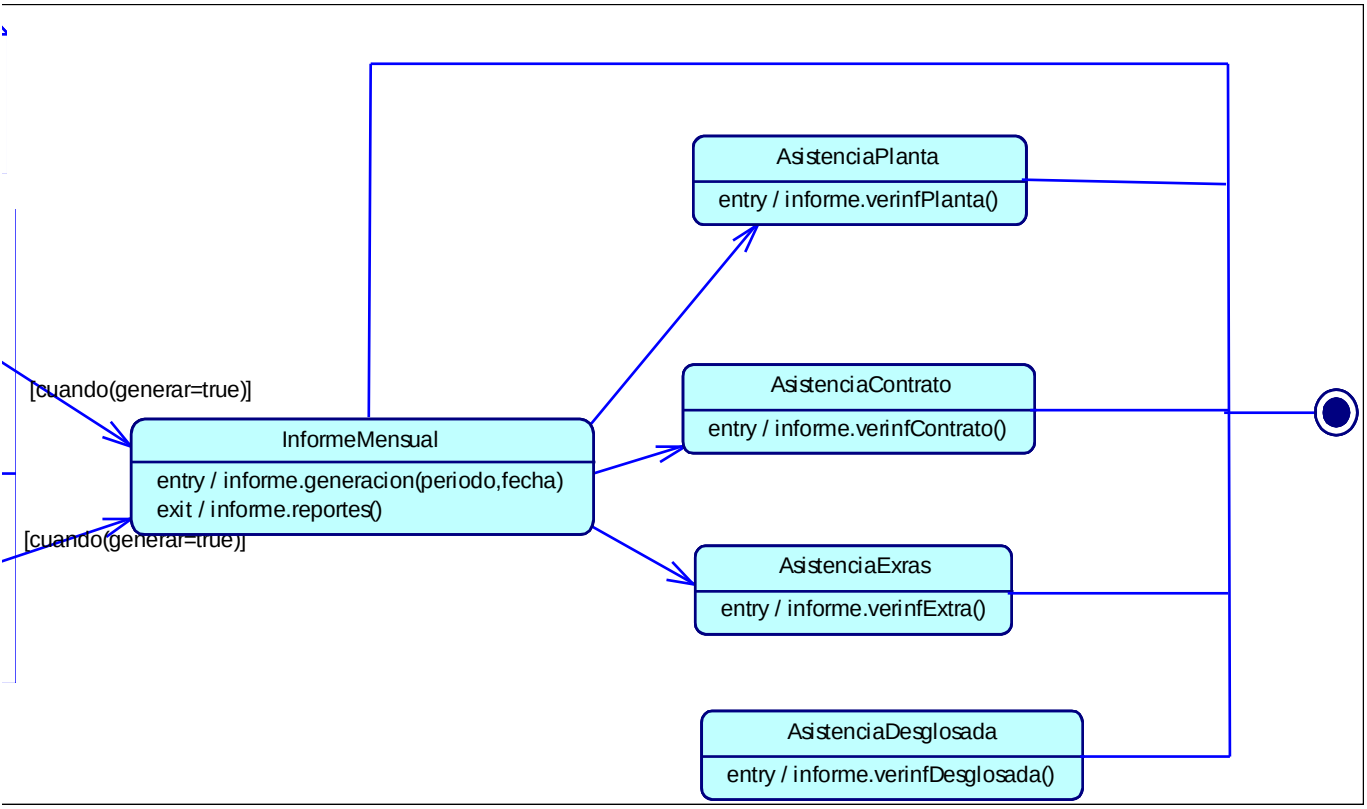
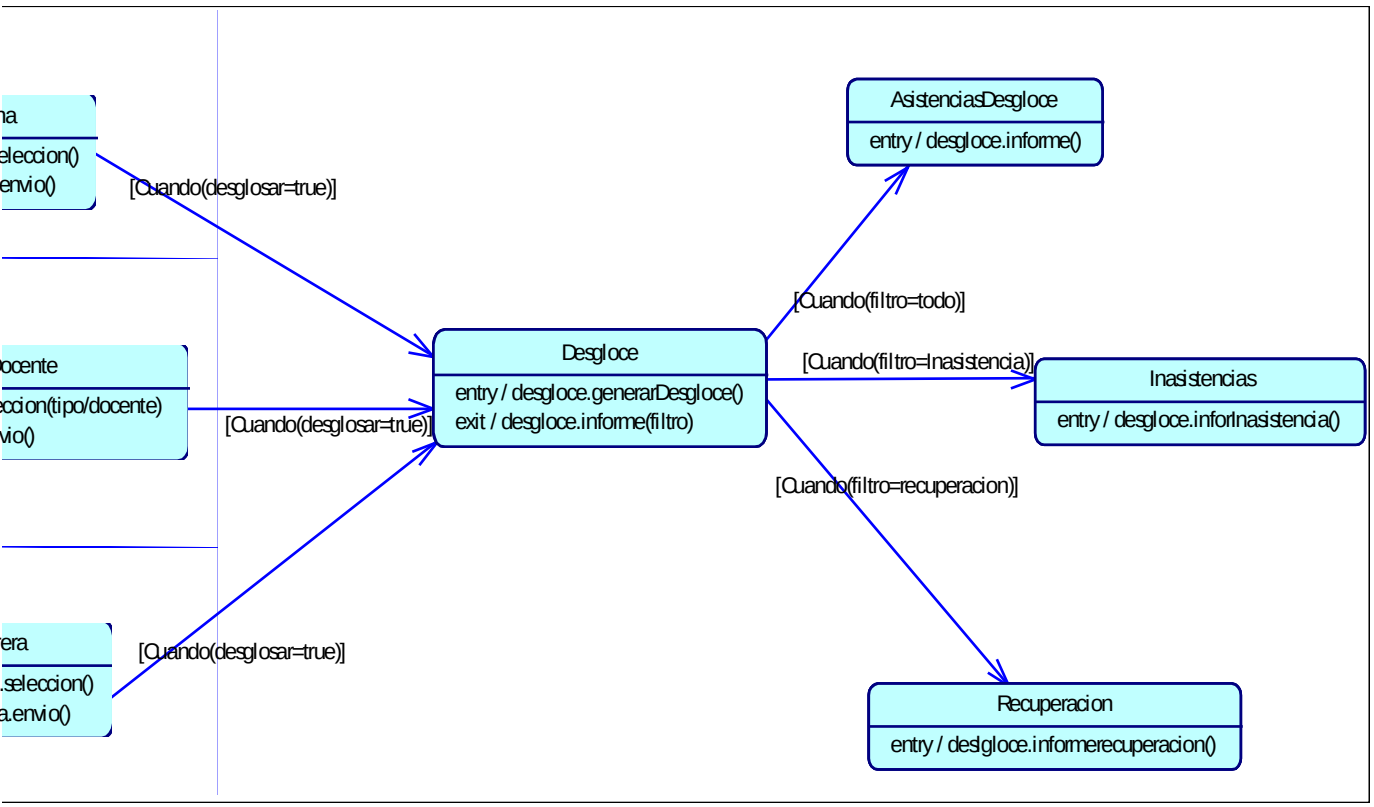


