
# UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

**FACULTAD DE ESTUDIOS SUPERIORES CUAUTITLÁN**

**DESARROLLO DE UN SISTEMA DE ADMINISTRACIÓN PARA LOS LABORATORIOS DE CÓMPUTO DE LA FACULTAD DE ESTUDIOS SUPERIORES CUAUTITLÁN CON TECNOLOGÍAS MICROSOFT .NET**

# T E S I S

**QUE PARA OBTENER EL TÍTULO DE:**

## LICENCIADO EN INFORMÁTICA

**P R E S E N T A N:**

**FRANCISCO VERA CERVANTES**

**JORGE ALBERTO CORTES GAMIÑO**

**ASESOR: ING. MARCO ANTONIO CRUZ MENDOZA**

**CUAUTITLÁN IZCALLI, EDO. DE MÉX. 2006**

# DEDICATORIAS

**De: Jorge Alberto Cortes Gamiño**

**A Dios:**

Te doy gracias por todo lo que has hecho en mi vida, en estos momentos no se donde estaría sino fuera por tu gracia y tu perdón. Y sobre todo por darme la oportunidad de conocerte y la fuerza necesaria para poder llegar al termino de este trabajo.

**A mi mamá:**

Te dedico este trabajo con todo mi corazón. Se que hemos pasado momentos difíciles, pero siempre te las arreglaste para cuidar de mi y mis hermanos. Tu ejemplo, consejo y fortaleza me han guiado hasta este punto de mi vida. Y aun estando lejos los tengo presentes todo el tiempo.

No hay palabras para decirte lo agradecido que estoy de tenerte como mi mamá.

**A Toño:**

Gracias por darme mi principal herramienta que me ha sido tan útil a lo largo de toda mi carrera.

**A Luis, Ale y Toñito:**

No saben la alegría que me da tenerlos como hermanos. Son toda una bendición a mi vida, los quiero mucho.

**A Francisco:**

Que te puedo decir hermano. Has sido mi mejor amigo a través de toda la carrera y un gran compañero en la realización de este trabajo.

**A mi papá:**

Aunque hace mucho tiempo que no nos vemos, se que estarías orgulloso de esta meta.

**De: Francisco Vera Cervantes**

**A Dios:**

A El toda la gloria, honra y honor. El siempre cumple sus promesas, una de ellas es el cumplimiento de una meta en mi vida. Por esa razón sólo le puedo dar mil gracias al creador por el amor que me ha permitido conocer el cual estoy seguro es incondicional hacia mi y ese amor es nada menos que el de su hijo Jesucristo.

**A mi esposa:**

Getsy eres realmente un regalo de Dios en mi vida no hay palabras para describir tan maravillosa experiencia el vivir a lado tuyo. Gracias por creer en mí, por soportar mi mal genio, por cuidarme, por el tiempo que cediste quedándote sola, y por ayudarme a llegar al final de esta empresa. Te amo princesa.

**A mis Papás:**

Francisco Vera Nava y Ma. Del Carmen Cervantes

Gracias Papá eres lo máximo, tu ejemplo ha sido de invaluable valor en mi vida, mama tu amor y comprensión me han alentado.

**A mis hermanos:**

Adrián, Santa, Norma, Rubén, Guillermo, Sofía y Mario Alan todos los traigo en mi pensamiento y en mi corazón, le doy gracias a Dios por dármelos como mis hermanos, los amo.

**A la hermana Arcelia:**

Gracias por dedicar su tiempo, esfuerzo, su apoyo moral e inclusive económico, es toda una guerrera de Dios, y yo soy producto de una batalla. ¡Victoria para Dios!

**A Jorge:**

En todo tiempo ama al amigo y es como un hermano en tiempo de angustia. (Proverbios 17:17). Has sido el mejor amigo que DTB.

# AGRADECIMIENTOS

**De: Jorge Alberto Cortes Gamiño**

**A Brenda:**

Por ser una gran amiga a través de toda la carrera, eres un gran apoyo.

**A Jacqueline:**

El apoyo incondicional que nos brindaste en el trámite de la tesis fue increíble, muchas gracias.

**A Marco y Oscar Trejo:**

Por abrirme las puertas de su casa cuando las circunstancias no me eran tan favorables. Se los agradezco demasiado.

**A mi grupo de Informática:**

Nunca pensé llegar a estar en un grupo tan unido. Siento un gran afecto por cada uno de ustedes y por hacer la estadía en la carrera de lo más amena. Siempre los mantendré en mi mente y corazón.

**A mi asesor:**

Por todo el tiempo que invirtió con nosotros. Muchas gracias por todo el apoyo y conocimiento que nos brindo para poder llevar acabo este trabajo.

**A mis sinodales:**

Gracias por su tiempo y sus comentarios para hacer de este un mejor trabajo.

**A mis jefes:**

Erik, Isabel y Alfredo gracias darme la oportunidad de desenvolverme profesionalmente. Además de todos los permisos que me concedieron para poder terminar mis estudios, les estaré eternamente agradecido.

**A mis pastores:**

Por ser un gran ejemplo y porque han sido como mi segunda familia desde que los conozco. Les agradezco todo su cariño.

**A la UNAM:**

Por permitirme formar parte de esta gran casa de estudios y darme la oportunidad de terminar mis estudios de licenciatura.

**De Francisco Vera Cervantes:**

**A mi grupo de Informática:**

Cada uno de ustedes logro que mi estancia en la carrera fuera tan placentera. Las lacras siempre estarán presentes.

**A mi asesor:**

Ingeniero Cruz gracias por compartir su valioso tiempo, su esfuerzo, dedicación y conocimientos para la realización de este proyecto.

**A mis pastores:**

Gracias le doy por sus vidas, sus oraciones han hecho mella sobre mi, son de gran aprecio en mi vida.

**A la UNAM:**

Por permitirme ser uno de sus integrantes, por los conocimientos y experiencias de todos y cada uno de los profesores que en sus aulas comparten con los alumnos, sin exigir retribución alguna más que, la de aprovechar el conocimiento, y demostrar que de esta forma en la Universidad Nacional Autónoma de México existen personas capaces.

**INDICE**

# OBJETIVOS

## GENERALES

Desarrollar un sistema para administrar los laboratorios de cómputo de la Facultad de Estudios Superiores Cuautitlan con el uso de la tecnología Microsoft .NET.

## ESPECIFICOS

Describir el modelo de aplicación en tres capas para el desarrollo de aplicaciones.