Figura 5.77 Diagrama de Estado de la Clase asistenciaDocente

ATIVO WEB



gura 5.78 Diagrama de Estado de la Clase generacionInformes



igura 5.79 Diagrama de Estado de la Clase desgloceAsistencia

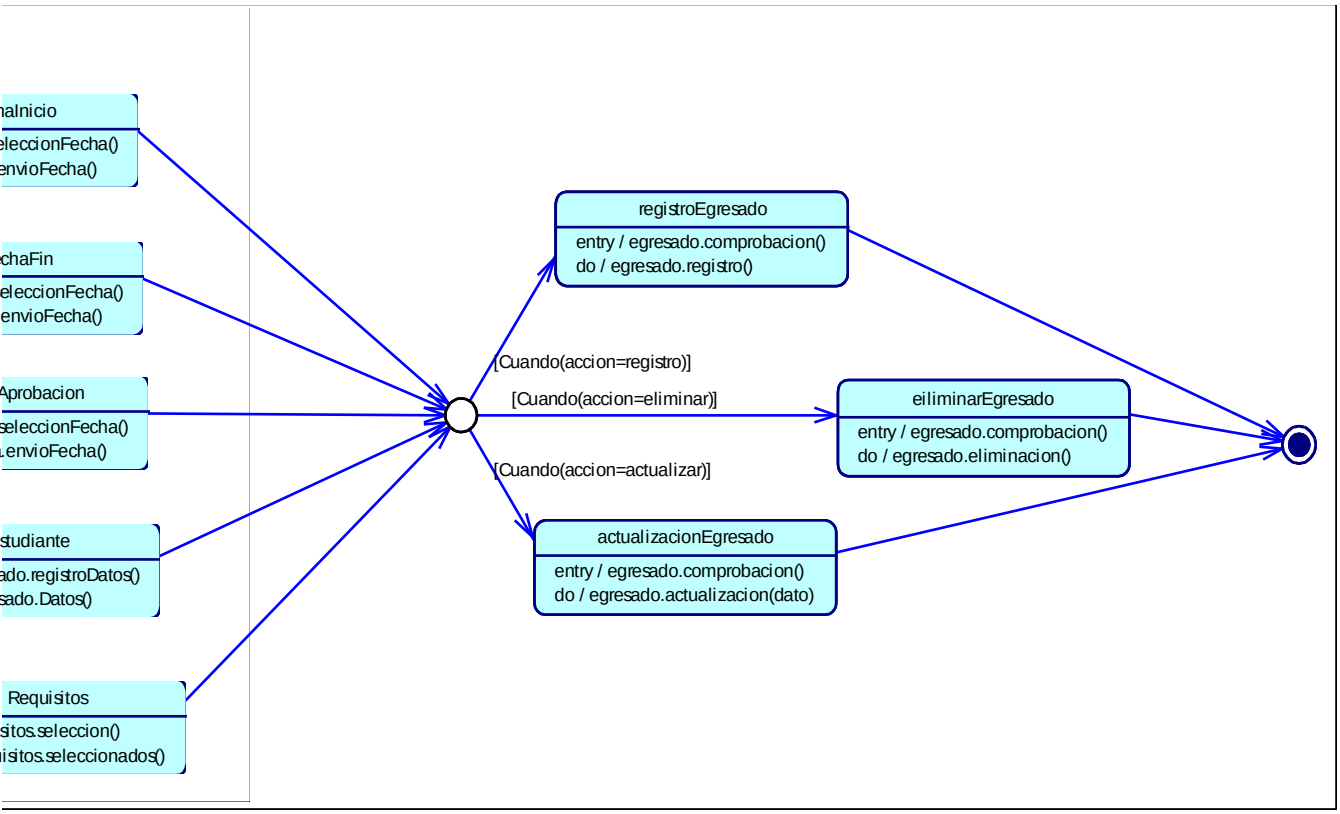


Figura 5.80 Diagrama de Estado de la Clase Egresos

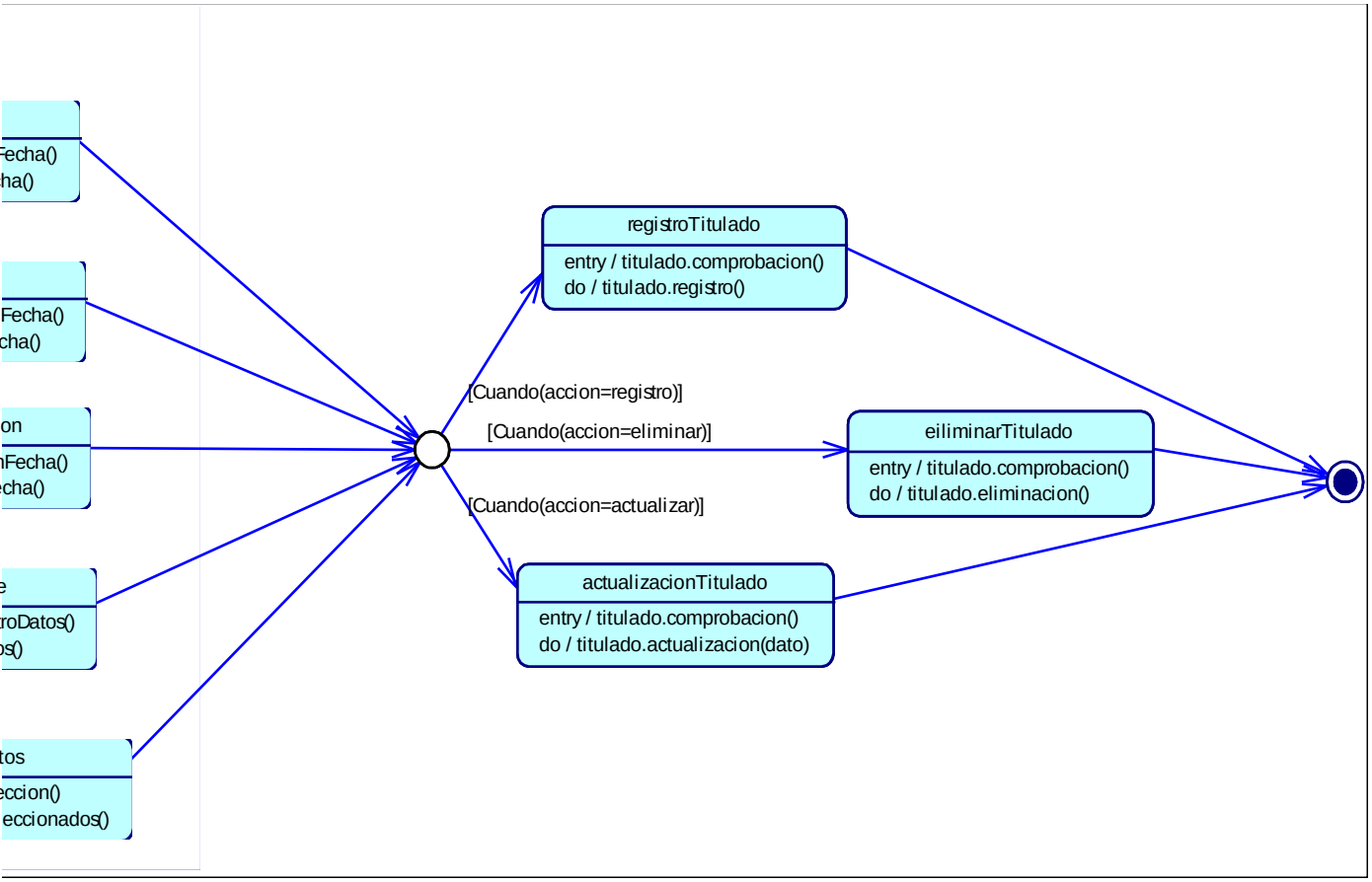


Figura 5.81 Diagrama de Estado de la Clase Titulación

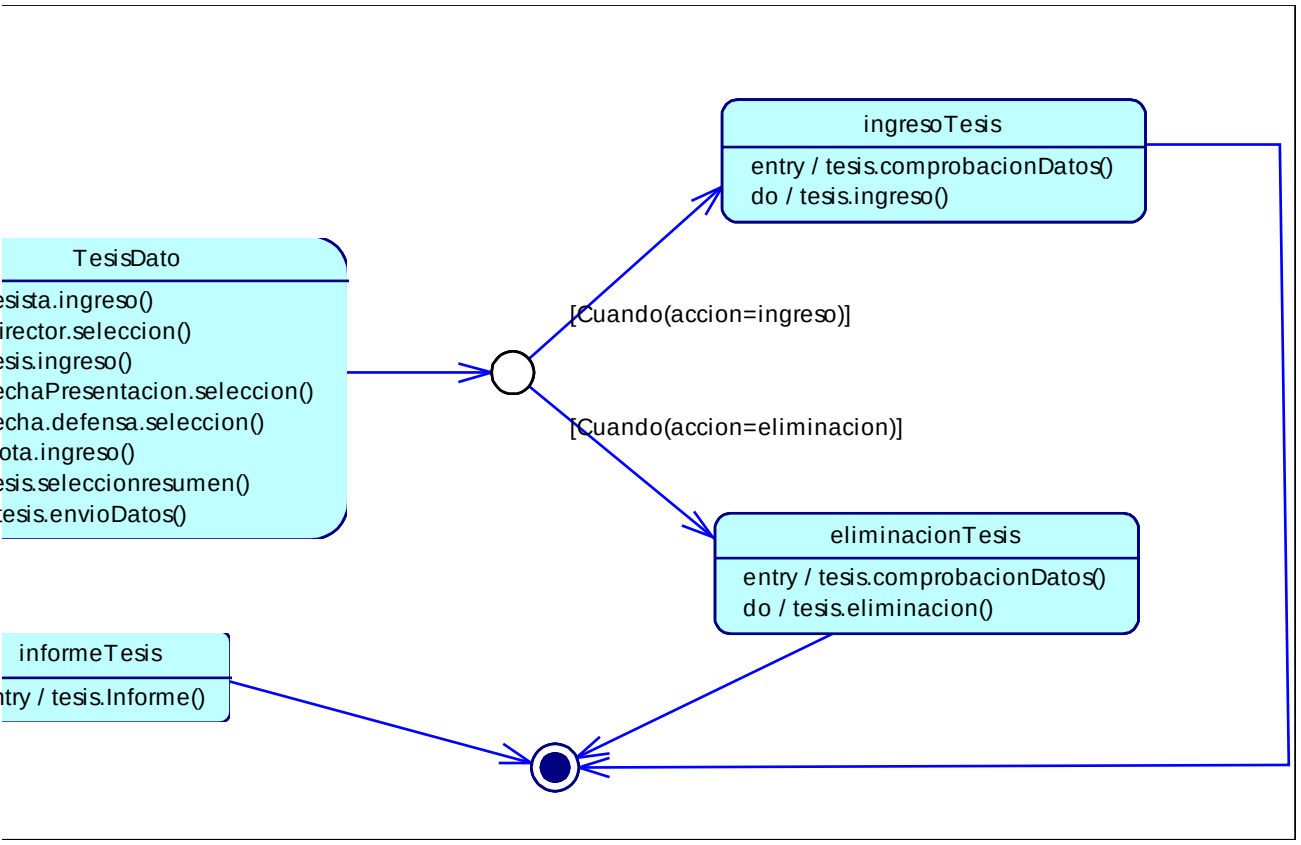


Figura 5.82 Diagrama de Estado de la Clase Tesis

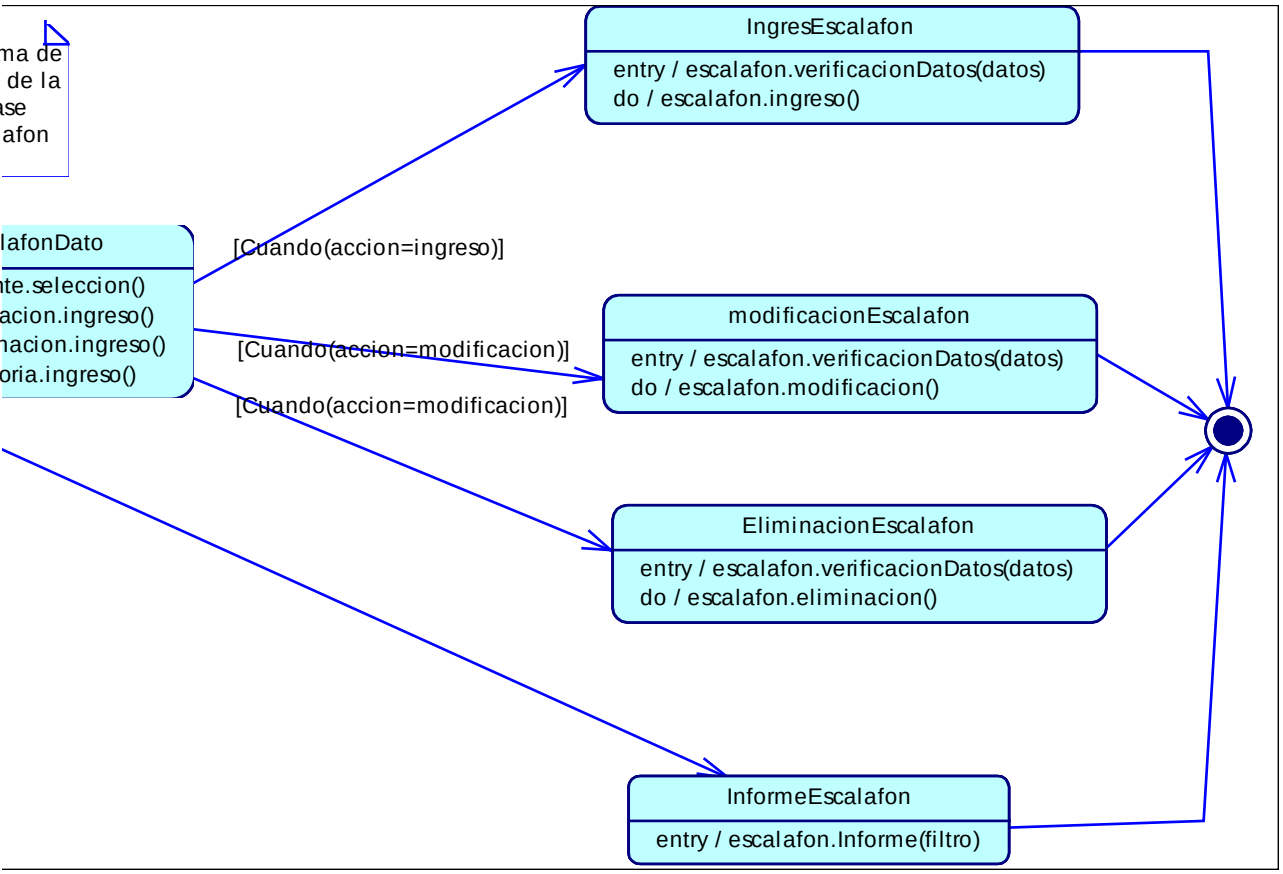


Figura 5.83 Diagrama de Estado de la Clase Escalafon

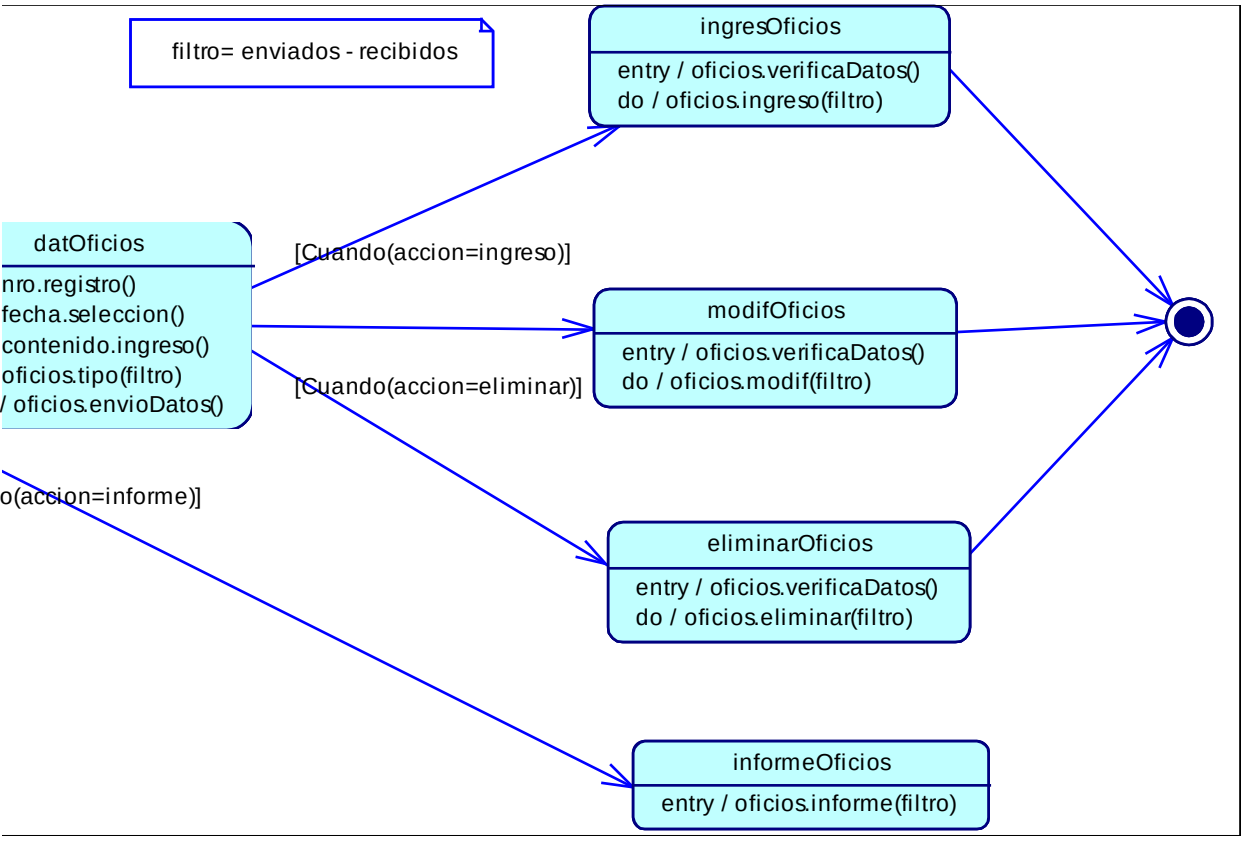


Figura 5.84 Diagrama de Estado de la Clase Oficios

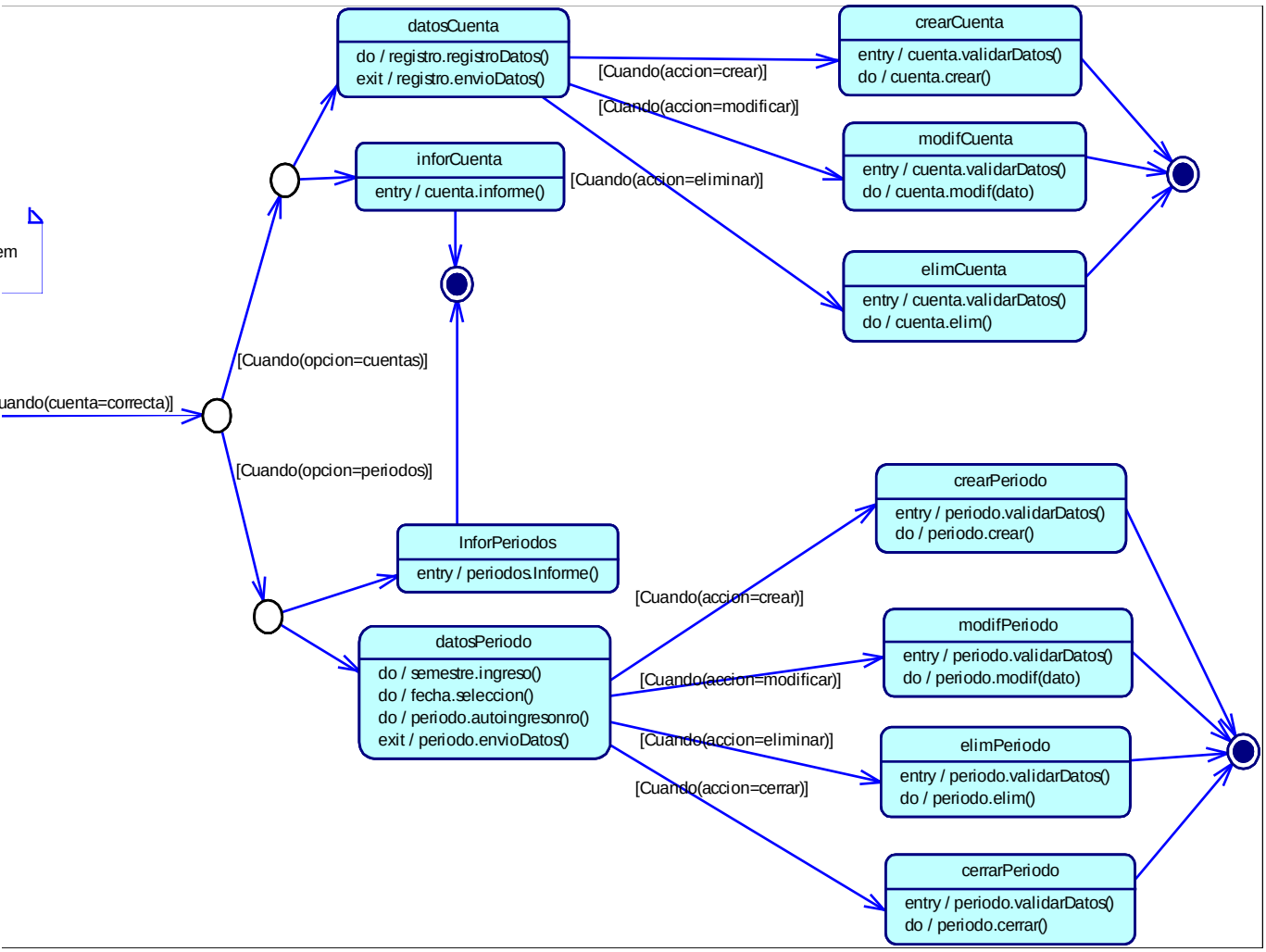
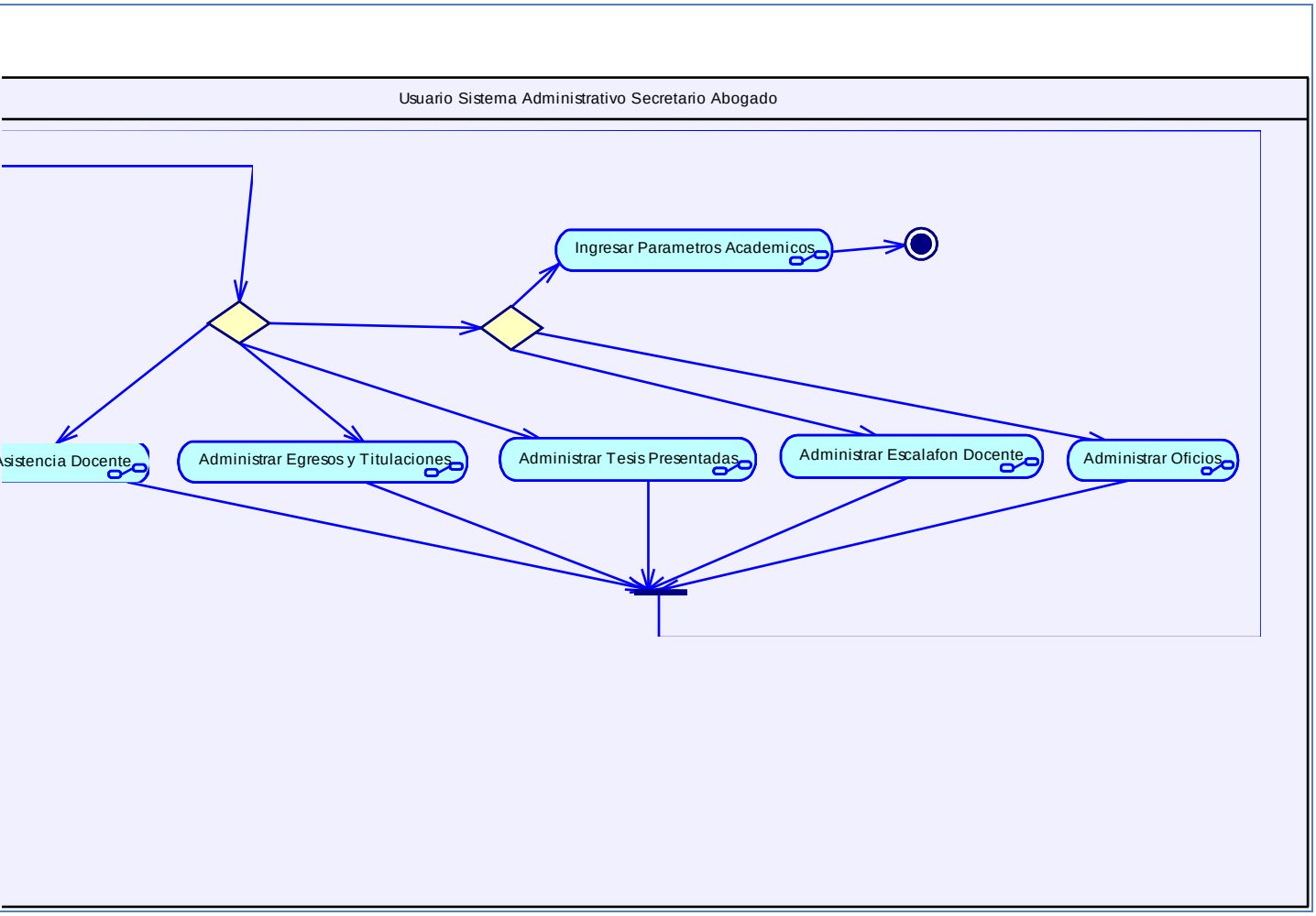


Figura 5.85 Diagrama de Estado de la Clase adminSistem

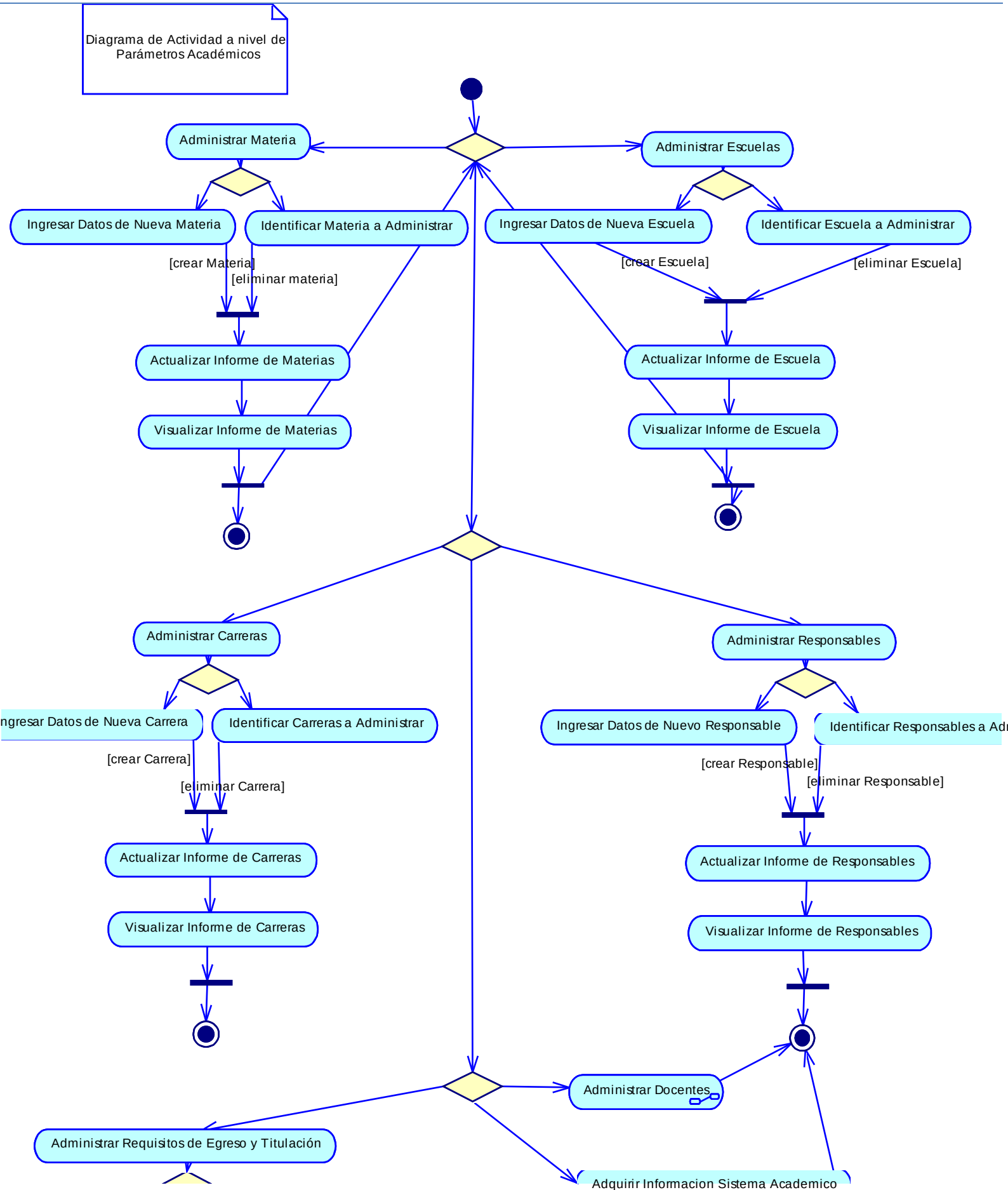
5.5.7. Diagramas de Actividad

Los diagramas de actividad son de gran utilidad para describir la coordinación de las actividades que se realizan en la ejecución de los diversos módulos del Sistema Administrativo Secretario Abogado FICA.



ivo Departamental Secretario Abogado con sus 2 tipos de Usuario

DESARROLLO DEL APLICATIVO WEB



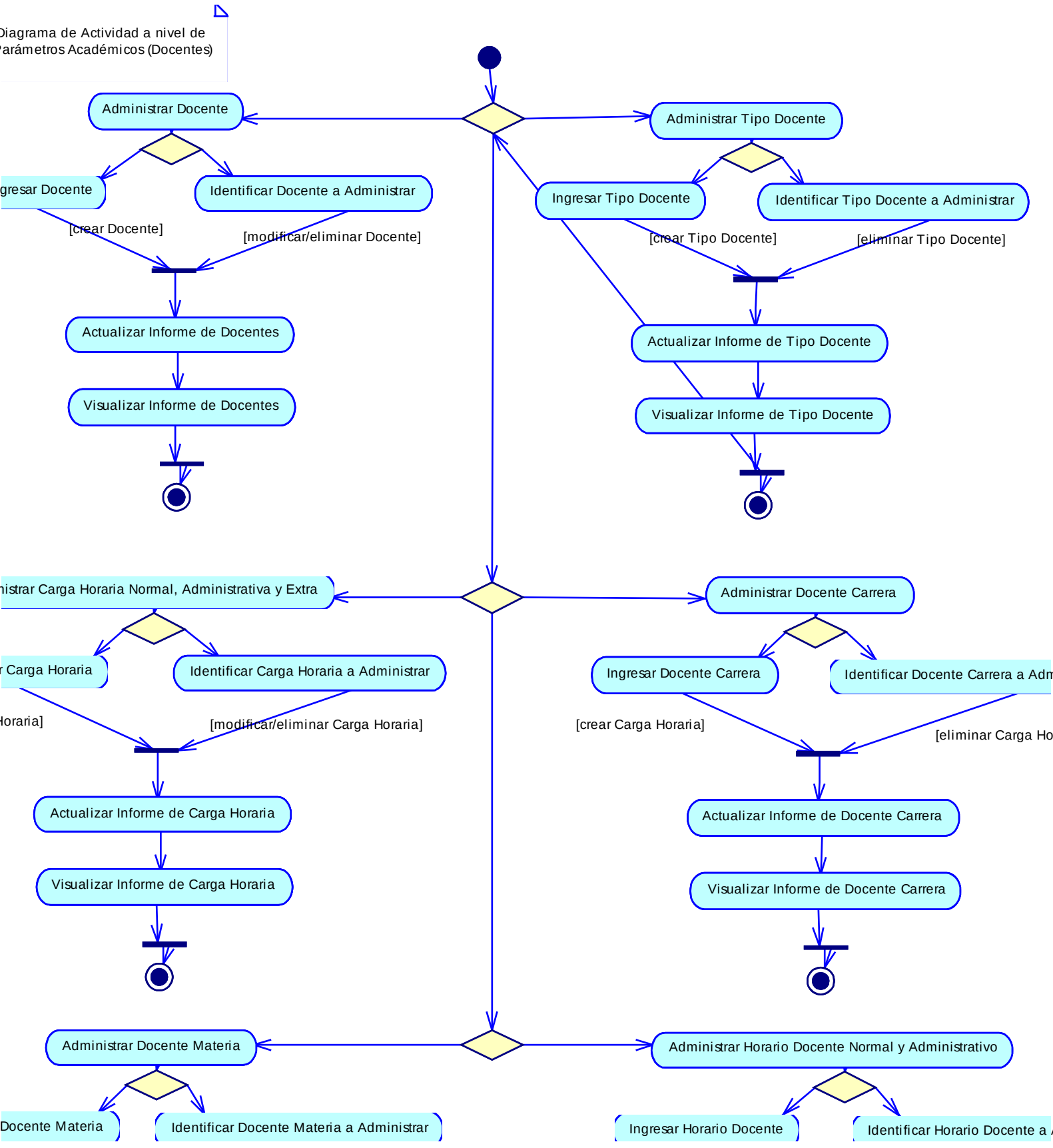


Diagrama de Actividad a nivel de
Parámetros Académicos (Docentes)
Administrar Docente
Administrar Tipo Docente
gresar Docente
Identificar Docente a Administrar
Ingresar Tipo Docente
Identificar Tipo Docente a Administrar
[crear Docente]
[modificar/eliminar Docente]
[crear Tipo Docente]
[eliminar Tipo Docente]
Actualizar Informe de Docentes
Actualizar Informe de Tipo Docente
Visualizar Informe de Docentes
Visualizar Informe de Tipo Docente
nistrar Carga Horaria Normal, Administrativa y Extra
Administrar Docente Carrera
r Carga Horaria
Identificar Carga Horaria a Administrar
Ingresar Docente Carrera
Identificar Docente Carrera a Adm
Horaria]
[modificar/eliminar Carga Horaria]
[crear Carga Horaria]
[eliminar Carga Hor
Actualizar Informe de Carga Horaria
Actualizar Informe de Docente Carrera
Visualizar Informe de Carga Horaria
Visualizar Informe de Docente Carrera
Administrar Docente Materia
Administrar Horario Docente Normal y Administrativo
Docente Materia
Identificar Docente Materia a Administrar
Ingresar Horario Docente
Identificar Horario Docente a A