Utilizar tecnologías Microsoft .NET para la construcción del sistema: ASP .NET para la construcción de la interfaz con el usuario, Visual Basic .NET como lenguaje de programación, SQL Server Express versión 2005 para el manejo de bases datos y ADO .NET para realizar la conexión entre la base de datos y el sistema.

# INTRODUCCIÓN

La Plataforma .NET es una tecnología creada por Microsoft la cual esta integrada por un conjunto de productos desde sistemas operativos, como Windows XP, Servidores de aplicaciones como SQL Server 2000, productos de oficina como Office XP, herramientas de desarrollo como Visual Studio .NET hasta servicios Web provistos por Microsoft como .NET Passport y .NET Alerts. Esta plataforma utiliza los servicios Web como un medio para poder interoperar a distintas tecnologías. Permitiendo conectar distintos sistemas operativos, dispositivos físicos, información y usuarios.

Partiendo de lo anterior, el trabajo descrito en este documento se refiere al Desarrollo de un sistema de administración para los laboratorios de cómputo de la Facultad de Estudios Superiores Cuautitlan mediante el uso de dicha tecnología. Este sistema permite ver la disponibilidad de las computadoras; Concentar información en una base de datos; La posibilidad de agregar funciones cuando así se requiera; Disposición de los saldos de los recibos en los laboratorios; entre otras. Todo esto de una forma de una sencilla, amigable y con todas las características de una aplicación Web que puede ser accesada desde cualquier explorador de Internet.

Las tecnologías para el desarrollo del sistema son ASP .NET y Visual Basic .NET (ambas versión 2.0) para la construcción de la interfaz del usuario así como la programación; SQL Server Express versión 2005 para la el manejo de la base de datos (también desarrollada en este trabajo) y ADO .NET (versión 2.0) para realizar la conexión entre la base de datos y el sistema.

La tecnología Microsoft .NET ha sido usada para este trabajo debido a que todos los programas necesarios para el desarrollo del sistema son gratuitos. Lo cual da la libertad de ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el software conforme se vaya requiriendo. Propone un modelo de desarrollo de aplicaciones Web compiladas en lugar de interpretadas por los navegadores de Internet. Además de que proporciona una librería de clases que pueden ser utilizadas y con las cuales se pueden crear clases nuevas compatibles

con diferentes lenguajes de programación (Visual Basic .NET, C# .NET, J# .NET, C++ .NET).

Se uso el modelo de base de datos relacional y la arquitectura cliente/servidor. El primero debido a que es el que maneja Microsoft SQL Server Express versión 2005 y el segundo por que el número de computadoras que va a acceder al sistema es reducido y por lo cual no se necesita de un modelo más complejo.

# CAPITULO 1

# PLATAFORMA .NET

En 1998 un equipo de trabajo de Microsoft comenzó a trabajar en un proyecto que denominaron Next Generation Windows Servicies (NGWS). Este equipo se fusionó con el grupo encargado de desarrollar la versión 7 del Visual Studio con el fin de desarrollar un entorno común para todos los lenguajes incluido en él de forma que permitiese a terceras empresas crear lenguajes adaptados al entorno. Finalmente, en el año 2000 Microsoft dio a conocer este trabajo que denominaron Microsoft .NET.

La plataforma .NET fue creada para proveer los cimientos para una generación diferente de software utilizando los Servicios Web como un medio para poder interoperar a distintas tecnologías[1]. Permitiendo conectar distintos sistemas operativos, dispositivos físicos, información y usuarios. Dando a los desarrolladores las herramientas y tecnologías para hacer rápidamente soluciones de negocios que involucren distintas aplicaciones, dispositivos físicos e instituciones.

La idea central detrás de la plataforma .Net es la de construir, instalar, consumir, integrar o agregar programas para que puedan ser accedidos mediante Internet. Esto se hace posible debido a que se tiene la comunicación global que es Internet cada vez más rápida y a un costo menor. El usuario de Internet puede con un explorador de Internet no solamente acceder a contenido como texto, imágenes o sonido, también puede hacer uso de Servicios Web. Estos son bloques de construcción o componentes sobre los cuales se basa el modelo de computación distribuida en Internet[1]. La plataforma .NET permite usar Internet y su capacidad de distribución para que los usuarios accedan desde cualquier dispositivo, en cualquier sistema operativo y lugar a la funcionalidad que los Servicios Web poseen.

Los desarrolladores por su parte tienen las herramientas para crear Servicios Web y hacer uso de ellos en programas. Es decir, se trata de aprovechar la capacidad de distribución a gran escala de Internet para acceder a servicios de software. También se trata de aprovechar el incremento de la capacidad de procesamiento de los nuevos dispositivos móviles (Smart Devices) para que el usuario haga uso de la funcionalidad que proveen los Servicios Web.

[1] Fuente adquirida de la página de Desarrollador 5 Estrellas de Microsoft Capitulo 1 Sección 1.2, **http://www.microsoft.com/spanish/msdn/comunidad/dce/1/default.asp#topic1**

La plataforma .NET para realizar todo esto utiliza tecnologías existentes, productos modificados y elementos nuevos (ver Figura 1.1) [2]:

- Productos Existentes (COM): Microsoft tiene una tecnología para la creación, invocación y uso de componentes llamada COM (Modelo de Objetos Componentes). Al igual que el protocolo SOAP (Protocolo Simple de Acceso a Objetos) utilizado para la invocación de los Servicios Web, COM establece las reglas acerca de cómo los objetos deben de ser invocados y como deben interactuar. También los componentes COM pueden tener funcionalidad similar a la de los Servicios Web. Sin embargo tienen dos puntos en contra: el primero es que no es una tecnología estándar para Microsoft y el segundo es que no pueden ser utilizados fuera de la barrera de seguridad que las empresas tienen para su comunicación hacia y desde Internet (firewall). Por lo tanto no sirven al modelo de computación distribuida en Internet[2].

  La plataforma .NET puede por medio de clases especiales hacer uso de COM y los objetos COM también pueden hacer uso de los Servicios Web. Lo que permite aprovechar la plataforma instalada para el desarrollo de nuevos proyectos.

- Productos Modificados: La familia de sistemas operativos Windows 2000 fue modificado para soportar a la plataforma .NET. También todos los servidores de aplicaciones fueron modificados para permitir la interoperatividad basada en XML.

- Elementos nuevos: **Biztalk Server** es un producto que sirve para entender y manipular datos en formato XML y para poder transformar datos en XML a otros formatos y viceversa. Permite coordinar el flujo de información entre aplicaciones dentro de la institución y también el flujo de datos con otras instituciones. Algo muy importante actualmente ya que las distintas aplicaciones deben de comunicarse entre si y muchas veces los formatos de los datos son incompatibles.

[2] Fuente adquirida de la página de Desarrollador 5 Estrellas de Microsoft Capitulo 1 Sección 1.6, **http://www.microsoft.com/spanish/msdn/comunidad/dce/1/default.asp#topic1**

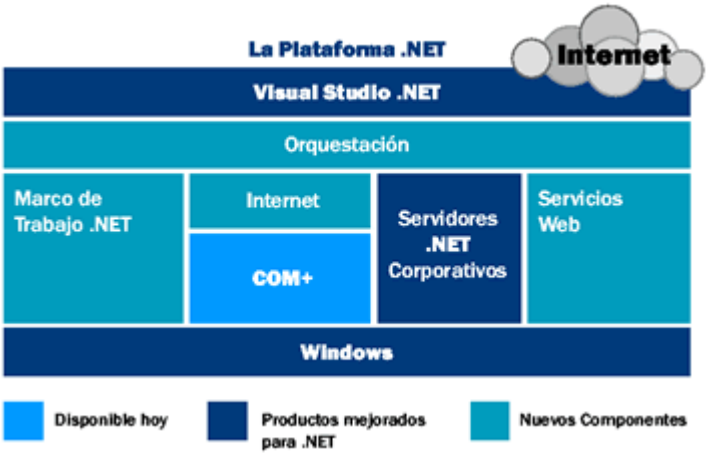



Figura 1.1. Tecnologías usadas por la plataforma .NET

## 1.1 Propósitos de la Plataforma .NET

El propósito de la plataforma .NET es simplificar el desarrollo de aplicaciones Web proveyendo las herramientas y tecnologías para transformar a Internet en una plataforma de computación distribuida a gran escala[3]. Así como de eliminar los diferentes problemas a los cuales se enfrentaba un desarrollador al desarrollarlas. Por lo que los objetivos de la plataforma .NET son:

- Eliminar los inconvenientes derivados del modelo COM, como el infierno de las DLL[4]. Para ello se emplea una aproximación basada en ensamblados y un almacén o caché global de los mismos.

- Proporcionar un sistema automático de administración de memoria denominado recolector de basura (garbage collector). Detecta cuándo el programa deja de utilizar la memoria y la libera automáticamente. De esta forma el programador no tiene por que liberar la memoria de forma explícita aunque también sea posible hacerlo manualmente[5].

[3] Erich R. Bühler, **Visual Basic .NET Guía de Migración y Actualización**, Edit. Mc. Graw Hill, Madrid 2002, Pág. 11.
[4] Para mayor información visitar la página **http://es.wikipedia.org/wiki/DLL_Hell**
[5] Para mayor información visitar la página **http://www.desarrolloweb.com/articulos/1329.php?manual=48**

- Ofrecer una mayor sencillez para utilizar componentes desarrollados en otros lenguajes, e incluso residentes en otros sistemas operativos.

- Hacer más fácil la instalación, a los efectos de que sea posible dejar un componente funcional simplemente copiándolo al directorio de la aplicación.

- Proveer el mismo modelo para utilizar un componente localmente o a través de una red.

- Hacer más fácil el desarrollo de aplicaciones distribuidas y especialmente de soluciones para Internet mediante un único entorno de desarrollo.

- Proveer ejecución segura a través de un modelo de seguridad para aplicaciones y componentes. Esto es de principal interés cuando se desea que un control sea ejecutado, y que los recursos al cual el mismo tiene acceso puedan ser limitados.

- Hacer más fácil el desarrollo de aplicaciones mediante un conjunto de funcionalidades previamente hechas, las cuales cubran varios aspectos comúnmente utilizados.

- Proveer características de orientación a objetos para todos los lenguajes de la infraestructura de forma nativa.

## 1.2 Componentes de la Plataforma .NET

La plataforma .NET no es sólo un producto. Es un conjunto de productos. Desde sistemas operativos como Windows XP, servidores de aplicaciones como SQL Server 2000, productos de oficina como Office XP, herramientas de desarrollo como Visual Studio .NET hasta Servicios Web provistos por Microsoft como .NET Passport. . Sin embargo, estos productos no están basados en .NET Framework, pueden funcionar dentro del entorno de

ejecución de .NET Framework, pero el único producto actualmente desarrollado bajo el nuevo entorno es Visual Studio .NET.

Aparte de los productos están los componentes de la plataforma .NET, los cuales son:

### 1.2.1 Clientes Inteligentes (Smart Clients)

Son dispositivos muy variados y sus características son:

- Permiten acceder a la información en el formato apropiado, en cualquier momento y lugar.
- Hacen uso de los Servicios Web.
- Optimizan de distintas maneras la forma en que la información es presentada y organizada. Por ejemplo: Pueden convertir texto en sonido en un celular o reconocer la escritura en un TabletPC.
- Proveen de una interfase sencilla y natural para que el usuario acceda a la información. Pueden utilizar la identidad del usuario, su perfil y datos para adaptar la información que es presentada.
- Pueden reconocer la presencia de otros dispositivos e intercambiar información
- Pueden adaptarse a las características de la red donde se conecten. Por ejemplo la velocidad de transmisión.
- Tienen capacidad de procesamiento propio, y distribuyen el procesamiento en la red haciendo uso de los Servicios Web.

Entre los clientes inteligentes se encuentran:

- PC de bolsillo (Pocket PC).
- Teléfono Inteligente (Smart Phone).
- Handhelds.
- TabletPC.

- Consolas de juego de Microsoft (Xbox).
- Computadoras Personales (PC).
- Computadoras Portátiles (NoteBooks).

### 1.2.2 Servidores

Proveen de la infraestructura para implementar el modelo de computación distribuida en Internet. Son sistemas operativos y de aplicación.