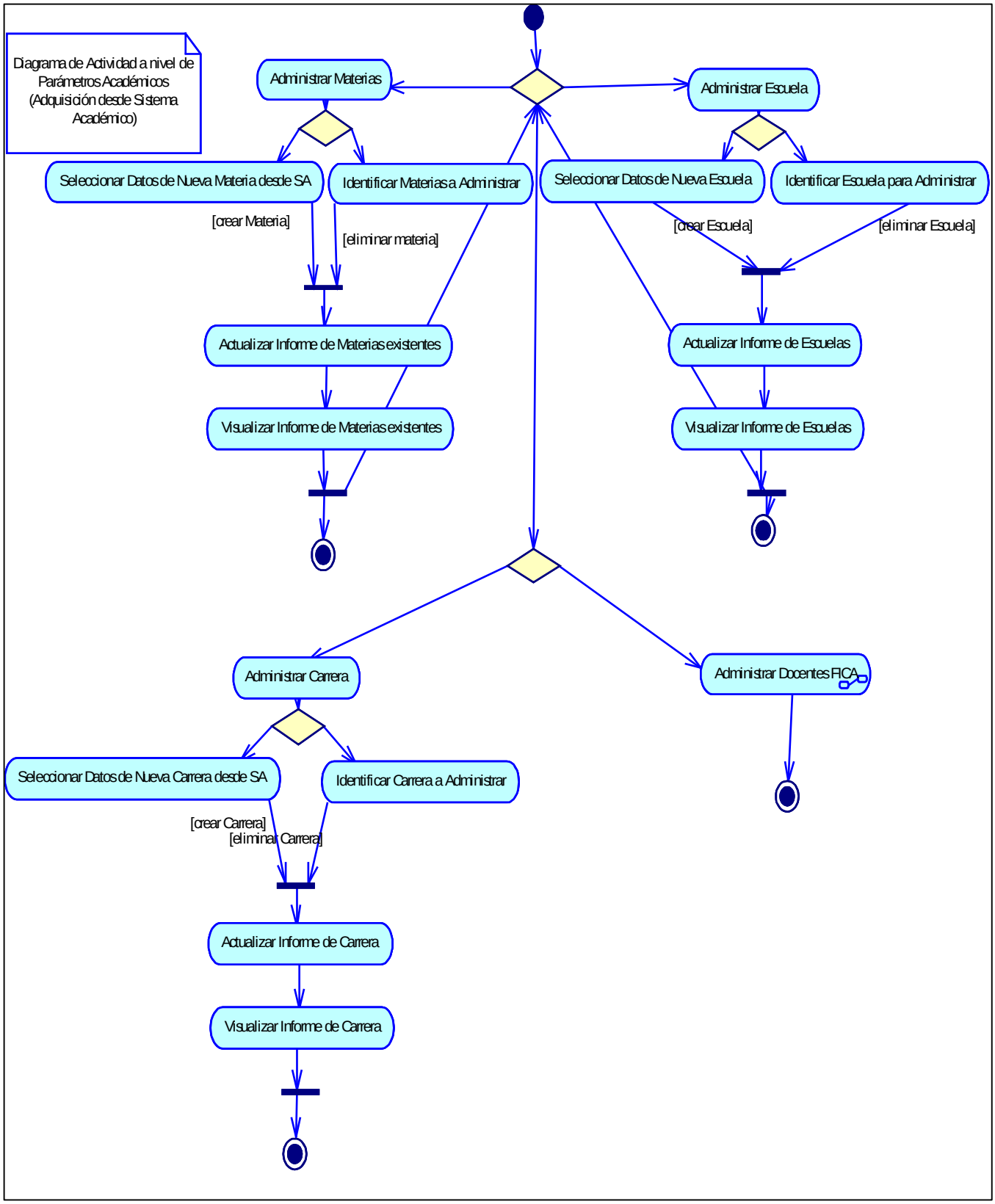
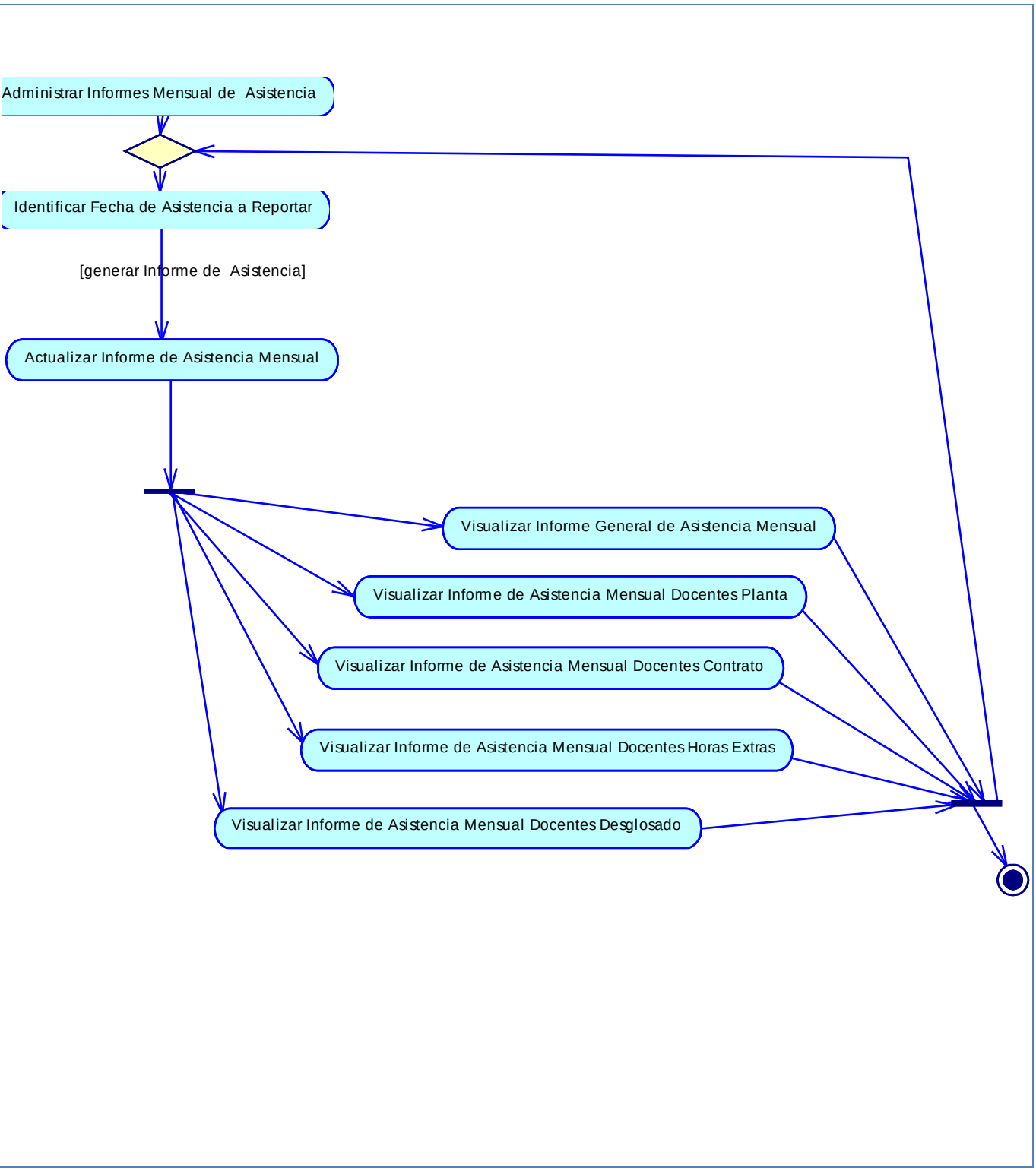
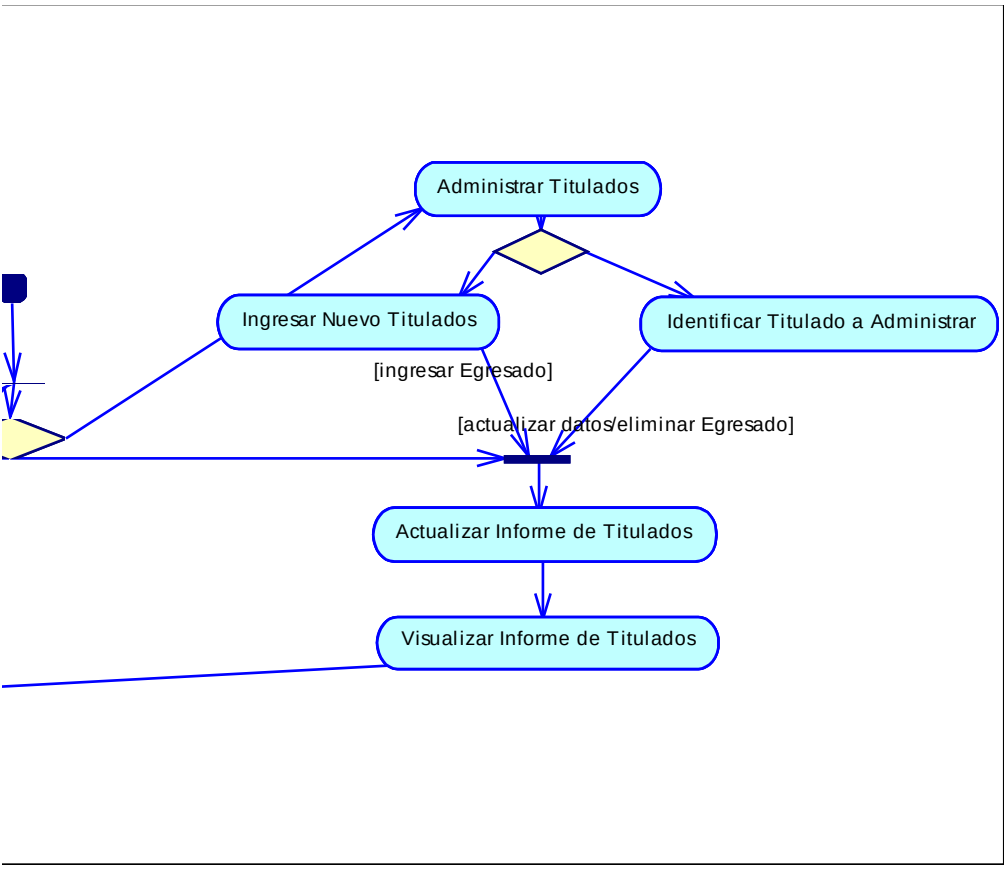


Figura 5.89 Diagrama de Actividad a nivel de Parámetros Académicos

(Adquisición desde Sistema Académico)



ad a nivel de Asistencia Docente



a nivel de Egresos y Titulaciones

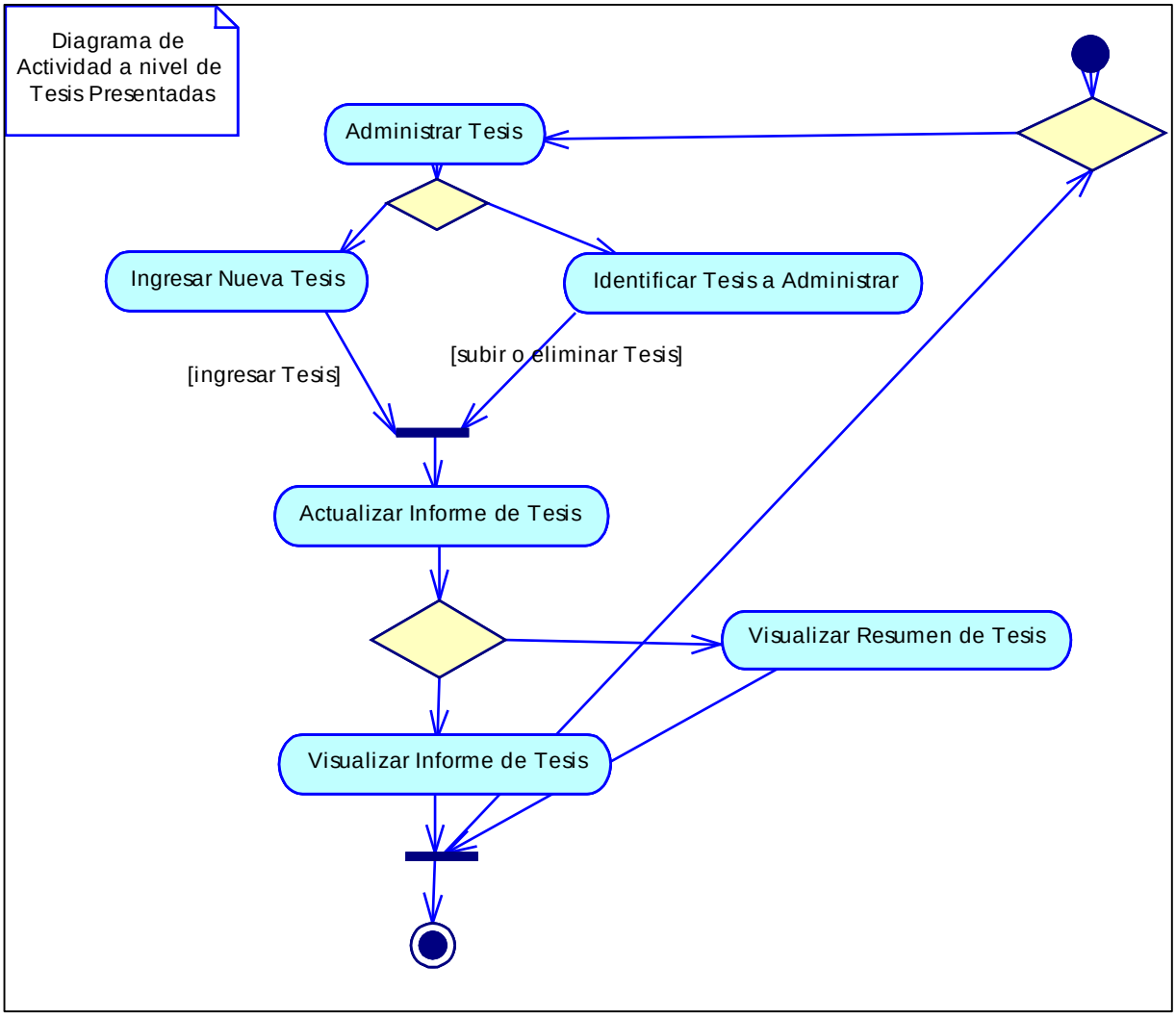


Figura 5.92 Diagrama de Actividad a nivel de Tesis Presentadas

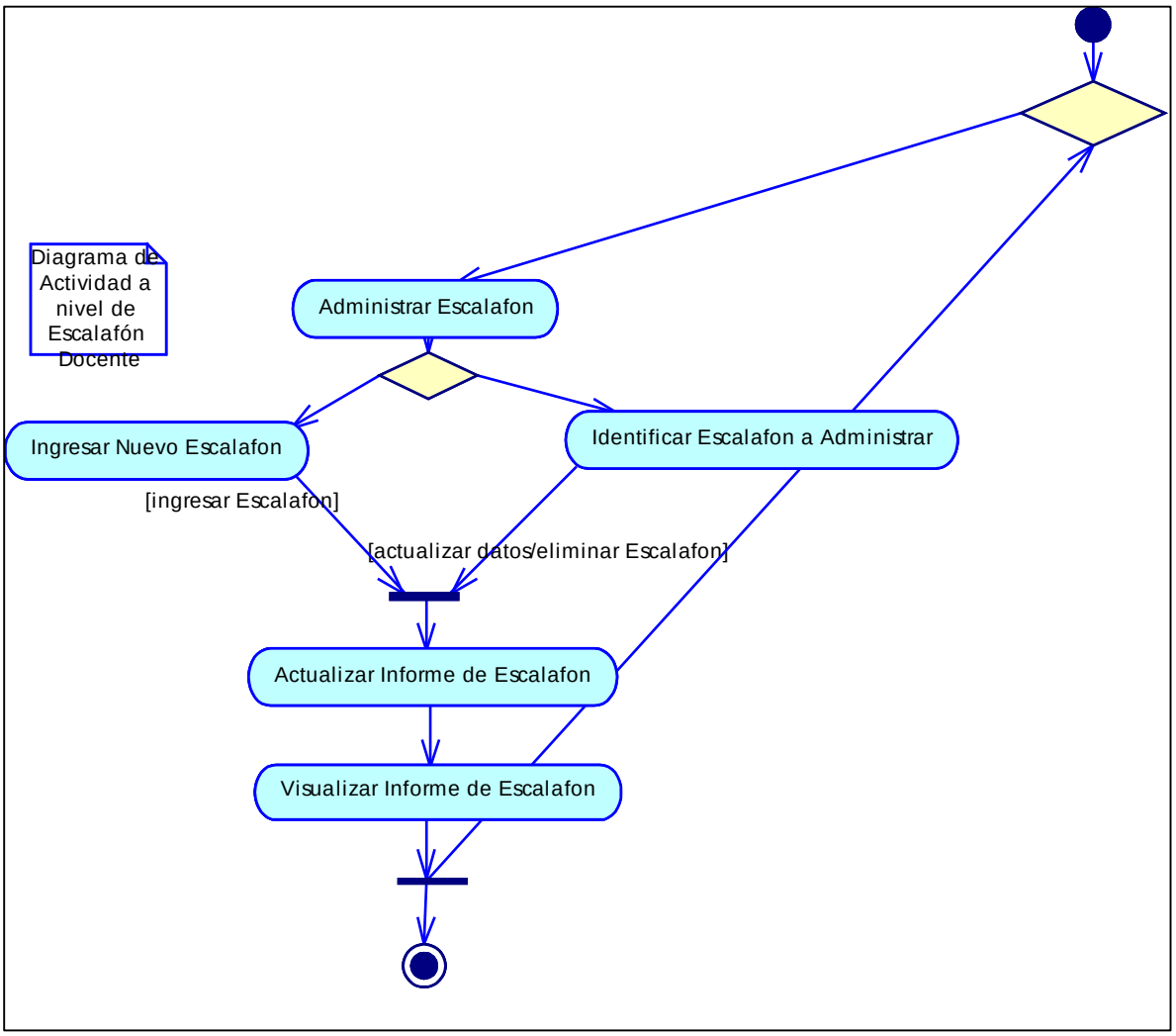


Figura 5.93 Diagrama de Actividad a nivel de Escalafón Docente

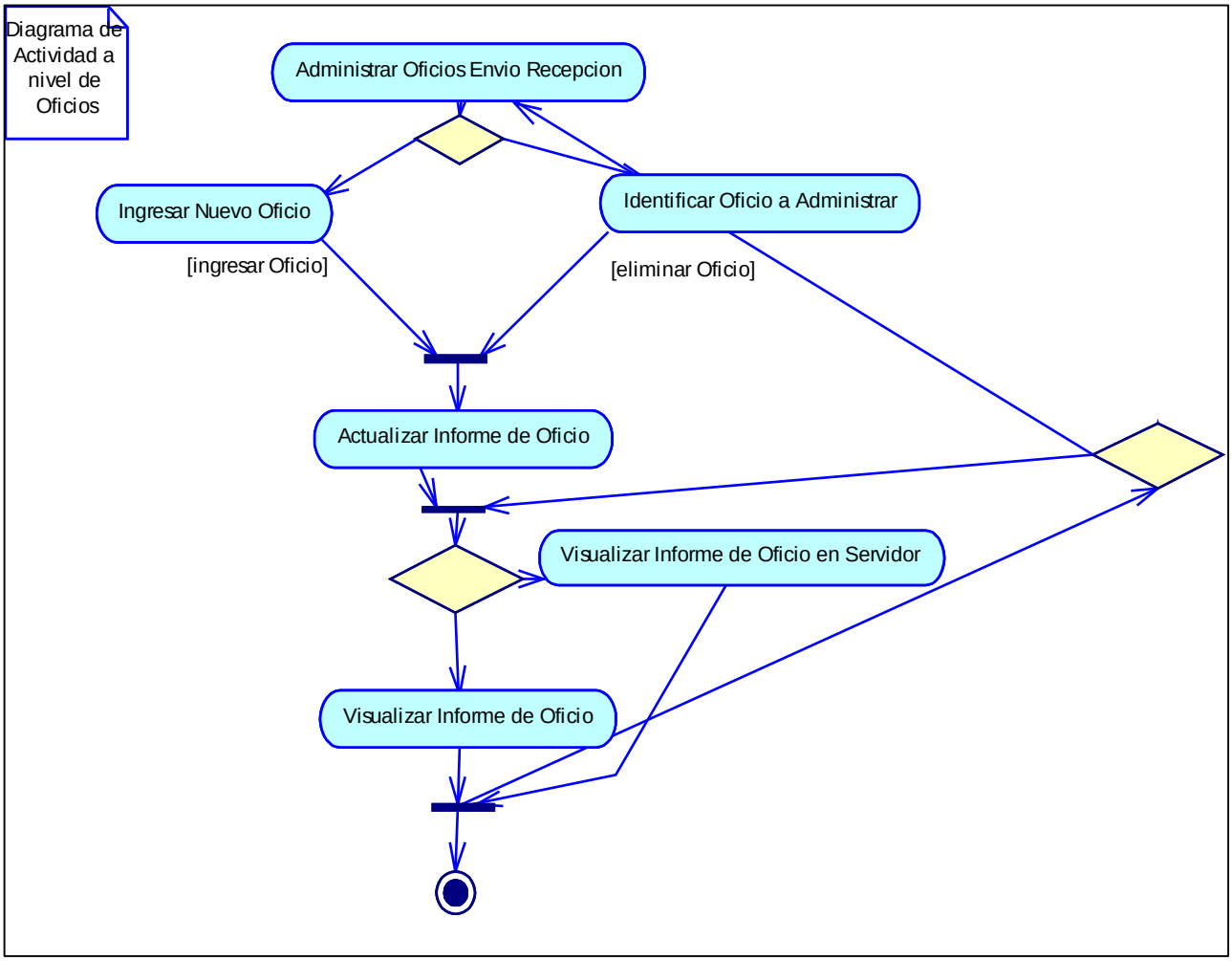


Figura 5.94 Diagrama de Actividad a nivel de Oficios

5.5.8. Diagrama de Flujo de Datos

El flujo de Datos que va a tener el Sistema nos da una visión general del funcionamiento del sistema y la interactuación de los módulos entre ellos.