- Microsoft Application Center 2000: Para instalar y administrar aplicaciones Web altamente disponibles y escalables.
- Microsoft BizTalk Server 2000: Para construir procesos de negocios basados en XML a través de distintas aplicaciones y organizaciones.
- Microsoft Comerse Server 2000: Para construir rápidamente soluciones de comercio electrónico escalables.
- Microsoft Content Management Server 2001: Para administrar contenido para sitios Web de comercio electrónico dinámicos.
- Microsoft Exchange Server 2000: Para permitir enviar mensajes y trabajar en forma colaborativa en cualquier momento y lugar.
- Microsoft Host Integration Server 2000: Para acceder a datos y aplicaciones en mainframes.
- Microsoft SQL Server 2000: Para almacenar, recuperar y analizar datos en formato XML.
- Microsoft SharedPoint Portal Server 2001: Para encontrar, compartir y publicar información de negocios.
- Microsoft Internet Security and Acceleration Server 2000: Para conectividad a Internet rápida y segura.
- Microsoft Mobile Information 2001 Server: Para soportar aplicaciones en dispositivos móviles como por ejemplo celulares.

### 1.2.3 Servicios Web Basados en XML

Son los bloques de construcción de la tercera generación de Internet. Algunas de sus características son:

- Permiten a las aplicaciones compartir datos: Son componentes. Es decir, unidades de código discretas, cada una haciendo una tarea en particular.
- Están basados en el lenguaje universal de intercambio de datos de Internet (XML): Pueden ser llamados desde distintos sistemas operativos, plataformas de hardware y lenguajes de programación.

Microsoft ofrece dos servicios Web que se complementan: .NET Passport y .NET Alerts, además del MapPoint .NET. Ambos proveen los ladrillos de construcción para hacer aplicaciones y sitios donde la interfase es consistente y sencilla.

- .NET Passport: Es un conjunto de Servicios Web. Desde 1999 permite entre otras cosas que un usuario se autentique una sola vez y luego acceda a los sitios que hacen uso del servicio sin necesidad de volver a identificarse. Estos sitios pueden tener en cuenta los datos del usuario como su perfil de usuario para adaptar la interfase o proveer de servicios específicos.
- .NET Alerts: Le permite a los sitios que lo utilizan llegar a sus clientes por medio de alertas implementadas como mensajes instantáneos que sus clientes pueden optar por recibir. Los usuarios pueden elegir o recibir las alertas en su computadora cuando esta en línea, en su casilla de e-mail o en su celular.
- MapPoint .NET: Es un Servicio Web programable. Con el se pueden integrar mapas, direcciones de transito, calculo de distancias, entre otras.

### 1.2.4 Experiencias del Usuario

Soluciones que implementan los Servicios Web para hacer la experiencia de uso de una aplicación más sencilla y centrada en el usuario.

La idea de estas Aplicaciones es que el usuario este no debe adaptarse a la aplicación sino por el contrario la aplicación debe reconocer al usuario, traer sus datos, su perfil y preferencias. Más concretamente, la información no esta relacionada con ningún dispositivo. La información es la información del usuario y el software debe llevarla hacia él sin importar en que dispositivo se encuentre trabajando.

### 1.2.5 Herramientas de Desarrollo

Visual Studio .NET y el .NET Framework ambos permiten al desarrollador hacer Servicios Web basados en XML además de otro tipo de aplicaciones. El .NET Framework viene incorporado directamente en la nueva línea de sistemas operativos Windows .NET. Para los dispositivos móviles se llama .NET Compact Framework.

Los componentes de la plataforma .NET pueden interactuar de distintas maneras. Esta comunicación es permitida por los Servicios Web que integran los distintos tipos de dispositivos y componentes.

Hay cuatro tipos de interacciones posibles:

- Cliente con Cliente: Clientes Inteligentes (Smart Clients) o dispositivos pueden proveer de Servicios Web y utilizarlos para permitir que la información este disponible en todo momento y lugar.

- Cliente con Servidor: Los Servicios Web permiten que un servidor comparta datos con una computadora personal (PC) o un dispositivo móvil vía Internet.

- Servidor con Servidor: Una aplicación es un servidor puede automáticamente acceder a otra aplicación utilizando un Servicio Web como interfase.

- Servicio con Servicio: Un servicio Web puede invocar a otro, aumentando de esta manera la funcionalidad disponible.

Para este trabajo sólo se utilizó la interacción cliente con servidor.

## 1.3 El .NET Framework

### 1.3.1 Concepto

El .NET Framework es una plataforma informática que contiene un conjunto de servicios de programación diseñados para simplificar el desarrollo y la distribución de aplicaciones en un entorno altamente distribuido como es Internet. Es el elemento principal sobre el que se asienta la plataforma Microsoft .NET.

Además de que es:

- Una plataforma para escribir aplicaciones enfocadas hacia Internet que adopta estándares abiertos como XML, HTTP, SOAP, etc.
- Una plataforma que proporciona un conjunto extenso y valioso de tecnologías de desarrollo de aplicaciones, como son Windows Forms, usada para construir las clásicas aplicaciones GUI en Windows y ASP.NET, que se usa para desarrollar aplicaciones Web.
- Una plataforma con una extensa librería de clases que proporciona soporte para acceso a datos (relacionales y XML), servicios de directorio, colas de mensajes, etc.
- Una plataforma con una librería de clases base (Base Class Library) que contiene cientos de clases para realizar tareas comunes como manipulación de ficheros, acceso al registro, seguridad, programación mutihilo y el trabajo con expresiones regulares.
- Una plataforma neutral al lenguaje (plataforma multilenguaje). Los desarrolladores pueden usar el lenguaje con el que se sientan más a gusto o con el que sean más productivos sin ningún tipo de desventaja con respecto a usar otro lenguaje de los muchos disponibles para la plataforma .NET (actualmente existen compiladores para plataforma .NET de C#, VB.NET, JS.NET, Pascal, Cobol, Fortran, etc.).
- Una plataforma que no olvida sus orígenes y tiene un gran soporte para la interoperabilidad con los componentes existentes que se han desarrollado usando COM o DLL’s estándar.

- Una plataforma con un entorno independiente de ejecución y manejo de código denominado CLR (Common Language Runtime), el cual asegura que la ejecución del código es segura y proporciona una capa de abstracción por encima del sistema operativo lo que significa que elementos del .NET Framework pueden ser ejecutados en otros sistemas operativos (proyecto Mono)[6] y dispositivos (PDA's, teléfonos móviles, etc.). Esto es debido a que los elementos de .NET se sitúan entre los programas y el sistema operativo constituyendo una interfaz genérica entre ambos.

### 1.3.2 Propósitos

El diseño de .NET Framework está enfocado a cumplir los propósitos siguientes:

- Proporcionar un entorno coherente de programación orientada a objetos, en el que el código de los objetos se pueda almacenar y ejecutar de forma local, ejecutar de forma local pero distribuida en Internet ó ejecutar de forma remota.
- Proporcionar un entorno de ejecución de código que reduzca lo máximo posible la implementación de software y los conflictos de versiones.
- Ofrecer un entorno de ejecución de código que garantice la ejecución segura del mismo, incluso del creado por terceras personas desconocidas o que no son de plena confianza.
- Proporcionar un entorno de ejecución de código que elimine los problemas de rendimiento de los entornos en los que se utilizan secuencias de comandos o intérpretes de comandos.
- Ofrecer al desarrollador una experiencia coherente entre tipos de aplicaciones muy diferentes, como las basadas en Windows o en el Web.
- Basar toda la comunicación en estándares del sector para asegurar que el código de .NET Framework se puede integrar con otros tipos de código.

[6] Para mayor información visitar la página
**http://es.tldp.org/Presentaciones/200103hispalinux/garcia/html/primera-sec.html**

### 1.3.3 Estructura del .NET Framework

.NET Framework permite el desarrollo de aplicaciones a través del uso de un conjunto de herramientas y servicios que proporciona, y que se agrupan en dos bloques principales:

- Common Language Runtime ó Entorno Común de Ejecución de Lenguajes (CLR).
- Librería de Clases Base.

#### 1.3.3.1 Entorno de Ejecución Común de Lenguajes (Common Language Runtime o CLR)

El Entorno de Ejecución Común de Lenguajes (Common Language Runtime o CLR), representa la parte central de .NET Framework y es el encargado de la ejecución del código de las aplicaciones.

Algunas de las características del CLR son:

- Proporciona un desarrollo de aplicaciones más sencillo y rápido gracias a que gran parte de las funcionalidades que tradicionalmente debía de crear el desarrollador, vienen implementadas en el entorno de ejecución.
- Administra el código en tiempo de ejecución, en todo lo referente a su carga, disposición en memoria, recuperación de memoria no utilizada a través de un recolector de basura, etc.
- Implementa características de gestión a bajo nivel (administración de memoria por ejemplo).
- Proporciona un sistema común de tipos para todos los lenguajes del entorno.
- Gestiona la seguridad del código que es ejecutado.
- Dispone de un diseño abierto a lenguajes y herramientas de desarrollo creadas por terceros fabricantes.

- Facilita la distribución e instalación de aplicaciones, ya que en teoría, es posible instalar una aplicación simplemente copiando los ficheros que la componen en uno de los directorios del equipo en el que se vaya a ejecutar, eliminando los conflictos de versiones entre librerías, problema conocido también con el nombre de Infierno de las DLL.

El CLR al ser el encargado de gestionar el código en ejecución utiliza todas los características mencionadas anteriormente.

El CLR provee lo que se llama código administrado, es decir, un entrono que provee servicios automáticos al código que se ejecuta (ver Figura 1.2). Entre los servicios están:

- Cargador de Clases: Permite cargar en memoria las clases.
- Compilador MSIL a nativo: Transforma código intermedio de alto nivel independiente del hardware que lo ejecuta a código de máquina propio del dispositivo que lo ejecuta.
- Administrador de Código: Coordina toda la operación de los distintos subsistemas del CLR.
- Recolector de Basura: Elimina de memoria objetos no utilizados.
- Motor de Seguridad: Administra la seguridad del código que se ejecuta.
- Motor de Depuración: Permite hacer un seguimiento de la ejecución del código aún cuando se utilicen lenguajes distintos.
- Verificador de Tipos: Controla que las variables de la aplicación usen el área de memoria que tienen asignado.
- Administrador de Excepciones: Maneja los errores que se producen durante la ejecución del código.
- Soporte de multiproceso (threads): Permite ejecutar código en forma paralela.
- Empaquetador de COM: Coordina la comunicación con los componentes COM para que puedan ser usados por el .NET Framework.
- Soporte de la Biblioteca de Clases Base: Interfase con las clases base del .NET Framework.


Figura 1.2 Servicios proporcionados por el CLR

### 1.3.3.2 Librerías de Clases Base

Las librerías de .NET Framework es una colección de clases reutilizables que se integran estrechamente con el CLR. La biblioteca de clases está orientada a objetos, lo que proporciona tipos de los que su propio código administrado puede derivar funciones. Esto ocasiona que los tipos de .NET Framework sean sencillos de utilizar y reduce el tiempo asociado con el aprendizaje de las nuevas características de .NET Framework. Además, los componentes de terceros se pueden integrar sin dificultades con las clases de .NET Framework[7].

Por ejemplo, las clases de colección de .NET Framework implementan un conjunto de interfaces que puede usar para desarrollar sus propias clases de colección. Éstas se combinarán fácilmente con las clases de .NET Framework.

Como en cualquier biblioteca de clases orientada a objetos, los tipos de .NET Framework permiten realizar diversas tareas de programación comunes, como son la administración de cadenas, recopilación de datos, conectividad de bases de datos y acceso a archivos. Además de estas tareas habituales, la biblioteca de clases incluye tipos adecuados para diversos

[7] Michael Amudsen, Paul Litwin, **Creación de Sitios Web con ASP.NET**, Edit. Prentice Hall 1a. Edición, Madrid 2002, Pág. 19-22.

escenarios de desarrollo especializados. Por ejemplo, se puede utilizar .NET Framework para desarrollar los siguientes tipos de aplicaciones y servicios:

- Aplicaciones de consola
- Aplicaciones GUI de Windows (Windows Forms)
- Aplicaciones de ASP.NET
- Servicios Web XML
- Servicios de Windows

Por ejemplo, las clases de Windows Forms son un conjunto completo de tipos reutilizables que simplifican enormemente el desarrollo de interfaces GUI para Windows. Si escribe una aplicación Web Form de ASP.NET, se pueden utilizar las clases de Web Forms.

Los principales grupos de librerías son:

- Clases de Sistema (ver Figura 1.3): Las clases de sistema ofrecen los servicios básicos necesarios para soportar el resto de las clases de las librerías. Esto incluye tareas de bajo nivel como el threading, seguridad, acceso remoto y seriación.

  También incluye tareas como el acceso a servicios en red, generales de E/S, colecciones, configuración, globalización[8], diagnostico y tratamiento de texto.