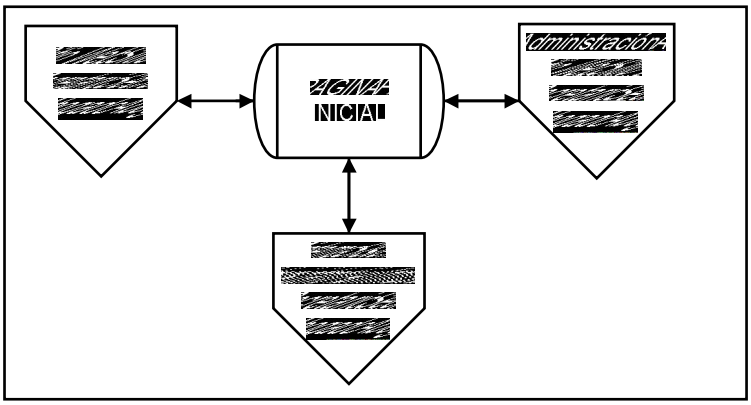


Figura 5.95 DFD Modulo Inicio

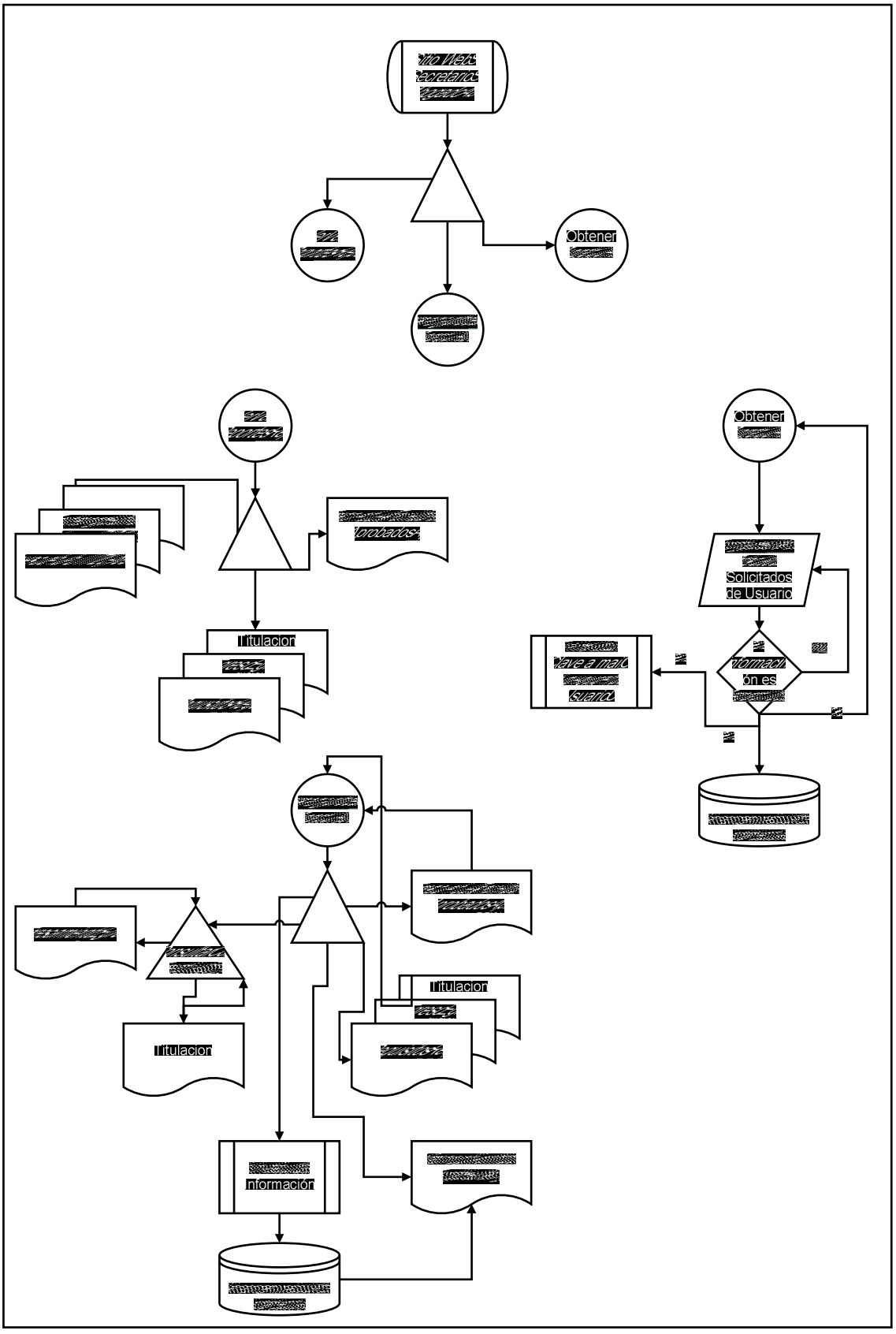


Figura 5.96 DFD Módulo Sitio Web

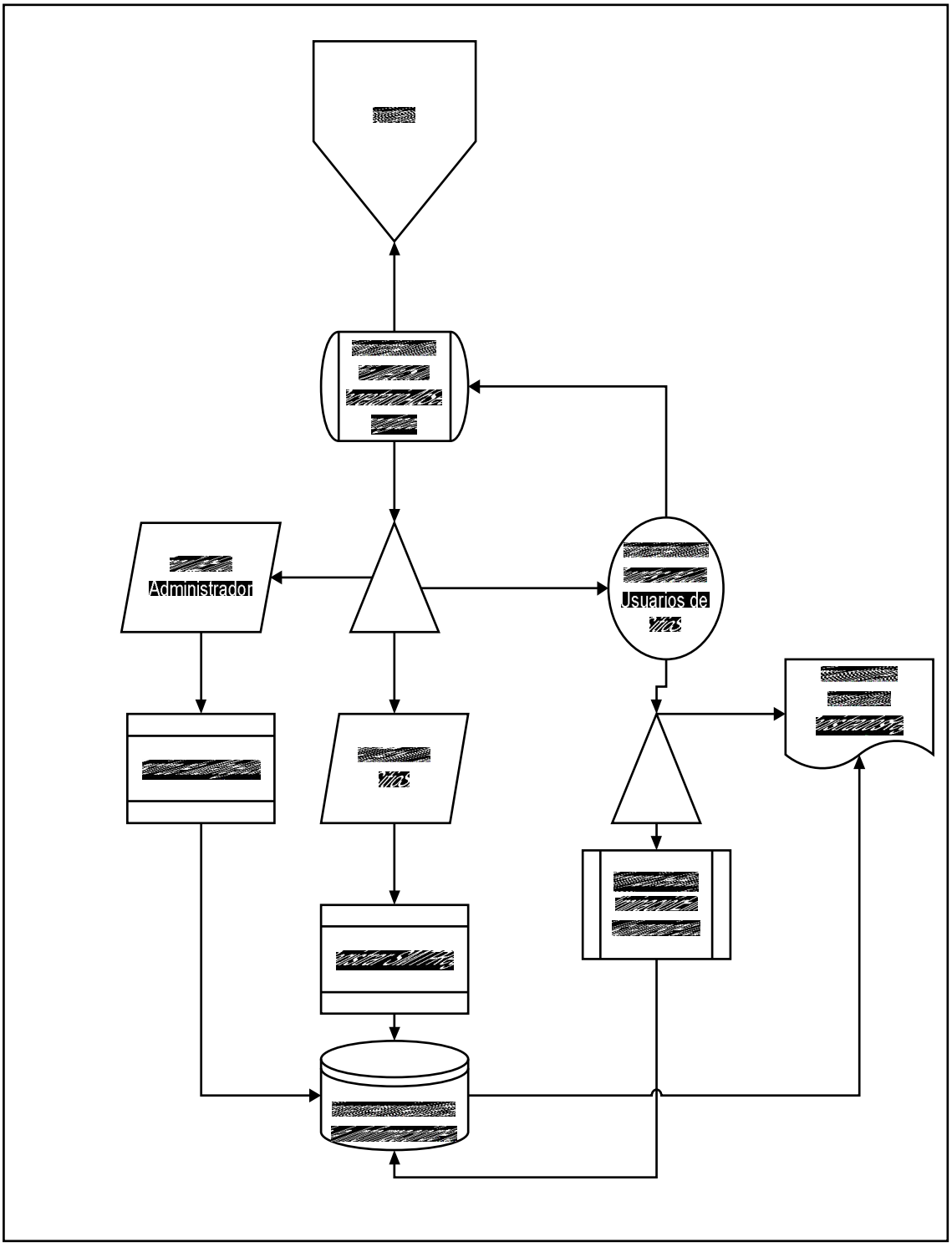


Figura 5.97 DFD Módulo Administración Sitio Web Sistema Secretario-Abogado

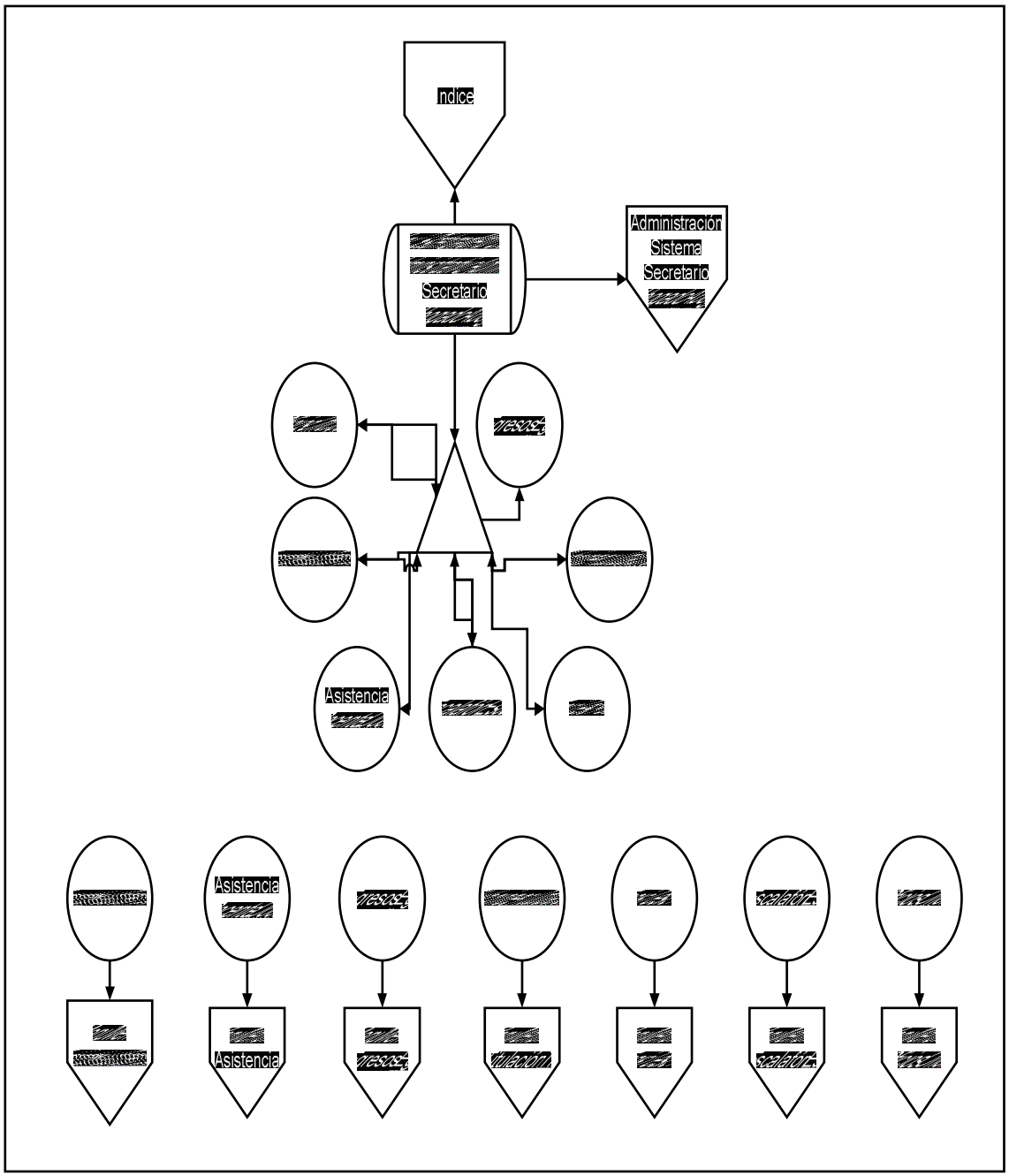


Figura 5.98 DFD Menú Sistema Administrativo Secretario Abogado

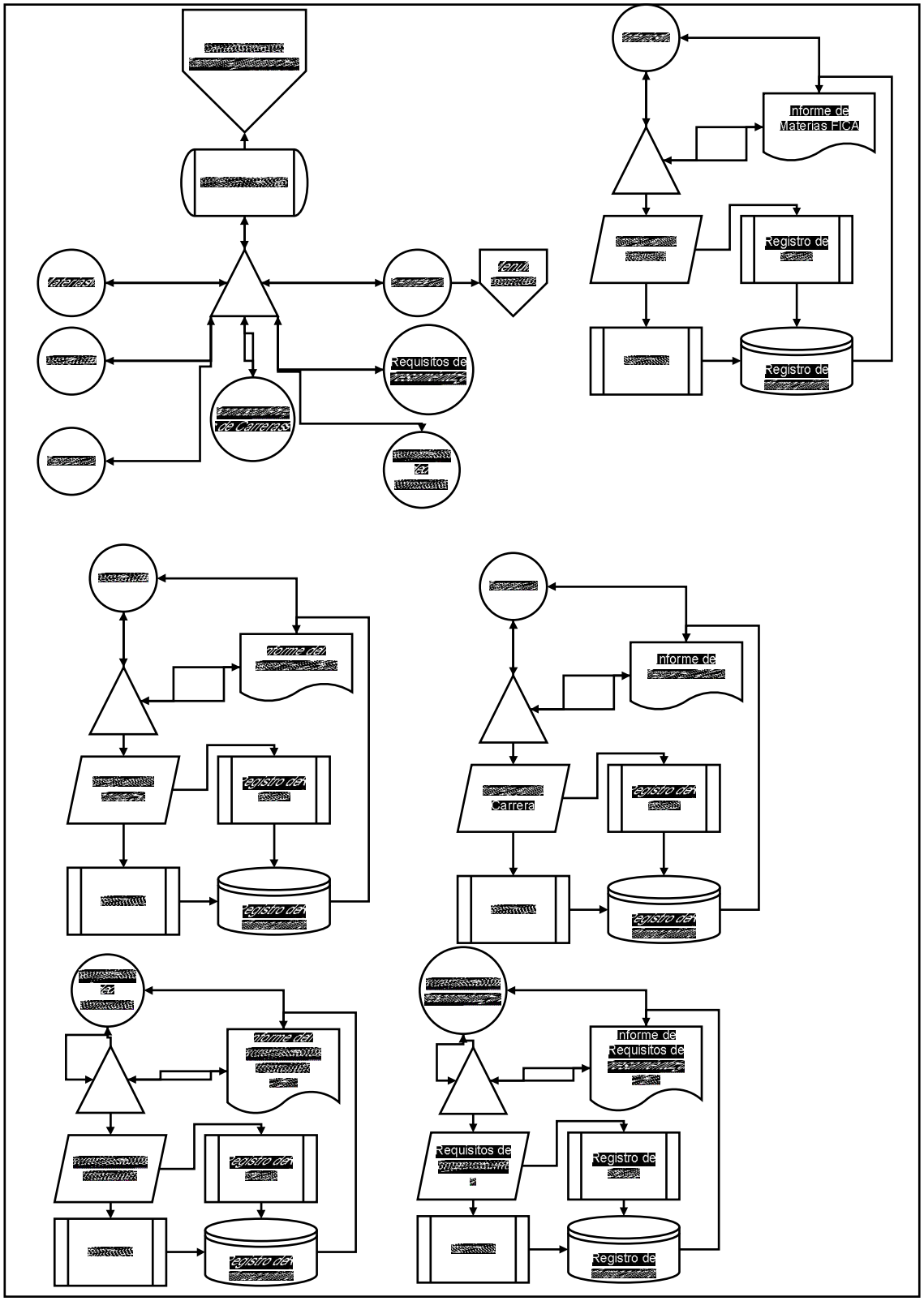


Figura 5.99 DFD Módulo Administración Datos Académicos FICA

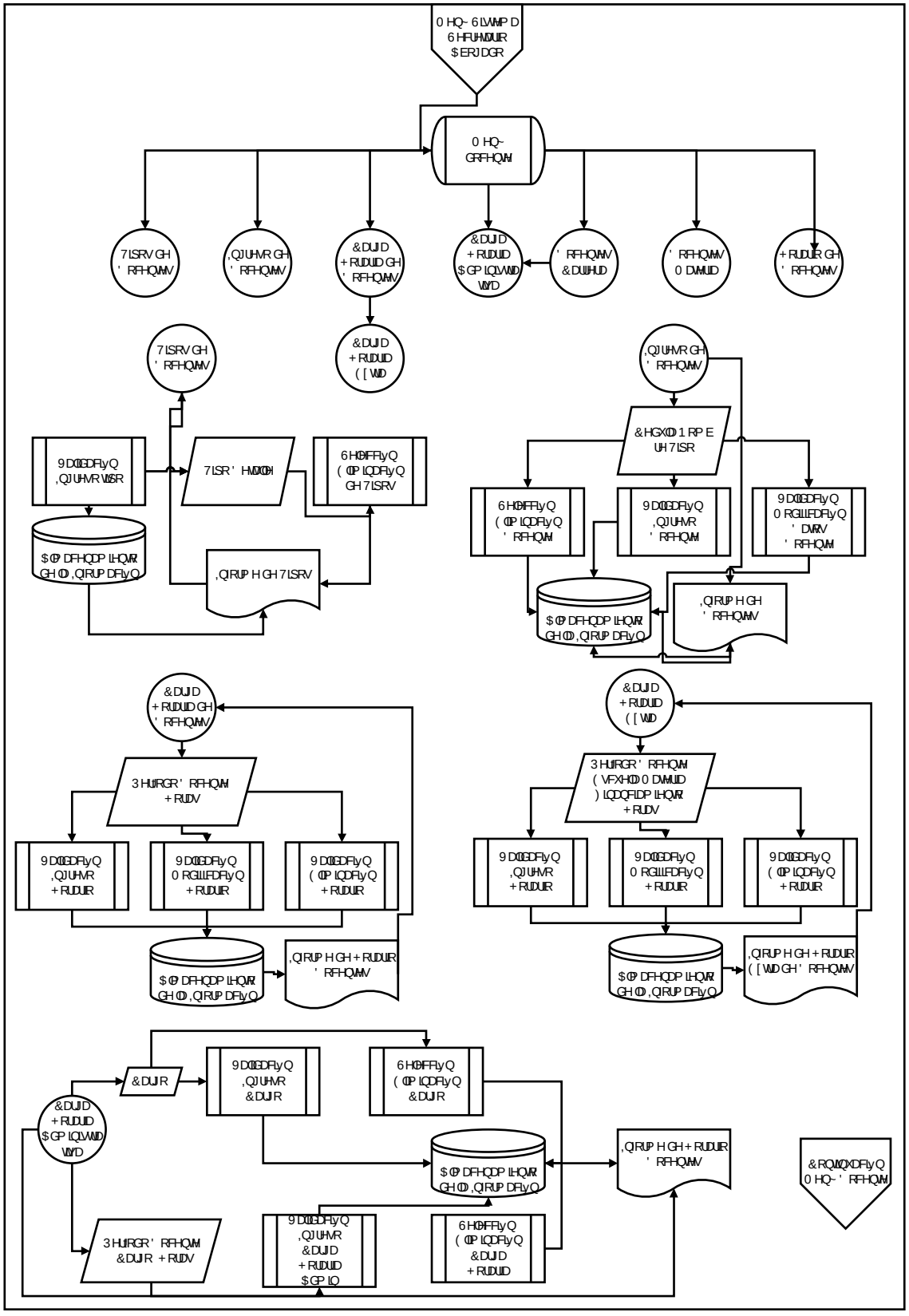


Figura 5.100 DFD Módulo Administración Datos Académicos FICA

(Menú Docente primera parte)

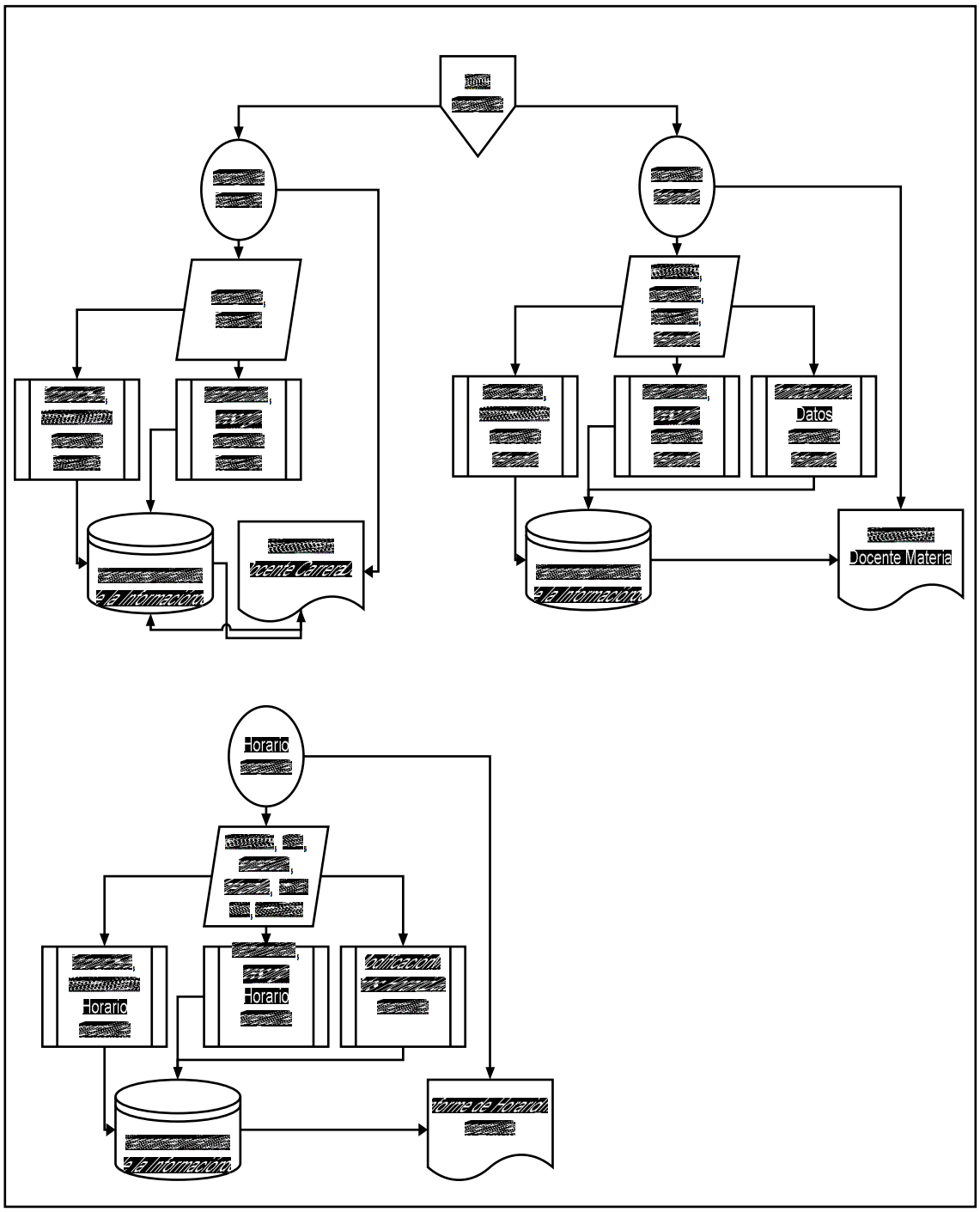


Figura 5.101 DFD Módulo Administración Datos Académicos FICA

(Menú Docente segunda parte)