  Las clases de sistema contienen todas las funciones y servicios para manejar operaciones matemáticas y cadenas básicas. También se ocupan de la gestión de colecciones a través de un gran número de clases para manejar pares, listas y arreglos. Además, existen clases para ocuparse de otros recursos, como las imágenes y la información de la configuración.

  Las clases del sistema se encargan de las tareas de comunicación con el CLR para servicios en tiempo de ejecución, manejando tareas de interoperabilidad entre el

[8] La globalización se refiere a tener elementos de configuración como idioma, tipo de moneda y otras configuraciones regionales de otros países para el desarrollo de un sistema. Para mayor información consultar **http://www.c-sharpcorner.com/Code/2004/Jan/GloblizationInDotNEt.asp**

CLR y el sistema que lo aloja, seriación de objetos y datos y la información remota a través de procesos y servidores. La tarea de suministrar reflexión (una manera que tienen las clases de inspeccionar el contenido de unas y de otras en tiempo de ejecución) también se gestiona con las clases de sistema.

Finalmente, las clases del sistema manejan las tareas generales de los recursos, tales como la globalización, diagnóstico, configuración y entradas y salidas básicas, así como los servicios de red.

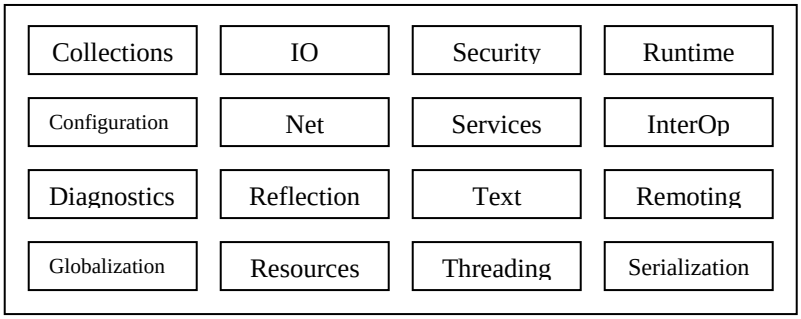



Figura 1.3 Clases de sistema del .NET Framework

- Clases de datos y XML (ver Figura 1.4): Las clases System.Data incluyen soporte para bases de datos conectadas OleDB, así como un conjunto optimizado de clases para acceder a Microsoft SQL Server directamente (SqlClient y SqlTypes). Estas clases de datos componen lo que Microsoft denomina ADO.NET.

  Como complemento a las clases de datos, las clases System.Xml ofrecen soporte para Schema, XPath, XLS y seriación en general para datos XML. Finalmente, todos los servicios de datos comparten una clase común para el manejo de tareas básicas.

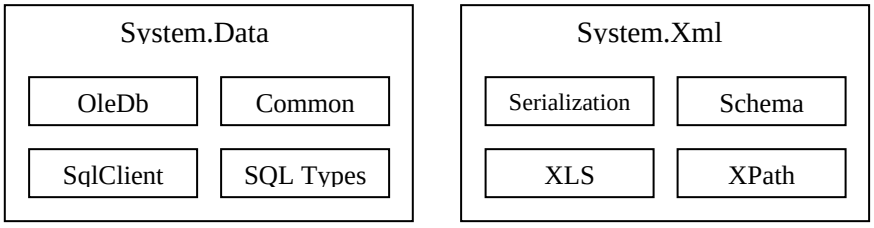



Figura 1.4 Clases de datos y XML del .NET Framework

- Clases de formularios Windows y de dibujo (ver Figura 1.5): Este grupo de clases ofrecen acceso a servicios básicos del escritorio como dibujar, imprimir, diseño de diálogos y soporte a los componentes de interfaz de usuario.

  Estas clases se pueden emplear para la creación de imágenes, la manipulación, la generación de fuentes complejas y otros servicios de dibujo. Las clases de dibujo también ofrecen acceso a la impresión y tareas de manejo de texto.

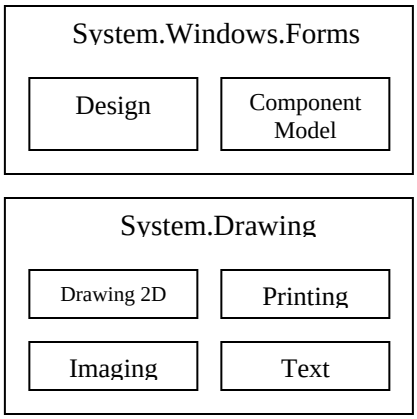



Figura 1.5 Clases de formularios Windows y de dibujo del .NET Framework

- Clases Web (ver Figura 1.6): Las clases web son las que manejan la interfaz del usuario y las tareas de Servicios Web en las aplicaciones web.

  Hay un conjunto de clases de interfaz de usuarios básicas que proveen soporte para páginas web, controles HTML y los nuevos controles de formularios web. Estas clases contienen todos los controles HTML estándar junto a otra clase de controles avanzados, como cuadrículas, listas de datos, controles de calendario y otros servicios de presentación de alto nivel. También hay un grupo de clases para dar acceso a los servicios web XML basados en SOAP, como protocolos SOAP, descubrimiento dinámico de Servicios Web, gestión de llamadas SOAP al Proxy, etc. Esto constituye la parte central del modelo de Servicios Web XML para soluciones web.

Finalmente, hay clases para gestionar tareas típicas de aplicaciones web tales como son la configuración de la aplicación, gestión de la sesión y servicios de seguridad basados en web.

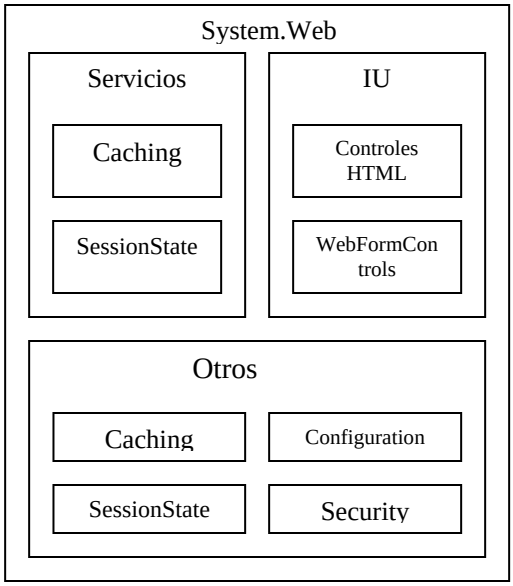



Figura 1.6 Clases web del .NET Framework

### 1.3.4 La Ejecución de Código Dentro del CLR y el Lenguaje Intermedio

Durante el proceso de compilación (ver Figura 1.7), el código fuente es tomado por el compilador del lenguaje utilizado para su escritura, y convertido, no directamente a código binario, sino a un lenguaje intermedio, que recibe el nombre de Microsoft Intermediate Languaje (MSIL o IL).