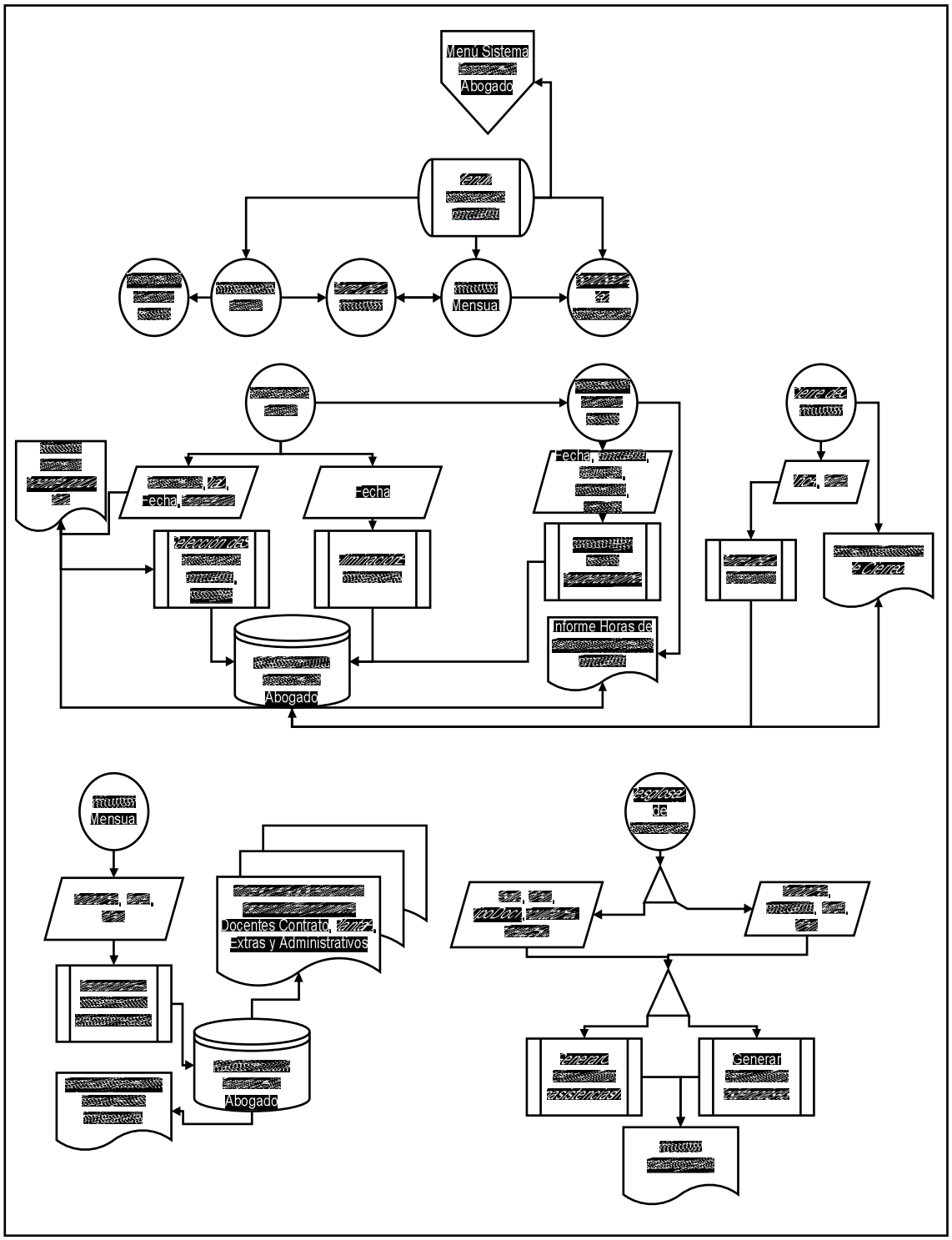


Figura 5.102 DFD Módulo Administración de Asistencia Docente

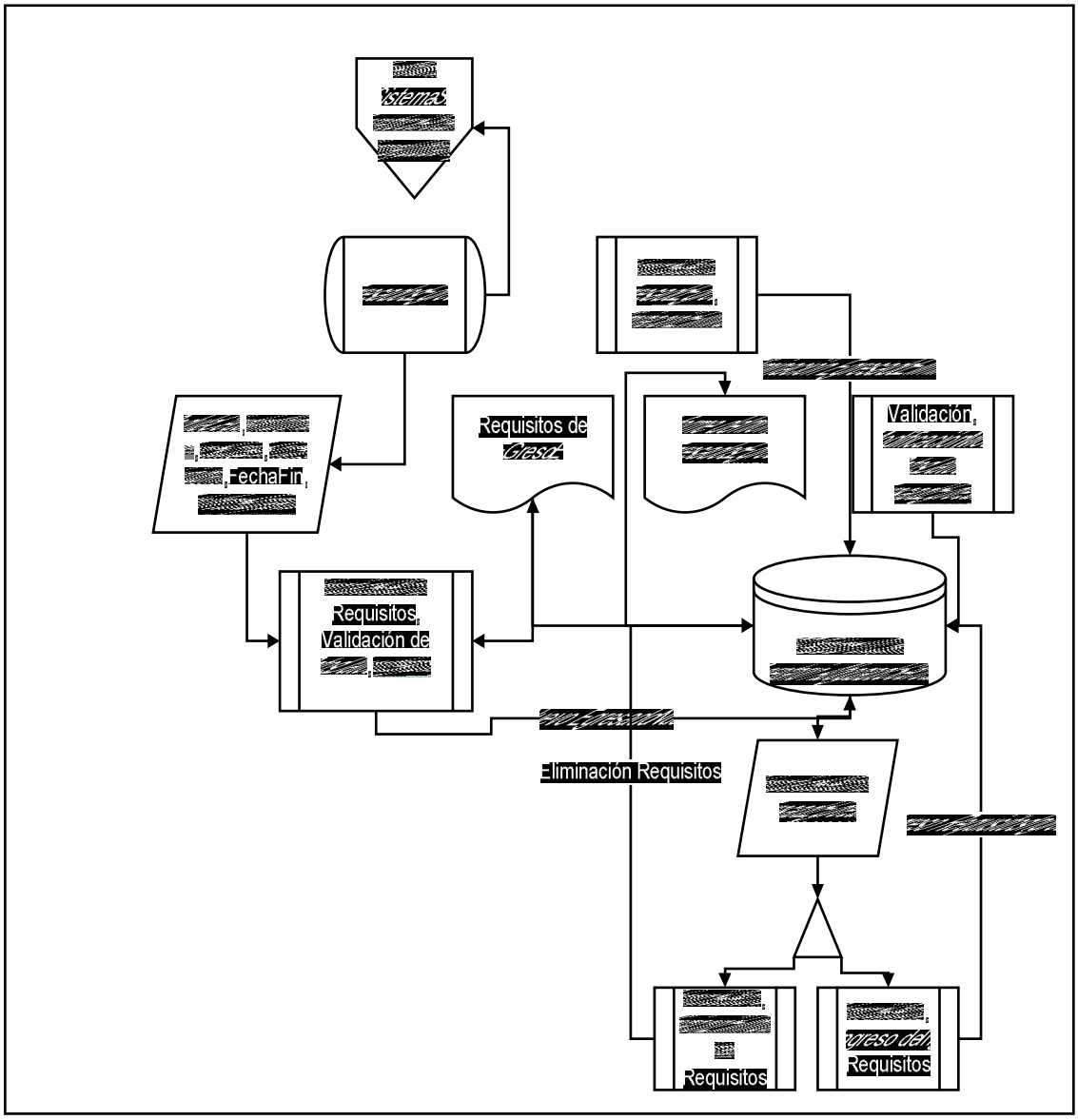


Figura 5.103 DFD Módulo Administración de Egresamiento

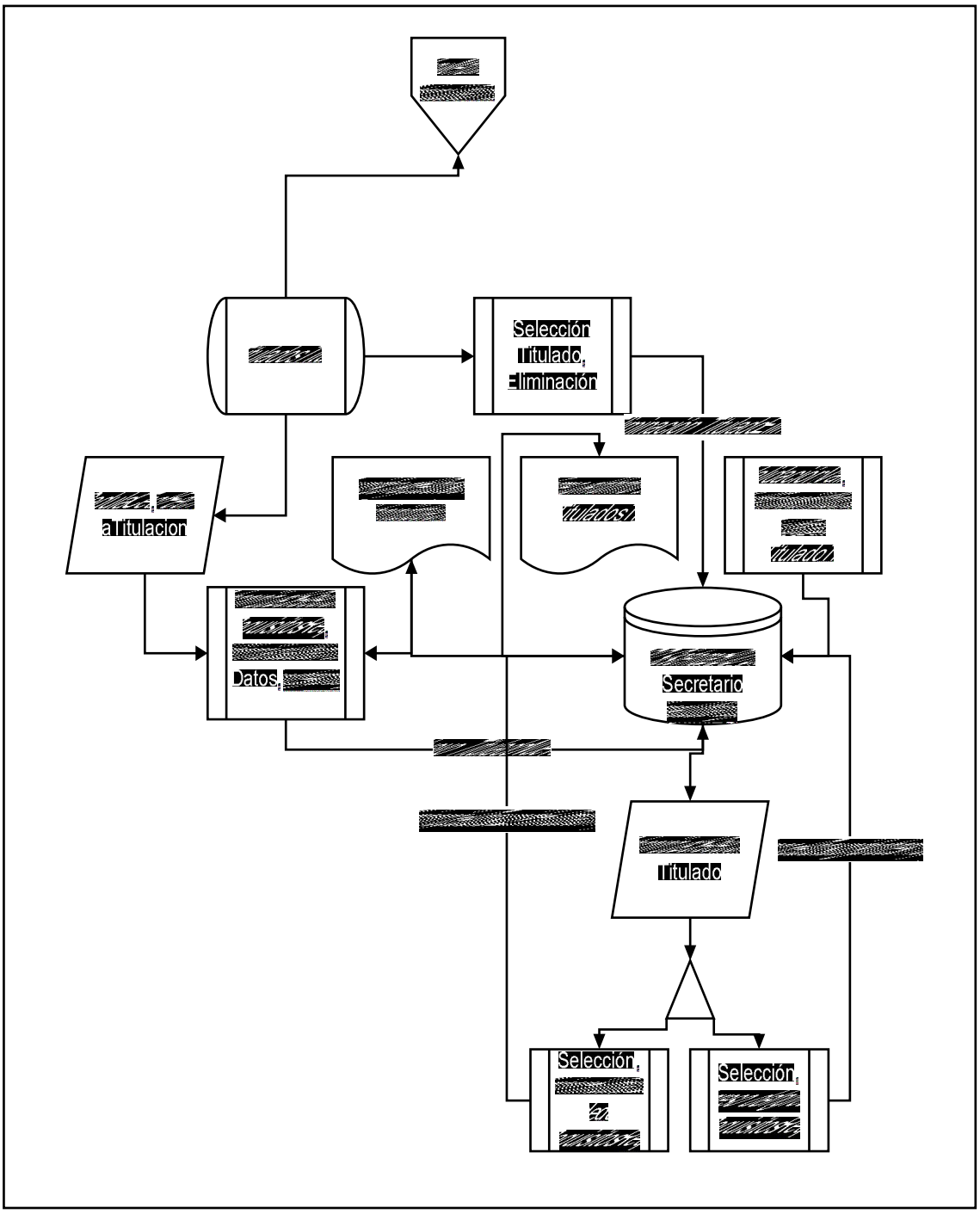


Figura 5.104 DFD Módulo Administración de Titulaciones

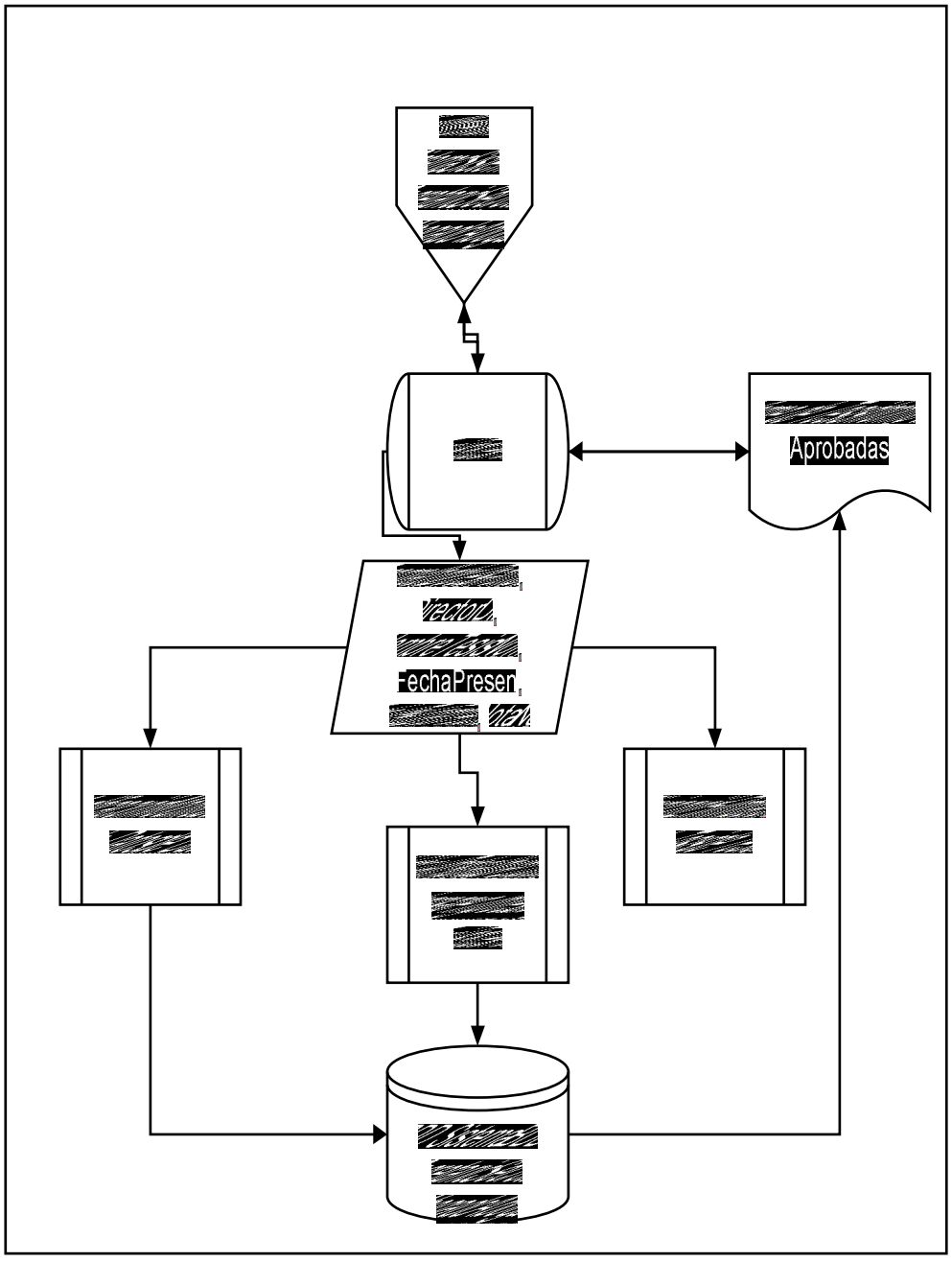


Figura 5.105 DFD Módulo Administración de Tesis Aprobadas

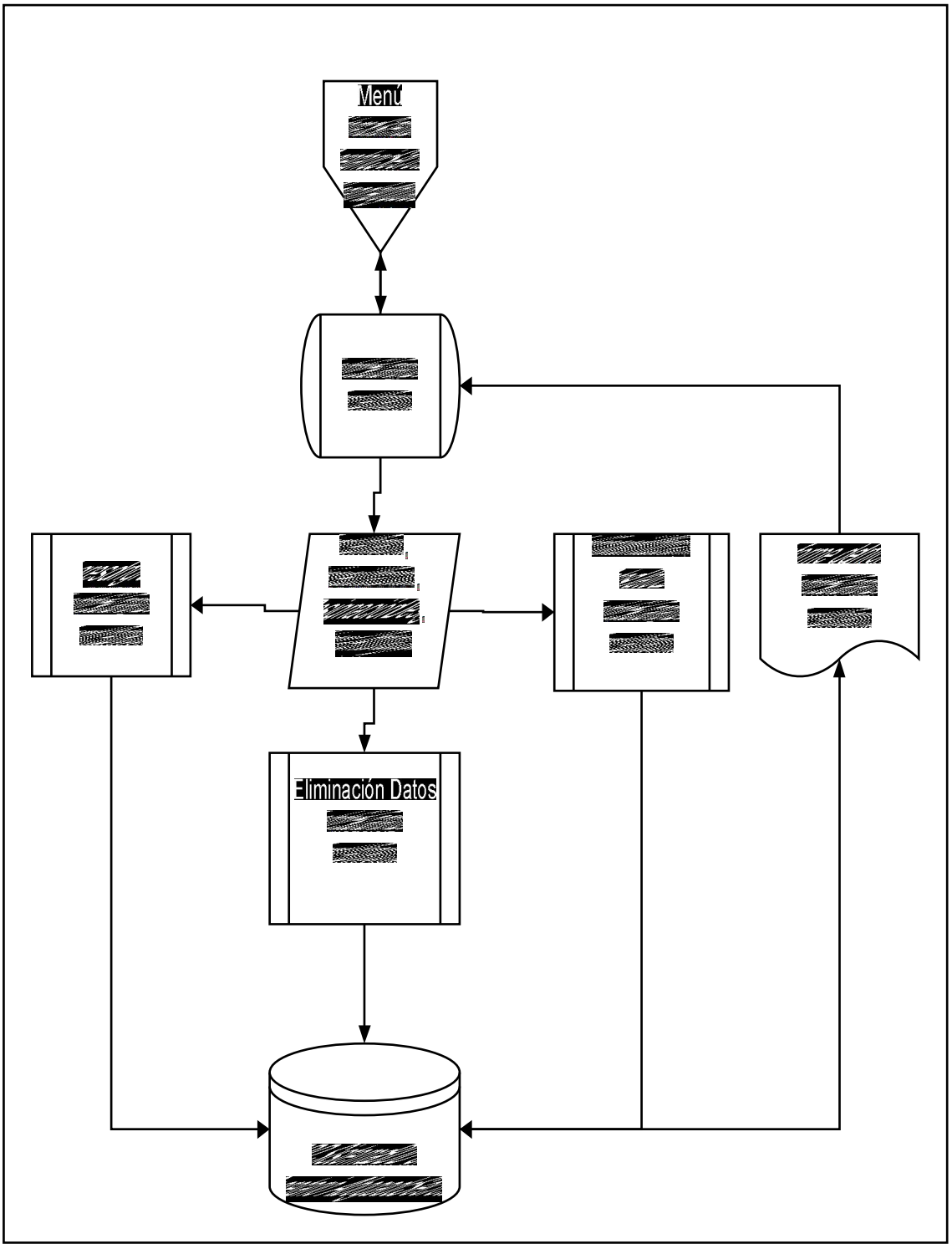


Figura 5.106 DFD Módulo Administración Información de Escalafón Docente

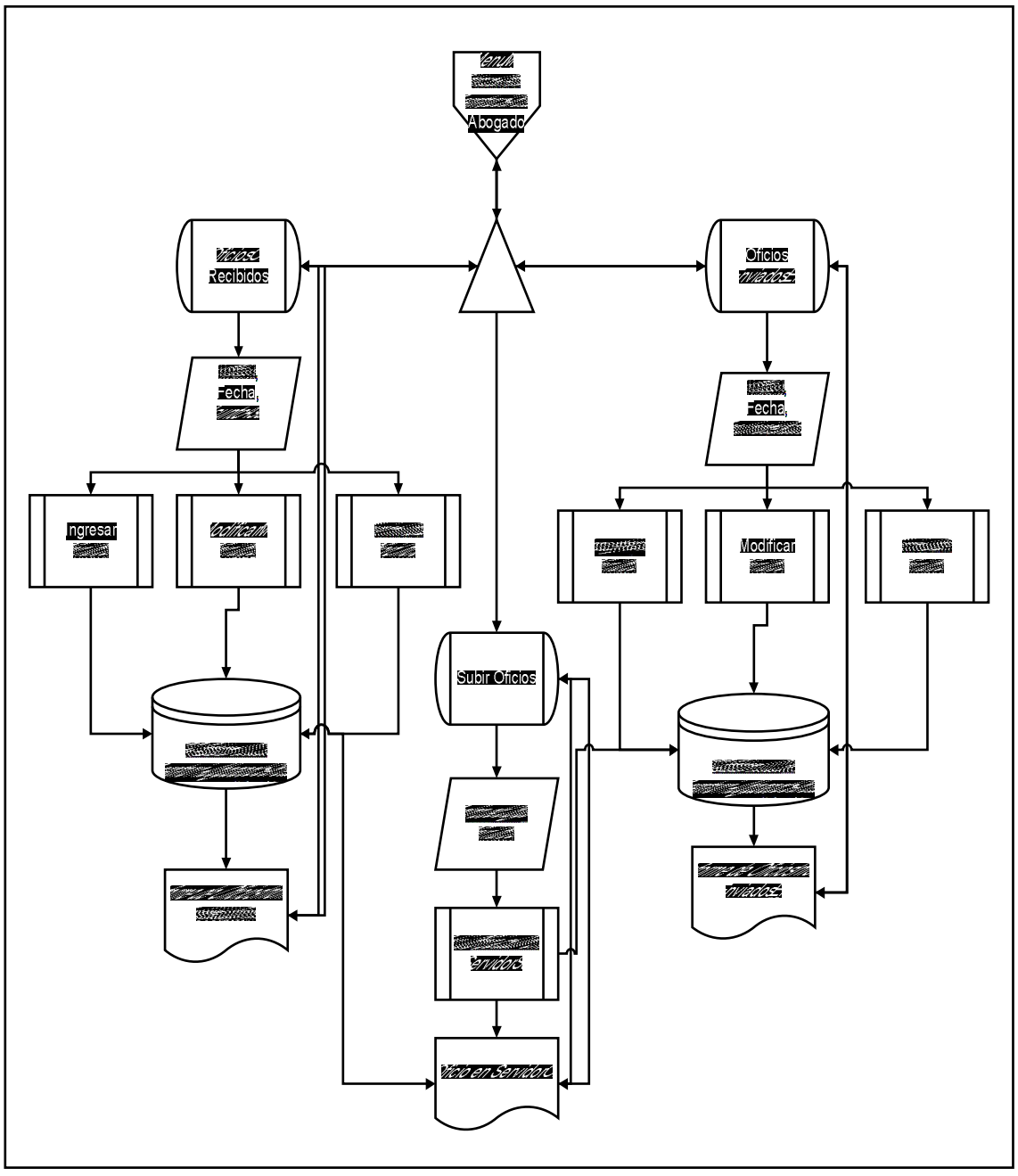


Figura 5.107 DFD Módulo Administración de Oficios Recepción-Envío

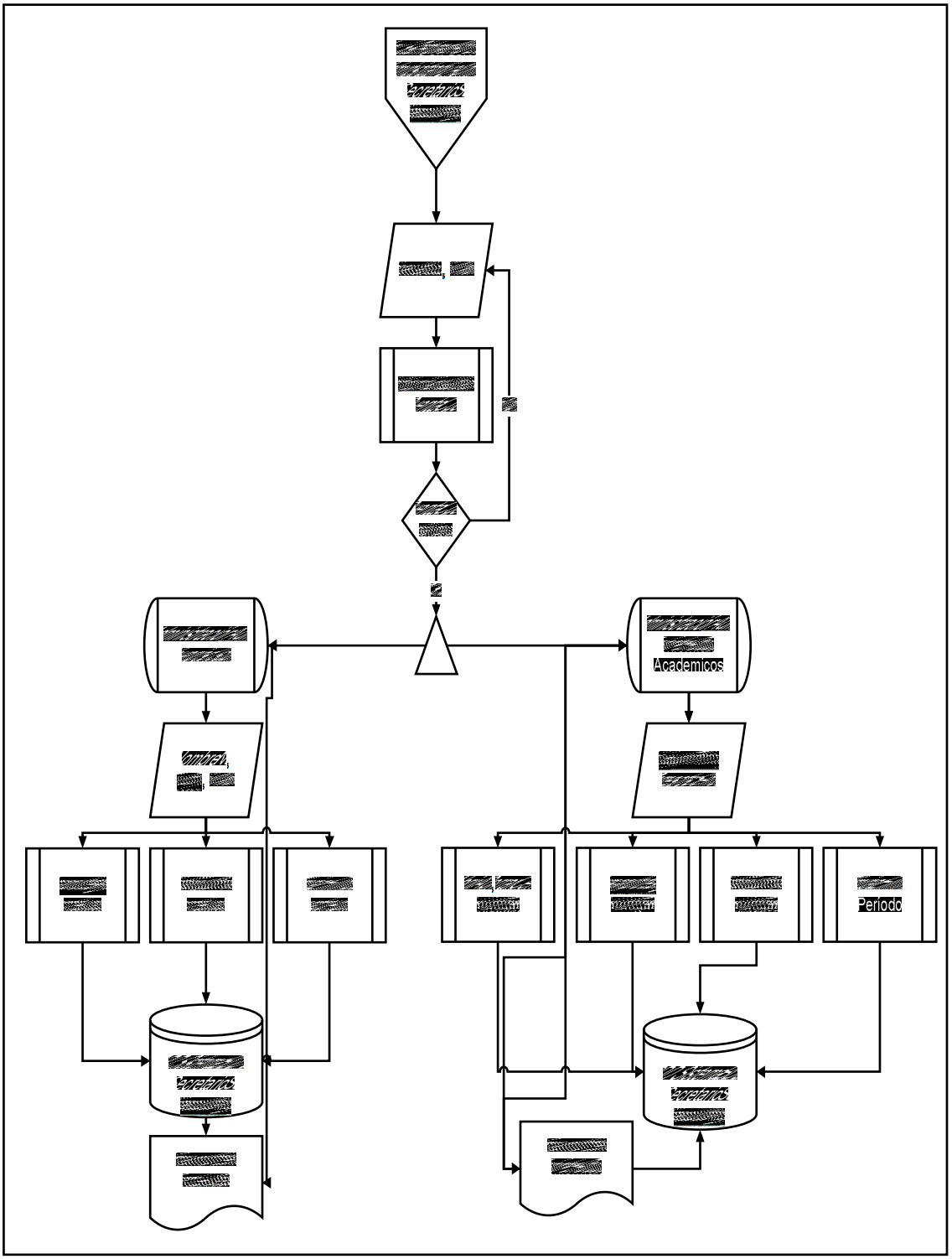


Figura 5.108 DFD Módulo de Administración del Sistema Departamental Secretario-Abogado

5.5.9. Modelos de la Base de Datos

El Diseño de la Base de Datos fue realizada con la herramienta case “Power Designer” e implementada en un Servidor de Datos SQL Server 2005.