Este lenguaje o código intermedio, generado por el compilador, consiste en un conjunto de instrucciones que son independientes del sistema operativo o procesador en el que vaya a ejecutarse el programa, y que se ocupan de la manipulación de objetos, accesos a memoria, manejo de excepciones, etc.

Además del código en IL, el compilador genera también metadatos, los cuales contienen información adicional a la aplicación que no forma parte del código ejecutable (como descripción del ensamblado, versión, clave y tipos que lo componen, etc.) y que serán utilizados por el CLR al ejecutar el programa.

Tanto el código en IL, como los metadatos generados, se guardan en un fichero de tipo EXE o DLL, basado en la especificación tradicional de Microsoft para ficheros con formato de ejecutable transportable (Portable Executable) y objeto común (Common Object File Format). Con el desarrollo de la tecnología .NET, esta especificación ha sido ampliada para dar cabida, además de código binario, código IL y metadatos.

Ya que el código obtenido en IL es independiente del procesador, en su estado actual no es posible todavía ejecutarlo, debido a que el IL no ha sido diseñado para conocer las instrucciones específicas del procesador en el que se va a ejecutar. Antes de realizar la ejecución, el código en IL debe ser convertido a código máquina, utilizando lo que se denomina un compilador instantáneo o compilador Just-In-Time (JIT compiler), que es el encargado de generar el código binario específico para el procesador en el que el programa será ejecutado.

Para optimizar la ejecución y mejorar su velocidad, el compilador JIT se basa en el hecho de que es posible que ciertas partes del código que compone la aplicación nunca sean ejecutadas. Por este motivo, al ejecutar la aplicación, no se toma todo su IL y se compila, sino que sólo se compila el código según se va necesitando y se almacena el código máquina resultante de modo que esté accesible en las siguientes llamadas

Durante la carga de la aplicación, el cargador de código del CLR, toma cada tipo incluido en el programa, y para cada uno de los métodos que componen el tipo, crea y pega una etiqueta indicativa de su estado.

En la primera llamada a un método, se comprueba su estado de compilación a través de la etiqueta de estado; como aún no está compilado, se pasa el control al JIT, que compila el código IL a código máquina. A continuación se modifica la etiqueta de estado, de modo que en las próximas llamadas a ese método, la etiqueta de estado informa que el método ya ha sido compilado, por lo que se evita repetir el proceso de compilación, ejecutando directamente el código máquina creado con anterioridad. Esta técnica optimiza notablemente la velocidad de ejecución.

Ya que el código máquina ejecutable, es obtenido a través de un compilador JIT, con las instrucciones adecuadas para un procesador determinado, .NET Framework proporciona varios compiladores JIT para cada una de las plataformas que soporta, consiguiendo así que la aplicación, una vez escrita, pueda funcionar en distintos sistemas operativos, y haciendo realidad el objetivo de que nuestro código sea independiente de la plataforma en la que se vaya a ejecutar, actuando .NET Framework como una capa intermedia, que aísla el código del sistema operativo.

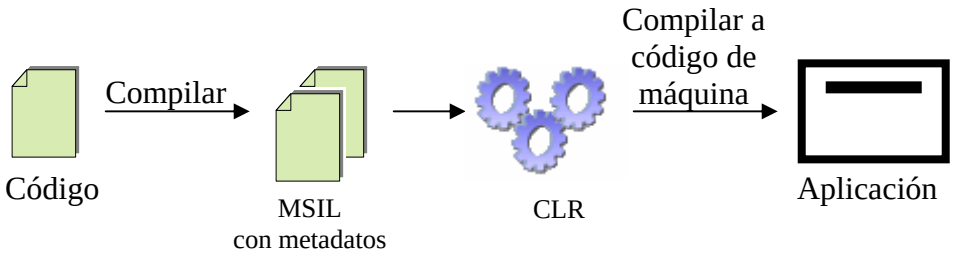



Figura 1.7 Proceso de compilación

### 1.3.5 Espacios de Nombres (NameSpaces)

Un espacio de nombres o Namepaces (ver Figura 1.8), también denominado nombre calificado, es el medio proporcionado por la plataforma para organizar las clases dentro del entorno, agrupándolas de un modo más lógico y jerárquico.

```
System.Data (namespace)
        DataRow   (Clase dentro del namespace)
        DataSet      (Clase dentro del namespace)
        DataTable  (Clase dentro del namespace)
System.Data.SqlClient (namespace)
        SqlConnection   (Clase dentro del namespace)
        SqlDataAdapter (Clase dentro del namespace)
```

Figura 1.8 Ejemplo de un namespace

### 1.3.6 Ensamblados (Assembly)

Un ensamblado o assembly, consiste en un conjunto de tipos y recursos, reunidos para formar la unidad más elemental de código que puede ejecutar el entorno de .NET Framework.

Los ensamblados se presentan como los bloques de construcción software, que se unen o ensamblan para crear aplicaciones. Una aplicación desarrollada para .NET Framework debe estar compuesta por uno o varios ensamblados.

Podemos establecer una analogía entre un ensamblado y una DLL, ya que ambos contienen clases, que se exponen a otras aplicaciones. Por dicho motivo, a un ensamblado también se le da el nombre de DLL lógica; el término DLL se emplea porque tiene un comportamiento similar al de las DLL's tradicionales, y el término lógica porque un ensamblado es un concepto abstracto, ya que se trata de una lista de ficheros que se referencian en tiempo de ejecución, pero que no se compilan para producir un fichero físico, a diferencia de lo que ocurre con las DLL's tradicionales.

Sin embargo, un ensamblado extiende su funcionalidad, ya que puede contener otros elementos aparte de clases, como son recursos, imágenes, etc.

Por otro lado, simplifican los tradicionales problemas de instalación y control de versiones sobre los programas, uno de los objetivos de la tecnología .NET, en la que en teoría, para instalar una aplicación, sólo sería necesario copiar los ficheros que la componen en un directorio de la máquina que la vaya a ejecutar.