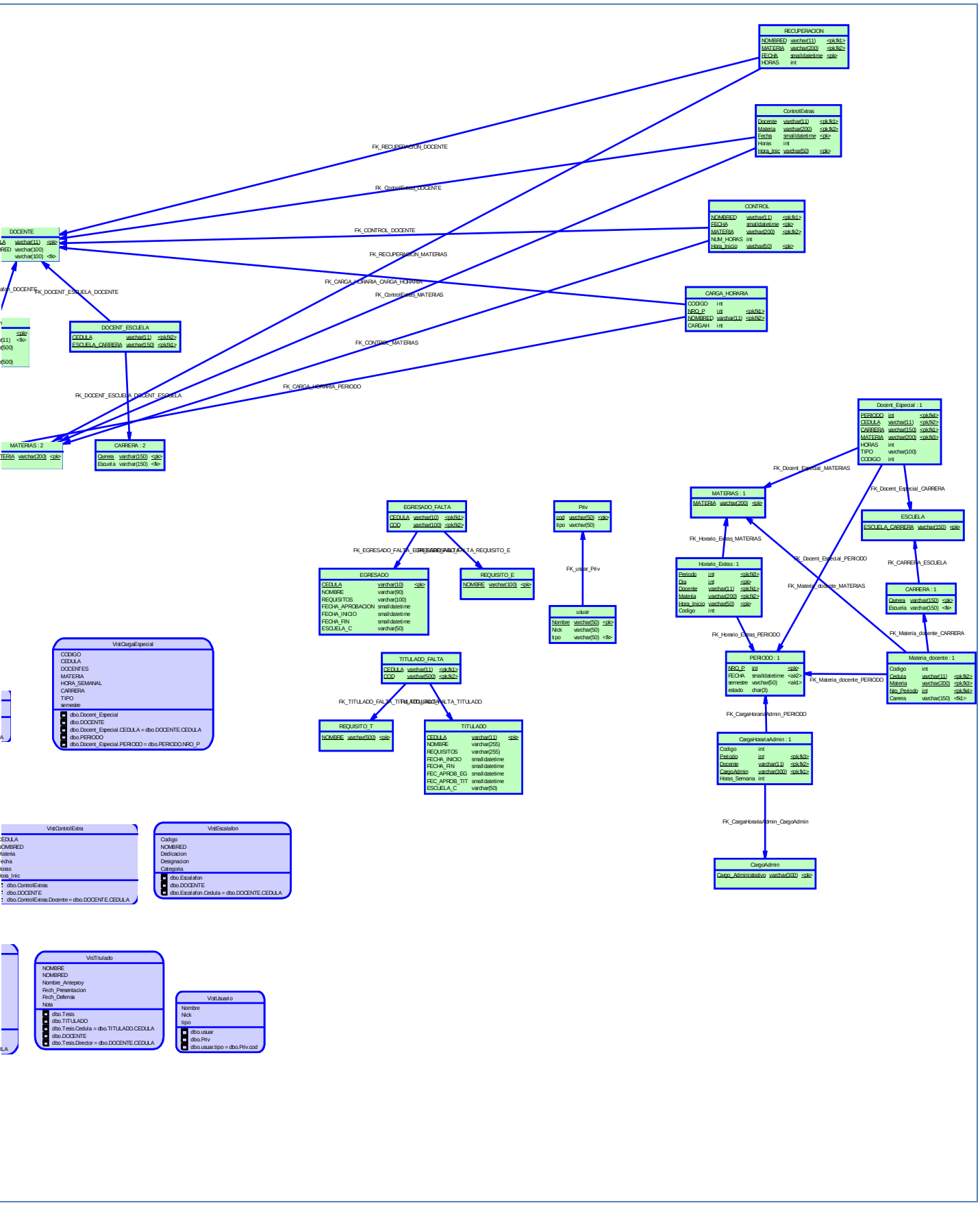
Los diferentes modelos nos permitirán describir las inter relaciones existentes entre las tablas, vistas y su cardinalidad.

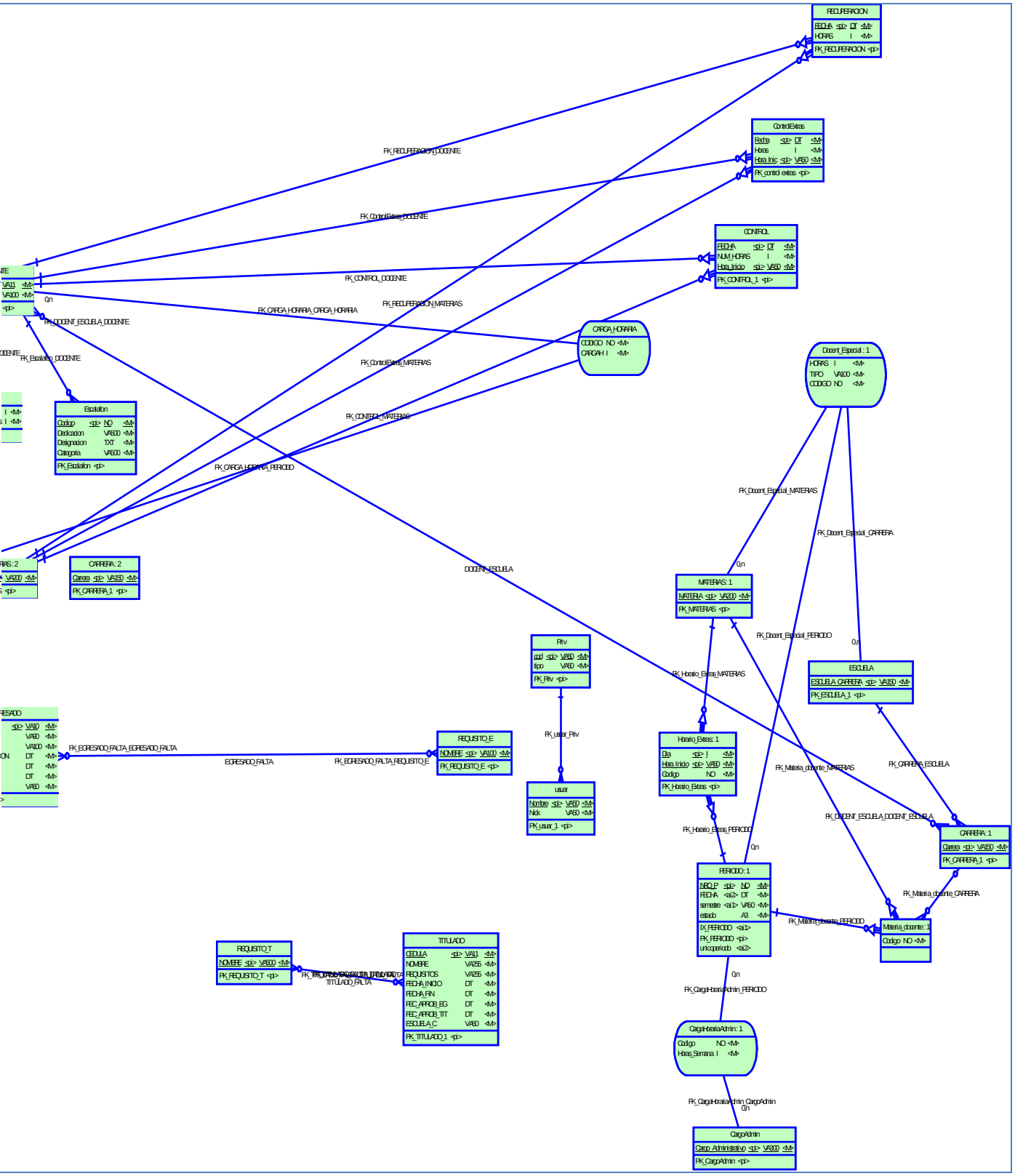
Los modelos a implementarse son los siguientes:

- Modelo Físico.
- Modelo Conceptual.
- Modelo XML.
- Modelo Orientado a Objetos.

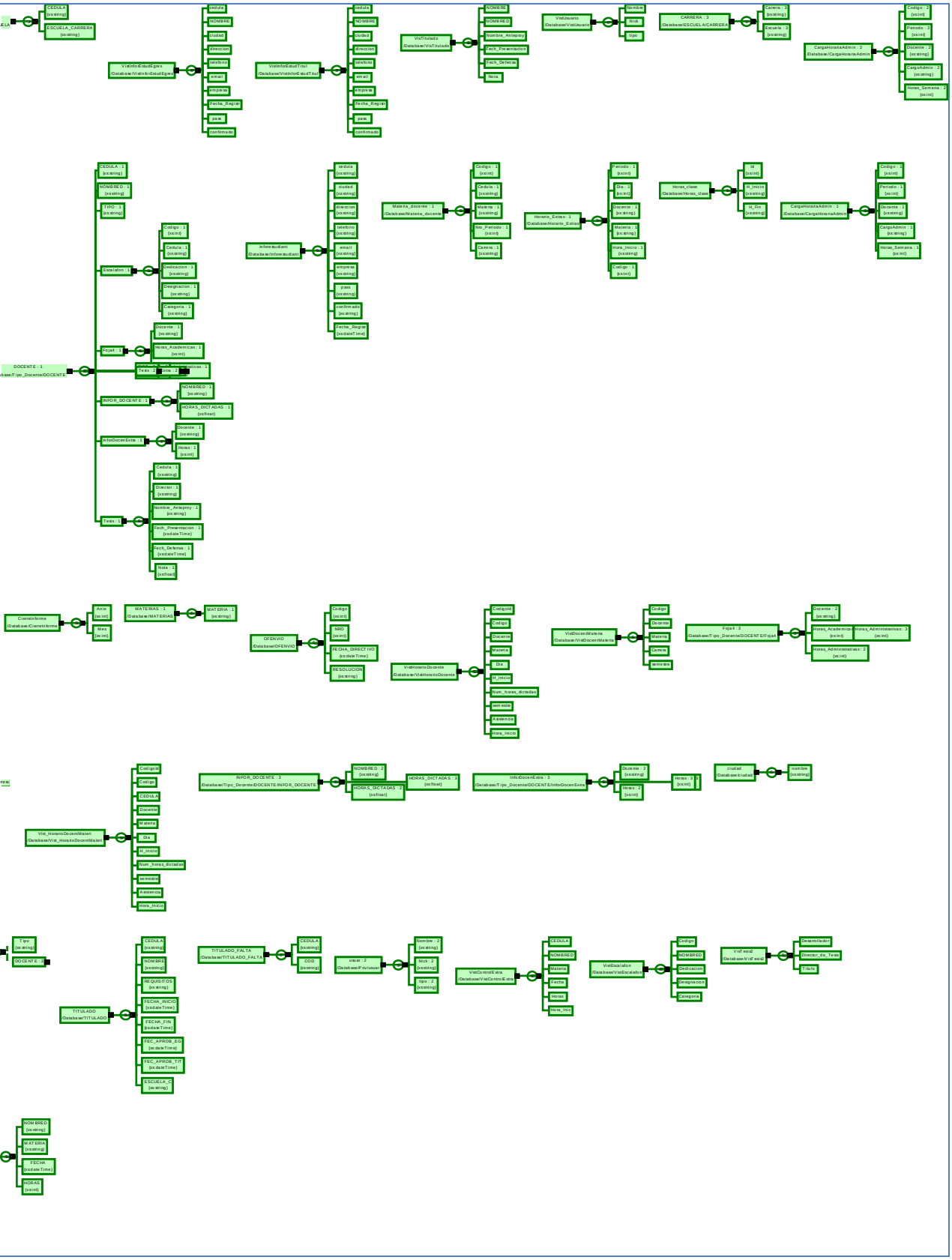
5.5.10. Nomenclatura de los Modelos de Datos

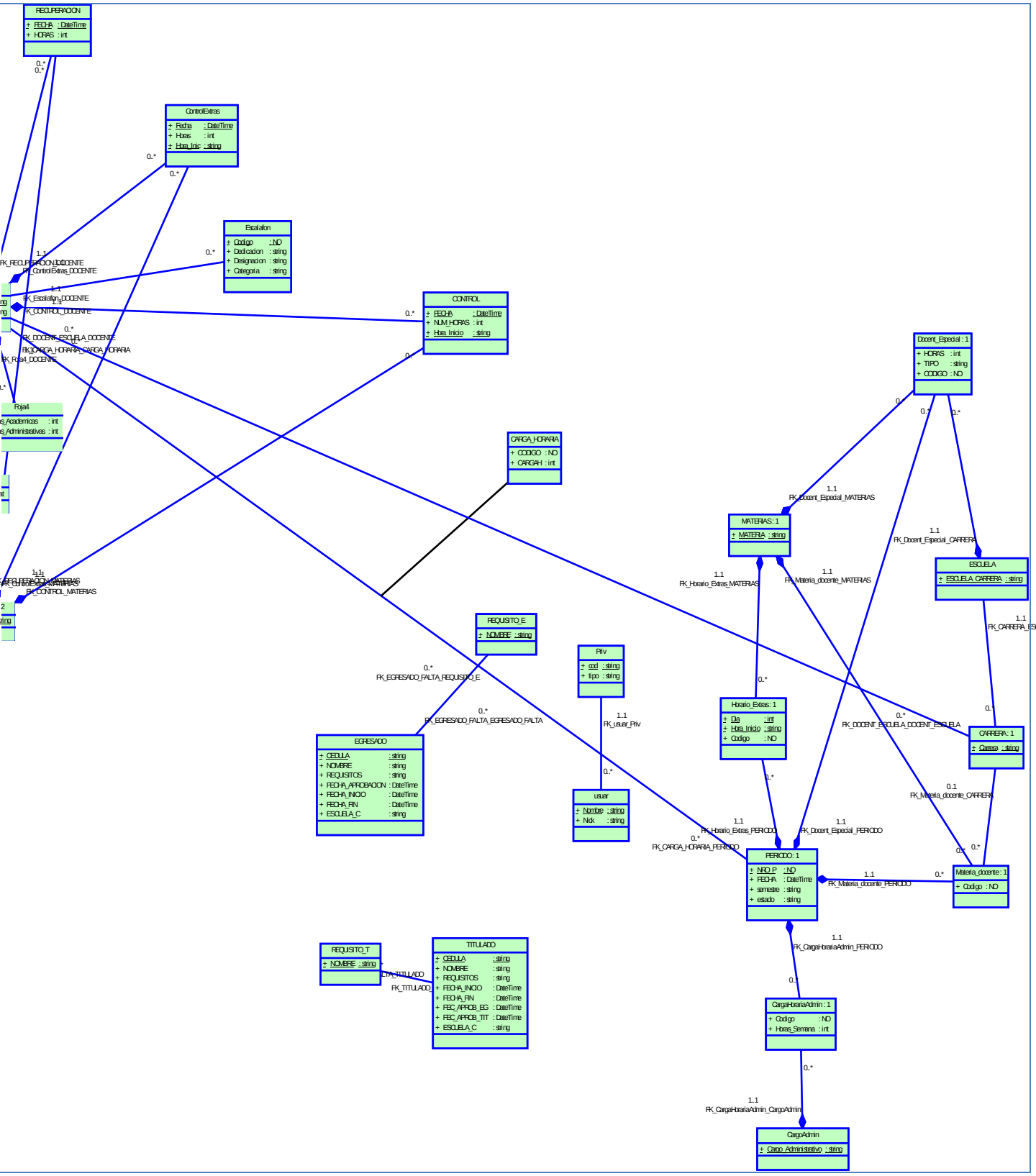
Nomenclatura	Significado
Pk (Primary Key)	Clave Primaria
Fk (Foreign Key)	Clave Foránea
Dbo (Data base Object)	Objeto de Base de Datos
Ix (Index)	Indexado
M (Mandatory)	Obligatorio
S (Sequence)	Secuencia
String	Tipo de dato Cadena caracteres
Int	Tipo de dato número entero
Float	Tipo de dato número flotante
Double	Tipo de dato número doble
No	Valor Autonumerico
DateTime	Tipo de Dato Fecha
1…1	Relación uno a uno
0…*	Relación a varios
0,n	Relación a varios


RECUPERACION
NOMBRED varchar(11) <pk,fk1>
MATERIA varchar(200) <pk,fk2>
FECHA smalldatetime <pk>
HORAS int
ControlExtras
Docente varchar(11) <pk,fk1>
Materia varchar(200) <pk,fk2>
Fecha smalldatetime <pk>
Horas int
Hora_Inic varchar(50) <pk>
CONTROL
NOMBRED varchar(11) <pk,fk1>
FECHA smalldatetime <pk>
MATERIA varchar(200) <pk,fk2>
NUM_HORAS int
Hora_Inicio varchar(50) <pk>
DOCENTE
varchar(11) <pk>
varchar(100)
varchar(100) <fk>
CARGA_HORARIA
CODIGO int
NRO_P int <pk,fk1>
NOMBRED varchar(11) <pk,fk2>
CARGAH int
DOCENT_ESCUELA
CEDULA varchar(11) <pk,fk2>
ESCUELA_CARRERA varchar(150) <pk,fk1>
MATERIAS : 2
varchar(200) <pk>
CARRERA : 2
Carrera varchar(150) <pk>
Escuela varchar(150) <fk>
FK_RECUPERACION_DOCENTE
FK_ControlExtras_DOCENTE
FK_CONTROL_DOCENTE
FK_RECUPERACION_MATERIAS
FK_CARGA_HORARIA_CARGA_HORARIA
FK_ControlExtras_MATERIAS
FK_DOCENT_ESCUELA_DOCENTE
FK_CONTROL_MATERIAS
FK_CARGA_HORARIA_PERIODO
FK_DOCENT_ESCUELA_DOCENT_ESCUELA
Docent_Especial : 1
PERIODO int <pk,fk4>
CEDULA varchar(11) <pk,fk2>
CARRERA varchar(150) <pk,fk3>
MATERIA varchar(200) <pk,fk1>
HORAS int
TIPO varchar(100)
CODIGO int
FK_Docent_Especial_MATERIAS
FK_Docent_Especial_CARRERA
FK_Docent_Especial_PERIODO
MATERIAS : 1
MATERIA varchar(200) <pk>
ESCUELA
ESCUELA_CARRERA varchar(150) <pk>
FK_Horario_Extras_MATERIAS
FK_CARRERA_ESCUELA
Horario_Extras : 1
Periodo int <pk,fk3>
Dia int <pk>
Docente varchar(11) <pk,fk1>
Materia varchar(200) <pk,fk2>
Hora_Inicio varchar(50) <pk>
Codigo int
FK_Materia_docente_MATERIAS
CARRERA : 1
Carrera varchar(150) <pk>
Escuela varchar(150) <fk>
FK_Horario_Extras_PERIODO
FK_Materia_docente_CARRERA
PERIODO : 1
NRO_P int <pk>
FECHA smalldatetime <alk>
semestre varchar(50) <alk>
estado char(3)
Materia_docente : 1
Codigo int
Cedula varchar(11) <pk,fk2>
Materia varchar(200) <pk,fk3>
Nro_Periodo int <pk,fk4>
Carrera varchar(150) <fk1>
FK_Materia_docente_PERIODO
FK_CargaHorariaAdmin_PERIODO
CargaHorariaAdmin : 1
Codigo int
Periodo int <pk,fk3>
Docente varchar(11) <pk,fk2>
CargoAdmin varchar(300) <pk,fk1>
Horas_Semana int
FK_CargaHorariaAdmin_CargoAdmin
CargoAdmin
Cargo_Administrativo varchar(300) <pk>
EGRESADO_FALTA
CEDULA varchar(10) <pk,fk1>
COD varchar(100) <pk,fk2>
FK_EGRESADO_FALTA_EGRESADO
FK_EGRESADO_FALTA_REQUISITO_E
EGRESADO
CEDULA varchar(10) <pk>
NOMBRE varchar(90)
REQUISITOS varchar(100)
FECHA_APROBACION smalldatetime
FECHA_INICIO smalldatetime
FECHA_FIN smalldatetime
ESCUELA_C varchar(50)
REQUISITO_E
NOMBRE varchar(100) <pk>
Priv
cod varchar(50) <pk>
tipo varchar(50)
FK_usuar_Priv
usuar
Nombre varchar(50) <pk>
Nick varchar(50)
tipo varchar(50) <fk>
TITULADO_FALTA
CEDULA varchar(11) <pk,fk1>
COD varchar(500) <pk,fk2>
FK_TITULADO_FALTA_TITULADO
REQUISITO_T
NOMBRE varchar(500) <pk>
TITULADO
CEDULA varchar(11) <pk>
NOMBRE varchar(255)
REQUISITOS varchar(255)
FECHA_INICIO smalldatetime
FECHA_FIN smalldatetime
REC_APROB_EG smalldatetime
REC_APROB_TIT smalldatetime
ESCUELA_C varchar(50)
VistCargaEspecial
CODIGO
CEDULA
DOCENTES
MATERIA
HORA_SEMANAL
CARRERA
TIPO
semestre
dbo.Docent_Especial
dbo.DOCENTE
dbo.Docent_Especial.CEDULA = dbo.DOCENTE.CEDULA
dbo.PERIODO
dbo.Docent_Especial.PERIODO = dbo.PERIODO.NRO_P
VistControlExtra
dbo.ControlExtras
dbo.DOCENTE
dbo.ControlExtras.Docente = dbo.DOCENTE.CEDULA
VistEscalafon
Codigo
NOMBRED
Dedicacion
Designacion
Categoria
dbo.Escalafon
dbo.DOCENTE
dbo.Escalafon.Cedula = dbo.DOCENTE.CEDULA
VistTitulado
NOMBRE
NOMBRED
Nombre_Anteproy
Fech_Presentacion
Fech_Defensa
Nota
dbo.Tesis
dbo.TITULADO
dbo.Tesis.Cedula = dbo.TITULADO.CEDULA
dbo.DOCENTE
dbo.Tesis.Director = dbo.DOCENTE.CEDULA
VistUsuario
Nombre
Nick
tipo
dbo.usuar
dbo.Priv
dbo.usuar.tipo = dbo.Priv.cod



a Base de Datos Secretario Abogado





a Base de Datos del Sistema Secretario Abogado

5.5.10.1. Descripción de la Base de Datos

Autoinfor:

Contiene Información de Administración para el Web Site Informativo de Egresados y Titulados (Mail de Administrador y dirección de SMTP).

CARGA_HORARIA:

Contiene la Carga Horaria Docente clasificada por períodos.

CargaHorariaAdmin:

Contiene la Carga Horaria de los docentes con funciones Administrativas.

CargoAdmin:

Contiene un Detalle de Cargo Administrativo.

CARRERA:

Contiene las Carreras Existentes en la FICA.

CierreInforme:

Contiene especificaciones de mes y año en el que se debe cerrar la edición de informes de control de Asistencia Docente.

Ciudad:

Contiene nombres de las ciudades del Ecuador en el que se pueden encontrar residiendo egresados o titulados de la FICA.

CONTROL:

Contiene la información de control de Asistencia Diario de los Docente de la FICA.

ControlExtras:

Contiene el control de las horas extras laboradas por los docentes.

DOCENT_ESCUELA:

Especifica la información de relación existente entre Docente y la Escuela en la que dicta clases.

Docent_Especial:

Contiene la asignación de Horas Extras asignadas a Docentes determinados, la asignación está clasificada por el tipo de financiamiento de una carrera y el período.

DOCENTE:

Información básica de docentes que imparten clases en la FICA.

EGRESADO:

Información curricular de los ex – estudiantes de la FICA que se encuentran en trámite de Egresamiento o ya son Egresados.

EGRESADO_FALTA:

Contiene la información de los requisitos faltantes en el trámite de egreso de los ex estudiantes de la FICA.

Escalafon:

Contiene información del Escalafón de los Docente de la FICA.

ESCUELA:

Contiene los nombres de las escuelas existentes en la FICA.

Foja4:

Tiene un desglose de las horas laboradas tanto académicas como administrativas.

Horario_Docente:

Aloja el horario de los docentes asignado en un distributivo para un determinado período.

Horario_Extras:

En esta tabla se detalla el horario de los docentes con respecto a las materias consideradas como extras a su carga horaria en el período.

Horas_clase:

Tiene la estructuración estándar de horas clase utilizadas por el Sistema Académico.

INFOR_DOCENTE:

Alberga el reporte mensual generado a petición en un determinado período académico y una fecha específica.

InforDocenExtra

Contiene el informe mensual de horas extras laboradas por los docentes en un determinado mes.

Inforestudiant:

Contiene los datos de registro de usuarios del Sitio Web Informativo de Egresados y Titulados.

Materia_docente:

Tiene la información relacional entre el Docente y la Materia que imparte.

MATERIAS:

Tiene los nombres de materias existentes en el pensum académico de la FICA.

OFENVIO:

Alberga la información resumida de los oficios enviados desde el dpto. Secretario – Abogado.

OFRECIBIDO:

Alberga la información resumida de los oficios receptados en el dpto. Secretario – Abogado.

PERIODO:

Contiene la información de los períodos académicos mediante el cual se controla los procesos y datos usados en el aplicativo Web.

Priv:

Alberga los privilegios de usuario con el que se controla una parte de la seguridad del Sistema Administrativo Secretario – Abogado.

RECUPERACION:

Contiene la información de horas clases recuperadas por los docentes en caso de no haber sido registrado en el control diario, horas asignadas por concepto de exámenes o desarrollo de tareas extra curriculares.

REQUISITO_E:

Esta tabla tiene el registro de los requisitos necesarios para tramitar el Egresamiento.

REQUISITO_T:

Esta tabla tiene el registro de los requisitos necesarios para tramitar la Titulación.

RequisitoLab:

Esta tabla tiene el registro de la lista de materiales a entregar en el Laboratorio de la FICA, con el cual se obtiene un paz y salvo, requisito necesario para tramitar la Titulación.

Responsables:

Contiene la lista de responsables de las carreras existentes en la FICA.

Tesis:

Contiene la información de las tesis aprobadas en la FICA (tesista, director de tesis, tema, calificación, fecha de desarrollo).

Tipo_Docente:

En esta tabla se registra el tipo docente, ya sea éste planta ó contrato.

TITULADO:

Contiene la información de los egresados de la FICA, que están realizando ó han terminado el proceso de titulación.

TITULADO_FALTA:

En esta tabla se registra los requisitos faltantes para completar el trámite de titulación de los egresados de la FICA.

usuar:

Contiene el registro de usuarios o logins del Sistema Administrativo Secretario – Abogado.

5.5.10.2. Diccionario de la Base de Datos

TABLA:
AUTOINFOR

Clave	Nombre de Columna	Tipo de Dato	Detalle
	nombre	varchar(200)	Nombre de la Información
Pk	tipo	varchar(200)	Detalle de la Información

TABLA:
CARGA_HORARIA

Clave	Nombre de Columna	Tipo de Dato	Detalle
	CODIGO	int	Codigo de Carga Horaria
Pk,Fk	NRO_P	int	Nro de Período Académico
Pk,Fk	NOMBRED	varchar(11)	Nro de Cédula Docente
	CARGAH	int	Nro de Horas asignadas para Carga horaria

TABLA:
CargaHorariaAdmin

Clave	Nombre de Columna	Tipo de Dato	Detalle
	Codigo	int	Código de Carga Horaria Administrativa
Pk,Fk	Periodo	int	Nro de Período Académico
Pk,Fk	Docente	varchar(11)	Nro de Cédula Docente
Pk	CargoAdmin	varchar(300)	Cargo Administrativo
	Horas_Semana	int	Horas por Semana

TABLA:
CargoAdmin

Clave	Nombre de Columna	Tipo de Dato	Detalle
Pk	Cargo_Administrativo	varchar(300)	Nombre del Cargo Administrativo

TABLA:
CARRERA

Clave	Nombre de Columna	Tipo de Dato	Detalle
Pk	Carrera	varchar(150)	Nombre de Carrera
Fk	Escuela	varchar(150)	Nombre de Escuela

TABLA:
CierreInforme

Clave	Nombre de Columna	Tipo de Dato	Detalle
Pk	Anio	int	Anio para Cierre de Informe
Pk	Mes	int	Mes para Cierre de Informe

TABLA:
ciudad

Clave	Nombre de Columna	Tipo de Dato	Detalle
Pk	nombre	varchar(50)	Ciudades del Ecuador

TABLA:
CONTROL

Clave	Nombre de Columna	Tipo de Dato	Detalle
Pk,Fk	NOMBRED	varchar(100)	Nro de Cédula Docente
Pk	FECHA	smalldatetime	Fecha de Control de Asistencia
Pk,Fk	MATERIA	varchar(75)	Materia Asistida
	NUM_HORAS	int	Nro de Horas Laboradas
Pk,Fk	Hora_Inicio	varchar(50)	Hora de Inicio de la Actividad

TABLA:
ControlExtras

Clave	Nombre de Columna	Tipo de Dato	Detalle
Pk,Fk	Docente	varchar(11)	Nro de Cédula Docente
Pk,Fk	Materia	varchar(200)	Materia Asistida
Pk	Fecha	smalldatetime	Fecha de Control de Asistencia
	Horas	int	Nro de Horas Laboradas
Pk,Fk	Hora_Inic	varchar(50)	Hora de Inicio de la Actividad

TABLA:
DOCENT_ESCUELA

Clave	Nombre de Columna	Tipo de Dato	Detalle
Pk,Fk	CEDULA	varchar(11)	Nro de Cédula Docente
Pk,Fk	ESCUELA_CARRERA	varchar(150)	Carrera en la que Labora el Docente

TABLA:
Docent_Especial

Clave	Nombre de Columna	Tipo de Dato	Detalle
Pk,Fk	PERIODO	int	Nro de Período Académico
Pk,Fk	CEDULA	varchar(11)	Nro de Cédula Docente
Pk,Fk	CARRERA	varchar(150)	Carrera en la que Labora el Docente
Pk,Fk	MATERIA	varchar(200)	Nombre de Materia Asignada
	HORAS	int	Nro de Horas Asignadas
	TIPO	varchar(100)	Tipo de Docente
	CODIGO	int	Código Docente Especial

TABLA:
DOCENTE

Clave	Nombre de Columna	Tipo de Dato	Detalle
Pk	CEDULA	varchar(11)	Nro de Cédula Docente
	NOMBRED	varchar(100)	Apellidos y nombres de Docente
	TIPO	varchar(100)	Tipo de Docente

TABLA:
EGRESADO

Clave	Nombre de Columna	Tipo de Dato	Detalle
Pk	CEDULA	varchar(10)	Cédula del Egresado
	NOMBRE	varchar(90)	Apellidos y nombres del Egresado
	REQUISITOS	varchar(100)	Estado de Requisitos de Egreso
	FECHA_APROBACION	smalldatetime	Fecha de Aprobación de Egreso
	FECHA_INICIO	smalldatetime	Fecha de Inicio de Estudios
	FECHA_FIN	smalldatetime	Fecha de Fin de Estudios
	ESCUELA_C	varchar(50)	Escuela de la que Egreso

TABLA:
EGRESADO_FALTA

Clave	Nombre de Columna	Tipo de Dato	Detalle
Pk,Fk	CEDULA	varchar(10)	Cédula del Egresado
Pk,Fk	COD	varchar(100)	Código de Requisito Faltante

TABLA:
Escalafon

Clave	Nombre de Columna	Tipo de Dato	Detalle
Pk	Codigo	int	Código de Escalafón
	Cedula	varchar(11)	Nro de Cédula Docente
	Dedicacion	varchar(500)	Dedicación actual del Docente
	Designacion	text	Designaciones dadas al Docente
	Categoria	varchar(500)	Categoria en la que se encuentra el Docente

TABLA:
ESCUELA

Clave	Nombre de Columna	Tipo de Dato	Detalle
Pk	ESCUELA_CARRERA	varchar(150)	Nombre de Escuela

TABLA:
Foja4

Clave	Nombre de Columna	Tipo de Dato	Detalle
Pk,Fk	Docente	varchar(11)	Nro de Cédula Docente
	Horas_Academicas	int	Horas de Labor Académica
	Horas_Administrativas	int	Horas de Labor Administrativa

TABLA:
Horario_Docente

Clave	Nombre de Columna	Tipo de Dato	Detalle
	CodigoId	int	Código de Horario Docente
Pk,Fk	Codigo	int	Código de Docente Materia
Pk	Dia	varchar(20)	Día de la Semana
Pk,Fk	Hora_Inicio	varchar(50)	Hora de Inicio de Labor
	Num_horas_dictadas	int	Nro de horas a Laborar
	Asistencia	varchar(5)	Confirmación de Asistencia

TABLA:
Horario_Extras

Clave	Nombre de Columna	Tipo de Dato	Detalle
Pk,Fk	Periodo	int	Nro de Período Académico
Pk	Dia	int	Día de la Semana
Pk,Fk	Docente	varchar(11)	Nro de Cédula Docente
Pk,Fk	Materia	varchar(200)	Nombre de Materia Asignada
Pk,Fk	Hora_Inicio	varchar(50)	Hora de Inicio de Labor
	Codigo	int	Código de Horario Extra

TABLA:
Horas_clase

Clave	Nombre de Columna	Tipo de Dato	Detalle
Pk	Id	int	Código de Identificación
	H_Inicio	varchar(50)	Hora de Inicio de Clases
	H_Fin	varchar(50)	Hora de Fin de Clases

TABLA:
INFOR_DOCENTE

Clave	Nombre de Columna	Tipo de Dato	Detalle
Pk,Fk	NOMBRED	varchar(100)	Nro de Cédula Docente
	HORAS_DICTADAS	float	Nro de Horas Laboradas

TABLA:
InforDocenExtra

Clave	Nombre de Columna	Tipo de Dato	Detalle
Pk,Fk	Docente	varchar(11)	Nro de Cédula Docente
	Horas	int	Nro de Horas Laboradas

TABLA:
inforestudiant

Clave	Nombre de Columna	Tipo de Dato	Detalle
Pk	cedula	varchar(11)	Cédula de Egresado o Titulado
	ciudad	varchar(100)	Ciudad de Residencia
	direccion	varchar(500)	Dirección de Domicilio
	telefono	varchar(20)	Nro Telefónico
	email	varchar(100)	e-mail del Egresado o Titulado
	empresa	varchar(100)	Empresa en la que Labora el Egresado o Titulado
	pass	varchar(50)	Password
	confirmado	varchar(50)	Confirmación de Registro
	Fecha_Registr	smalldatetime	Fecha de Registro

TABLA:
Materia_docente

Clave	Nombre de Columna	Tipo de Dato	Detalle
	Codigo	int	Código de Identificación
Pk,Fk	Cedula	varchar(11)	Nro de Cédula Docente
Pk,Fk	Materia	varchar(200)	Nombre de Materia Asignada
Pk,Fk	Nro_Periodo	int	Nro de Período Académico
	Carrera	varchar(150)	Nombre de Carrera en la que Labora

TABLA:
MATERIAS

Clave	Nombre de Columna	Tipo de Dato	Detalle
Pk	MATERIA	varchar(200)	Nombre de Materia

TABLA:
OFENVIO

Clave	Nombre de Columna	Tipo de Dato	Detalle
	Codigo	int	Código de Identificación
Pk	NRO	int	Nro de Oficio
Pk	FECHA_DIRECTIVO	smalldatetime	Fecha del Directivo
	RESOLUCION	text	Resolución dada

TABLA:
OFRECIBIDO

Clave	Nombre de Columna	Tipo de Dato	Detalle
	Codigo	int	Código de Identificación
Pk	NRO	int	Nro de Oficio
Pk	FECHA	smalldatetime	Fecha de Recepción
	ASUNTO	text	Asunto que trata

TABLA:
PERIODO

Clave	Nombre de Columna	Tipo de Dato	Detalle
Pk	NRO_P	int	Nro de Período Académico
	FECHA	smalldatetime	Fecha de Período Académico
	semestre	varchar(50)	Semestre Académico
	estado	char(3)	Estado del Período

TABLA:
Priv

Clave	Nombre de Columna	Tipo de Dato	Detalle
Pk	cod	varchar(50)	Código de Identificación
	tipo	varchar(50)	Tipo de Privilegio de Usuario

TABLA:
RECUPERACION

Clave	Nombre de Columna	Tipo de Dato	Detalle
Pk,Fk	NOMBRED	varchar(100)	Nro de Cédula Docente
Pk,Fk	MATERIA	varchar(200)	Nombre de Materia
Pk	FECHA	smalldatetime	Fecha de Recuperación
	HORAS	int	Nro de Horas de Recuperación

TABLA:
REQUISITO_E

Clave	Nombre de Columna	Tipo de Dato	Detalle
Pk	NOMBRE	varchar(100)	Requisitos para el Egresamiento

TABLA:
REQUISITO_T

Clave	Nombre de Columna	Tipo de Dato	Detalle
Pk	NOMBRE	varchar(500)	Requisitos para la Titulación

TABLA:
RequisitoLab

Clave	Nombre de Columna	Tipo de Dato	Detalle
Pk	requisl	varchar(500)	Requisitos Titulación entregado en Laboratorio
	carr	varchar(150)	Nombre de la Carrera

TABLA:
Responsables

Clave	Nombre de Columna	Tipo de Dato	Detalle
Pk	Responsable	varchar(100)	Nro de Cédula de Responsable de Carreras
	Carrera	varchar(50)	Nombre de Carrera
	Tipo	varchar(50)	Cargo de Asignación

TABLA:
Tesis

Clave	Nombre de Columna	Tipo de Dato	Detalle
Pk	Cedula	varchar(11)	Nro de Cédula del Tesista
	Director	varchar(11)	Nro de Cédula del Director de Tesis
	Nombre_Anteproy	text	Titulo de Tesis o Proyecto
	Fech_Presentacion	smalldatetime	Fecha de Presentación de Tesis/Proyecto
	Fech_Defensa	smalldatetime	Fecha de Defensa de Tesis/Proyecto
	Nota	float	Nota Obtenida por la Tesis

TABLA:
Tipo_Docente

Clave	Nombre de Columna	Tipo de Dato	Detalle
Pk	Tipo	varchar(255)	Tipo de Docente

TABLA:
TITULADO

Clave	Nombre de Columna	Tipo de Dato	Detalle
Pk	CEDULA	varchar(11)	Nro de Cédula del Titulado
	NOMBRE	varchar(255)	Apellidos y nombres del Titulado
	REQUISITOS	varchar(255)	Estado de Requisitos de Titulación
	FECHA_INICIO	smalldatetime	Fecha de Inicio de estudios
	FECHA_FIN	smalldatetime	Fecha de Finalización de Estudios
	FEC_APROB_EG	smalldatetime	Fecha de Aprobación de Egreso
	FEC_APROB_TIT	smalldatetime	Fecha de Aprobación de Titulación
	ESCUELA_C	varchar(50)	Escuela de la que se Titula

TABLA:
TITULADO_FALTA

Clave	Nombre de Columna	Tipo de Dato	Detalle
Pk,Fk	CEDULA	varchar(11)	Nro de Cédula del Titulado
Pk,Fk	COD	varchar(500)	Código de Requisito para Titulación

TABLA:
usuar

Clave	Nombre de Columna	Tipo de Dato	Detalle
Pk	Nombre	varchar(50)	Nombre de Usuario
	Nick	varchar(50)	Nick de Usuario
	tipo	varchar(50)	Tipo de Permiso

CAPITULO VI

CONCLUSIONES Y RECOMENDACIONES

6.1.Solución Tecnológica

6.2.Conclusiones

6.3.Recomendaciones

6. Conclusiones y Recomendaciones

6.1. Solución Tecnológica

En la presente Tesis se ha aplicado la tecnología ASP.NET en el desarrollo de aplicaciones web dedicadas a la automatización de procesos y actividades departamentales.

En el capítulo I se realizó el estudio de la tecnología ASP.NET, en el que se describió los claros beneficios proporcionados por ASP.NET en relación con ASP al usar librerías Framework. Se describió los aportes, beneficios y uso que nos dan los Controles de Servidor, desplazamientos de páginas, creación y administración de Controles de Usuario aplicando estilos. Además, Se realizó un estudio sobre los Controles de Validación de información, el acceso a Datos en un Servidor a través de Aplicaciones ASP.NET, Estados de aplicaciones, uso de cache en los aplicativos, configuración de ficheros, secciones y controladores, configuración del modelo de procesos, control de errores e implementación de Seguridades en los aplicativos. Al final del mismo, se realizó un estudio sobre las métricas de rendimiento, recomendaciones en el desarrollo y configuración de las aplicaciones, para aumentar su rendimiento.

En el Capitulo II, se realizó el estudio de beneficios y seguridad que nos brinda SQL Server 2005 al unirse con un aplicativo Web ASP.NET; para ello se realizó un estudio de los beneficios que nos ofrece el Servidor SQL 2005 al trabajar en una red, presentándonos sus diversas características para el trabajo en la misma y los niveles de seguridad con los que cuenta para evitar la intromisiones mediante autenticaciones, aplicación de roles y permisos.

En el Capitulo III se estudió el Desarrollo e Implementación de Aplicaciones Informáticas Web mediante el uso de diversas Metodologías de Desarrollo ya sean estas estructuradas o no estructuradas, su impacto en el Entorno de Desarrollo, características deseables en cuanto a Metodologías y una visión al Diseño de Aplicaciones Distribuidas con orientación Web.

En el Capitulo IV, se trató el tema de Seguridad Web Aplicada a Servidores, Sitios y Aplicaciones Web, mediante la implementación de la seguridad Física y Ambiental de equipos, resguardo de su información, implementación de técnicas seguras en sitios y páginas web del servidor para su correcto funcionamiento durante transacciones y acceso realizados o peticionados en la Red. Además de ello, se

realizó un sondeo de herramientas de Informáticas que nos proveen de Seguridad tanto para la red como para las aplicaciones teniendo presente principios básicos para la implementación de seguridad.

El Capítulo V se dedicó al análisis del desarrollo de los aplicativos, en este veremos la metodología usada, características de los diferentes módulos que componen el Sistema, el uso de IIS como servidor Web para el sitio del Aplicativo en ASP.NET con base de datos en un Servidor SQL 2005; luego de ello analizamos la arquitectura del Aplicativo Web, los Diagramas de Casos de Uso, Diagramas de Clases, Diagramas de Colaboración, Diagramas de Secuencia, Diagramas de Estado, Diagramas de Actividad, usados para el análisis y desarrollo de sus módulos, además de la descripción de la Base de Datos del Sistema mediante el análisis de los modelos físico, conceptual, XML y orientado a Objetos.

Por último en el capítulo VI, se pueden dar conclusiones obtenidas con respecto al Estudio de la Tecnología ASP.NET y al Desarrollo e Implementación del Aplicativo realizado con esta tecnología. Al final de este capítulo se realiza las recomendaciones pertinentes.

6.2. Conclusiones

6.2.1. Conclusiones con Respecto al Estudio

- Tras el análisis de la tecnología ASP.NET se ha llegado a la conclusión de que esta tecnología es una de las mejores para el desarrollo de aplicaciones web por la facilidad de escritura de código, al estar separada de la interfaz, su programación en diferentes lenguajes compatibles con Common Lenguage Runtime (CLR), capacidad de reutilización de controles de servidor y funciones debido a su orientación a objetos.
- Se ha concluido que para el uso de aplicaciones web con tecnología Microsoft (Windows Server, SQLServer 2005), ASP.NET es la respuesta al desarrollo de aplicaciones web ya que tiene un acople incomparable con todas estas herramientas colaborativas obteniendo un desempeño óptimo e incomparable al aprovechar todas y cada una de las características de las herramientas con las que se acopla, tanto en rendimiento como en seguridad.
- Al trabajar con aplicaciones web desarrolladas en ASP.NET, se ha concluido que la instalación de las aplicaciones se las realiza de una manera fácil y

óptima al no tener que detener el servidor web debido a la capacidad de re-compilación de aplicativos o simple envío de ficheros ya sea mediante un ftp o copia directa de los mismos al Sitio Web.

- La factibilidad de resumir la configuración de aplicaciones web mediante el archivo web.config, hacen que la administración, mantenimiento y corrección de errores sea una tarea mas sencilla y debido a la seguridad heredada a este archivo por las configuraciones establecidas en el archivo machine.config, tenemos una mayor fiabilidad de que las configuraciones personales hechas a nuestros aplicativos estarán seguras, no cambiarán y funcionarán de acuerdo a nuestras necesidades.
- Tras el análisis de seguridad de acceso al Servidor SQL Server 2005, mediante 3 capas de protección, y la versatilidad de interacción con la Seguridad implementada en Servidores Windows alojadores de paginas ASP.NET, nos dan como conclusión que el uso de este motor de datos y su Administrador operable (SQL Server Management Studio), son la mejor opción para ser el servidor de base de datos de las Aplicaciones Web.
- Debido a las conclusiones anteriormente mencionadas y la factibilidad de uso gratuito de Software Microsoft, se ha llegado a la conclusión de que, el desarrollo de aplicaciones web en ASP.NET interactuando con bases de datos en SQL Server 2005 y un entorno de trabajo Microsoft son la mejor opción por cuestiones de escalabilidad, seguridad, rendimiento y facilidad de uso.

6.2.2. Conclusiones con respecto al Aplicativo

- La fácil obtención de reportes de docentes, carga horaria docente, docente-carrera, docente-materia y horario docente hacen del aplicativo web una herramienta de incomparable importancia en la eficaz y rápida capacidad de respuesta a peticiones realizadas en el dpto. Secretario – Abogado.
- La capacidad de obtener oportunos reportes de requisitos para Egresamiento y Titulación ahorran tiempo y molestias tanto para los consultantes como para los miembros del dpto. Secretario – Abogado.
- El control diario y reporte mensual de asistencia docente, es una de las actividades del sistema de mayor ayuda para el dpto. Secretario – Abogado ya que automatiza la ardua tarea del control manual que anteriormente se lo realizaba durante semanas y hoy en día se lo realiza en contados minutos.
- El desglose de asistencia docente permite a los miembros del dpto. Secretario – Abogado, directores de escuela y coordinadores de carrera

tener un mejor control sobre el cumplimiento de asistencia docente, faltas y recuperación de horas clase.

- La administración de información de Egreso y Titulación proporcionados por el Aplicativo, es de gran ayuda para los miembros del dpto. Secretario Abogado ya que no tienen que informar de forma individual los estados de trámite de determinada persona, ahorrándoles tiempo y molestias, además los efectivos y oportunos reportes de Egresados y Titulados permiten al dpto. brindar una respuesta rápida a las constantes solicitudes de este tipo de información.
- La información de trámites de Egreso y Titulación proporcionada por el Sistema hacia sus tramitantes les permite ahorrar tiempo y dinero en el proceso de consultas de su tramitación (llamadas telefónicas, consultas presenciales, entre otras), debido a su disponibilidad en la intranet y con escalabilidad al internet.
- Los reportes de tesis elaboradas por titulados de la FICA, permite a sus consultantes orientarse en la adopción y planteamiento de un tema de tesis, director de proyecto/tesis, y mediante el resumen ejecutivo, tener una idea más clara de la estructuración de una tesis/proyecto.
- El acceso a la información de Escalafón Docente, permite a sus consultantes obtener información de Escalafón de los Docentes de la FICA.
- La administración de envío-recepción de oficios del dpto. Secretario – Abogado es de gran ayuda para tener una mejor operatividad y capacidad de respuesta ante las puntualizaciones y resoluciones contenidas en los oficios. Además de ello, nos permite tener mayor seguridad y organización en los oficios ya que se guardan en el servidor de la aplicación web.
- El aplicativo permite una gran operatividad y comunicación con la base de datos del Sistema Administrativo Secretario Abogado FICA.
- El Aplicativo nos brinda una interfaz segura y fácil de entender y usar, promoviendo con esto el aprovechamiento al 100% de las capacidades del sistema.
- El desarrollo e implementación del Sistema es de gran ayuda para el aumento de eficacia, rendimiento y control de los diversos ítems automatizados en el sistema.

6.3. Recomendaciones

- Constante alimentación de Información a la Base de Datos del Aplicativo Web para obtener un correcto funcionamiento del mismo.
- Realizar constantes mejoras y actualizaciones a los diversos módulos del Aplicativo Web.
- Es Recomendable la creación de Equipos de Desarrollo de Software a nivel de Facultad para la creación de sistemas que ayuden a la automatización de actividades y procesos realizados manualmente en la Facultad.
- Se recomienda el uso de los sistemas creados con propósitos de automatización de actividades dentro de la Facultad pues varios de ellos a pesar de ser útiles pasan desapercibidos y no son implementados y puestos en producción.
- Es recomendable el mantenimiento, actualización de software y hardware en los computadores y en algunos casos el cambio de los mismos, pertenecientes a los diferentes departamentos para evitar problemas en la utilización de Aplicativos instalados a nivel de red y de terminal.
- Se recomienda no cerrarse al uso de tecnologías Microsoft pues la universidad cuenta con un convenio de uso gratuito de software con fines académicos, las aplicaciones desarrolladas con su tecnología tienen amplia acogida a nivel empresarial, tanto pública como privada, además de ello, los docentes de la FICA y el resto de la Universidad Técnica del Norte imparten conocimientos e inculcan habilidades de y en las herramientas informáticas Microsoft.
- Se recomienda mayor cooperación entre las direcciones de escuelas/carreras y el dpto. Secretario-Abogado para obtener una mejor fluctuación de datos actuales y correctos.
- Es recomendable tener un mayor control con respecto a los cambios de horario para que puedan ser actualizados a tiempo evitando problemas en el futuro.
- Es recomendable concientizar a los docentes en la importancia de registrar las firmas en las carpetas correspondientes.

GLOSARIO

ASP

Las Active Server Pages son un ambiente de aplicación abierto y gratuito en el que se puede combinar código HTML, scripts y componentes ActiveX del servidor para crear soluciones dinámicas y poderosas para el web.

API

API (Application Program Interface). Interface entre Programa y Aplicación. Conjunto de normas que determinan como debe usarse una determinada función de un programa en una aplicación.

CLR

El CLR es el encargado de convertir este lenguaje intermedio en leguaje máquina del procesador, esto normalmente se hace en tiempo real por un compilador JIT (Just-In-Time) que lleva incorporado

COM

COM (Component Object Model).Los lenguajes de programación clásicos fueron diseñados para desarrollar aplicaciones secuenciales compuestas de módulos, todos ellos codificados con un solo lenguaje. Sin embargo, hay situaciones en las que no es práctico restringirse al uso de un único lenguaje. La tecnología COM aborda la solución a este problema proporcionando un sencillo, pero a la vez potente modelo para construir sistemas software a partir de la interacción de objetos (componentes).

COOKIE

Archivo creado y almacenado en el disco duro por determinadas web para facilitar la navegación por ellas. Suelen guardar datos sobre la cuenta de usuario y sus preferencias, permitiendo recordar al visitante en sus posteriores accesos a la página.

FIREWALL

Sistema que ejecuta una política de control de acceso entre redes y que por lo tanto no protege contra acceso internos indebidos.

HARDENING

Implementación de las mejores prácticas provistas por un proveedor o bien por instituciones competentes sobre la configuración de un sistema.

HOSTING

Servicio por el cual un proveedor proporciona el/los servidores para alojar y publicar un sitio Web. En este caso todo el equipamiento pertenece al proveedor.

HOUSING

Servicio por el cual un proveedor proporciona un lugar físico para alojar el/los servidores destinados al sitio Web, así como los medios de conectividad necesarios para su implementación y mantenimiento en línea. En este caso el/los servidores pertenecen al cliente.

HTTP

HTTP(Hypertext Transfer Protocol)es el Protocolo de Tranferencia de Hypertexto es un protocolo de nivel de aplicación del Modelo OSI, aplicado para la comunicación cliente-servidor en sistemas de la World Wide Web.

IIS

Internet Information Service (IIS) es el Servidor Web del entorno Windows.

INCIDENTE

Evento adverso en un sistema de computadoras, o red de computadoras, que compromete la confidencialidad, integridad o disponibilidad, la legalidad y confiabilidad de la información. Puede ser causado mediante la explotación de alguna vulnerabilidad o un intento o amenaza de romper los mecanismos de seguridad existentes.

JIT

El Just–in–Time JIT es el compilador incorporado del CLR, compila código en tiempo real de lenguaje intermedio a lenguaje maquina.

MD5

Algoritmo fuerte de cifrado que utiliza una función de “hash” o digesto.

NGWS

Nueva Generación de Windows Server (NGWS).

PROXY

Un servidor proxy actúa como una barrera o servidor de seguridad entre la intranet e Internet, lo que evita que otras personas obtengan acceso en Internet a información confidencial en la red interna o en el equipo. El administrador del sistema de telefonía puede proporcionarle el nombre o la dirección IP correctos para especificarlos aquí.

SCP

El SCP (Simple Control Protocol) es un protocolo utilizado para la transferencia de archivos en forma segura.

SFTP

SFTP (Secure Protocol) es un protocolo utilizado para la transferencia de archivos en forma segura.

SHA-1

Secure Hash Algorithm es un algoritmo fuerte de cifrado que utiliza una función de “hash” o digesto, denominado de una sola vía ya que una vez cifrada la información, no permite efectuar la “vuelta atrás” del algoritmo (no posee función inversa).

SSH

Secure Shell es el protocolo utilizado para acceder a máquinas a través de una red, utilizando técnicas de cifrado de la información transferida.

SSL

Secure Sockets Layer (SSL) puede utilizarse en IIS para intercambiar información cifrada mediante el protocolo HTTPS. SSL proporciona cifrado en ambas direcciones: la información se transmite al usuario mediante cifrado y la información que envía el usuario a la aplicación está asimismo cifrada.

TCP/IP

Protocolo de control de transmisiones/Protocolo Internet. Es el protocolo estándar de comunicaciones en red utilizado para conectar sistemas informáticos a través de Internet

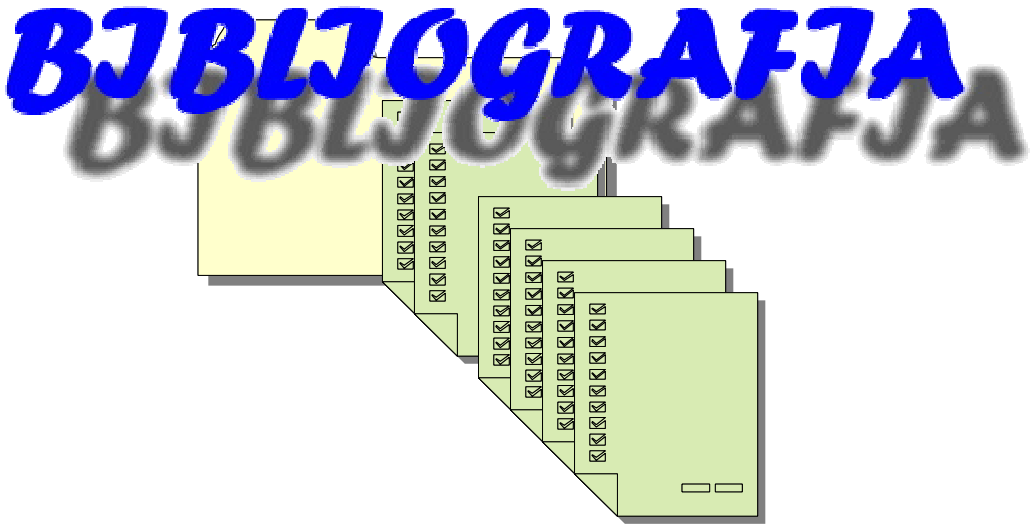
TRY/CATCH

La instrucción try-catch consta de un bloque try seguido de una o más cláusulas catch, las cuales especifican controladores para diferentes excepciones.

URL

URL (localizador universal de recursos), método de identificación de documentos o lugares en Internet, que se utiliza principalmente en World Wide Web (WWW). Un URL es una cadena de caracteres que identifica el tipo de documento, la computadora, el directorio y los subdirectorios en donde se encuentra el documento y su nombre.

BIBLIOGRAFIA



SITIOS DE INTERNET

http://es.wikipedia.org/wiki/ASP.NET
[www.001]

http://www.webestilo.com/aspnet/aspnet00.phtml

[www.002]

http://www.webestilo.com/aspnet/aspnet01.phtml

[www.003]

http://es.gotdotnet.com/quickstart/aspplus/doc/stateoverview.aspx

[www.004]

http://es.gotdotnet.com/quickstart/aspplus/doc/webdataaccess.aspx.htm

[www.005]

ms-help://MS.VSCC.v80/MS.MSDN.v80/MS.VisualStudio.v80.es/dv_aspnetcon/html/a5ff235e-397f-4bbe-9bfe-2720b6e7ab9d.htm

[www.006]

http://www.aprendamas.net

[www.007]

http://es.gotdotnet.com/quickstart/aspplus/doc/webdataaccess.aspx.htm#delete

[www.008]

http://es.gotdotnet.com/quickstart/aspplus/doc/webdataaccess.aspx.htm#xmldata

[www.009]

http://es.gotdotnet.com/quickstart/aspplus/doc/applications.aspx.htm#lifetime

[www.010]

ms-help://MS.VSCC.v80/MS.MSDN.v80/MS.VisualStudio.v80.es/dv_aspnetcon/html/1e0caf13-e8f0-46d5-9ca0-8b53e4b1c08e.htm

[www.011]

ms-help://MS.VSCC.v80/MS.MSDN.v80/MS.VisualStudio.v80.es/dv_aspnetcon/html/0218d965-5d30-445b-b6a6-8870e70e63ce.htm

[www.012]

ms-help://MS.VSCC.v80/MS.MSDN.v80/MS.VisualStudio.v80.es/dv_vwdcon/html/3847984b-858a-4d43-b565-f10fa86b6bcc.htm

[www.013]

http://es.gotdotnet.com/quickstart/aspplus/doc/cachingoverview.aspx.htm

[www.014]

http://es.gotdotnet.com/quickstart/aspplus/doc/datacaching.aspx.htm

[www.015]

http://es.gotdotnet.com/quickstart/aspplus/doc/configformat.aspx.htm

[www.016]

http://es.gotdotnet.com/quickstart/aspplus/doc/procmodel.aspx.htm

[www.017]

ms-help://MS.VSCC.v80/MS.MSDN.v80/MS.VisualStudio.v80.es/dv_aspnetcon/html/c34d6f4f-f64d-4697-bd32-02dd2ddf726f.htm

[www.018]

http://es.gotdotnet.com/quickstart/aspplus/doc/windowsauth.aspx.htm

[www.019]

http://es.gotdotnet.com/quickstart/aspplus/doc/perfoverview.aspx.htm

[www.020]

http://es.gotdotnet.com/quickstart/aspplus/doc/perftuning.aspx.htm

[www.021]

http://www.buayacorp.com/archivos/10-errores-mas-comunes-que-pueden-causar-problemas-de-seguridad-en-aspnet

[www.022]

http://www.microsoft.com/latam/sql/evaluation/features/default.asp

[www.023]

http://mredison.wordpress.com/2007/09/28/seguridad-en-sql-server-2005/

[www.024]

http://www.WINDOWSTIMAG.COM

[www.025]

http://www.microsoft.com/argentina/technet

[www.026]

http://www.portalsql.com/

[www.027]

http://www.rhernando.net/modules/tutorials/doc/ing/met_soft.html

[www.028]

http://www.chuidiang.com/ood/metodologia/metodologia.php

[www.029]

http://es.wikipedia.org/wiki/Desarrollo_Ã¡gil_de_software

[www.030]

http://www.monografias.com/trabajos14/aplicacion-distrib/aplicacion-distrib.shtml

[www.031]

http://www.arcert.gov.ar/webs/tips/sitiosweb.htm

[www.032]

http://www.comunidadhosting.com/asuntos-tecnicos-seguridad-y-configuracion/2673-consideraciones-de-seguridad-en-servidores-web.html

[www.033]

http://www.wikilearning.com/monografia/historia_y_funcionamiento_de_internet-seguridad_en_paginas_web/3443-18

[www.034]

http://www.desarrolloweb.com/articulos/996.php

[www.035]

OTRAS FUENTES BIBLIOGRÁFICAS DE INTERNET

http://www.cs.ualberta.ca/~pfiguero/soo/metod/uml-met.html
http://www.arcert.gov.ar/webs/textos/OWASP_Top_Ten_2004_Spanish.pdf
http://www.owasp.org/guide/
http://www.owasp.org/
http://developers.slashdot.org/article.pl?sid=02/09/25/1317210
http://www.linuxjournal.com/article.php?sid=6061
http://www.dwheeler.com/secure-programs/
http://www.isecom.org/projects/spsmm-es.htm
http://www.hispasec.com/unaaldia.asp?id=1071
http://msdn2.microsoft.com/en-us/library/ms972975.aspx
http://msdn2.microsoft.com/en-us/library/ms972975.aspx#usercontrols_topic9
http://msdn2.microsoft.com/en-us/library/ex526337.aspx

REFERENCIAS BIBLIOGRAFICAS

Craig Larman,UML y Patrones. PRENTICE HALL, México,1999.

James Rumbaugh, El Lenguaje Unificado de Modelado. PEARSON EDUCATION S.A., Madrid, 2000.

Douglas J. Reilly, Diseño de Aplicaciones con Microsoft. Mc Graw Hill, USA, 2002.

Paul Litwin, Creación de Sitios Web con ASP.NET.PEARSON EDUCATION S.A., Madrid, 2002.

Dino Esposito, Programación Avanzada de Aplicaciones con Microsoft ASP.NET 2.0. ANAYA MULTIMEDIA, 2007.

ANEXOS

En el CD de Tesis se adjunta una carpeta llamada ANEXOS, en la cual se encuentran los siguientes anexos

1. Estudio de Viabilidad del Sistema Administrativo Departamental Secretario Abogado

ANEXOS/Estudio de Viabilidad/

2. Entrevistas y Captura de Requisitos para el Sistema Administrativo Departamental Secretario Abogado.

ANEXOS/Entrevistas/

3. Diseño del Sistema Administrativo Departamental Secretario Abogado.

ANEXOS/Diseño del Sistema/

4. Código Fuente del Sistema Administrativo Departamental Secretario Abogado.

ANEXOS/Codigo Fuente/

5. Manual de Usuario, Manual Técnico y Manual de Operaciones del Sistema Administrativo Departamental Secretario Abogado.

ANEXOS/Manuales/

6. Instaladores del Sistema Administrativo Departamental Secretario Abogado, software necesario para el funcionamiento del sistema.

ANEXOS/Software/