Un ensamblado está compuesto por los siguientes elementos (ver Figura 1.9):

1. Manifiesto del ensamblado, que contiene información acerca de los elementos que forman el ensamblado. Ya que uno de los imperativos de la tecnología .NET, radica en que todos los componentes que se ejecuten dentro de la plataforma sean auto

descritos, esto es, que no necesiten de elementos exteriores al propio componente para obtener información acerca del mismo, la forma que tienen los ensamblados de proporcionar esta información, es a través de metadatos contenidos en su interior. Los metadatos de un ensamblado reciben el nombre de manifiesto. Un manifiesto contiene la siguiente información:

- Nombre. Una cadena con el nombre del ensamblado.
- Versión. Número de versión.
- Cultura. Información sobre idioma y otros elementos culturales que soporta el ensamblado.
- Nombre seguro. En el caso de ensamblados compartidos, este nombre permite identificar al ensamblado a través de una clave.
- Lista de ficheros. Los nombres y un resumen de cada uno de los ficheros que forman el ensamblado.
- Referencia de tipos. Información que usa el entorno para localizar el código IL de cada uno de los tipos que contiene el ensamblado.
- Ensamblados referenciados. Lista de los ensamblados con los que el actual mantiene dependencias.

De los puntos que acabamos de describir, los cuatro primeros forman lo que se denomina la identidad del ensamblado.

2. Metadatos sobre los tipos que contiene el ensamblado.

3. Módulos de código con los tipos compilados en IL.

4. Recursos adicionales.

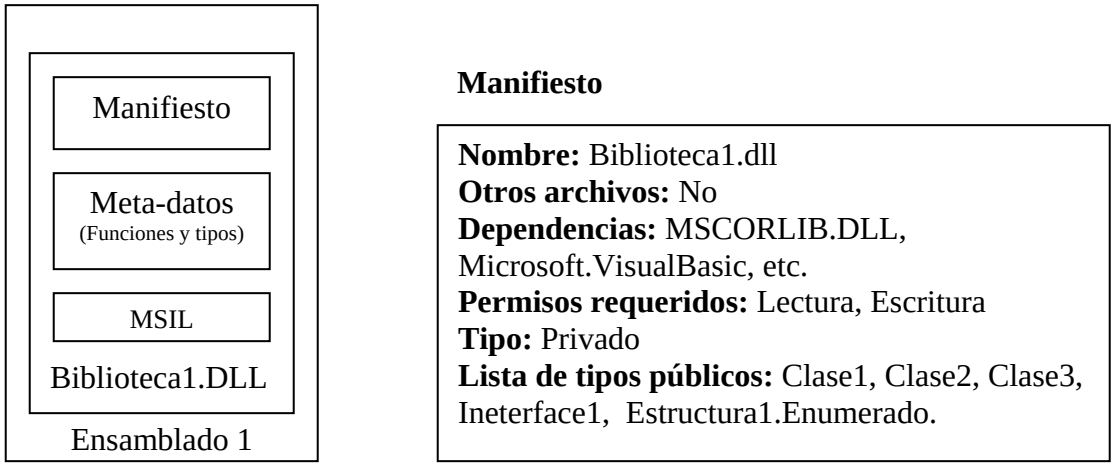



Figura 1.9 Estructura de un ensamblado.

### 1.3.7 Seguridad

El .Net Framework ofrece a los desarrolladores y administradores de sistema un control total sobre la seguridad de sus aplicaciones y los recursos que éstas deben o pueden usar. A los desarrolladores el Framework les permite utilizar un conjunto de herramientas de fácil uso para poder implementar en sus aplicaciones sistemas de autenticación, autorización y criptografía.

Entre los sistemas de seguridad que proporciona el .NET Framework están:

- Seguridad basada en evidencias.
- Seguridad de acceso al código (CAS).
- Seguridad basada en roles.
- Criptografía.
- Dominios de Aplicación.

#### 1.3.7.1 Seguridad Basada en Evidencias

Los conceptos fundamentales dentro de la seguridad basada en evidencia son tres:

- La política o policy de seguridad es configurada por los administradores de sistemas que establecen unos valores o parámetros conforme a los cuales se concede el acceso a ciertos recursos. La política de seguridad es administrada a través de la herramienta Caspool.exe (Code Access Security Policy Tool) que nos ofrece el Framework o a través de la herramienta de configuración del framework. Los administradores establecen las políticas de seguridad para los ensamblados y dominios de aplicación; el CLR utiliza la evidencia (que explicaré posteriormente) para identificar un ensamblado y a continuación utiliza la política de seguridad para determinar los permisos que el ensamblado tiene en tiempo de ejecución.
- Los permisos son esenciales para las políticas de seguridad. Describen los derechos asociados a los distintos objetos (recursos). El Framework de .Net ofrece métodos para la petición y concesión de acceso a través de los permisos. En tiempo de ejecución los ensamblados requieren unos permisos de acceso y éstos serán concebidos o denegados por el CLR una vez valorada las evidencias conforme a lo establecido por las políticas de seguridad.
- Los ensamblados traen con ellos metadatos entre los que se encuentra las evidencias. El CLR examina estos metadatos y si es posible les asignará a los ensamblados los permisos solicitados a través de la evidencia. Esta evidencia puede venir dada a través de distintas fuentes, como puede ser un certificado digital firmado por la empresa desarrolladora del ensamblado, o el nombre del espacio de nombres del ensamblado o la procedencia del código.

#### 1.3.7.2 Seguridad de Acceso al Código (CAS)

La seguridad de acceso del código se preocupará sólo de las aplicaciones administradas. El CAS se encargará de que la seguridad no sea comprometida en ningún momento por la ejecución de una aplicación y también se encargará de que las aplicaciones no excedan los

permisos que le han sido otorgados. Cuando un ensamblado es cargado le es asignado una serie de permisos, cuando una parte del código del ensamblado quiere tener acceso a un determinado recurso se comprueba que el ensamblado tiene el permiso para ello, en caso de que no lo tenga se lanzará una excepción de seguridad.

#### 1.3.7.3 Seguridad Basada en Roles

Del mismo modo que la seguridad de acceso al código se basa en la identidad del código, la seguridad basada en roles se basa en la identidad del usuario que está ejecutando el código. La seguridad basada en roles supone una forma de configurar y reforzar la autorización al usuario de tal forma que sólo a ciertos usuarios se les permite la realización de ciertas operaciones.

A la hora de ejecutar un ensamblado de .Net dos son las preguntas que debe hacerse la plataforma .net durante la ejecución. Una de ellas es si el ensamblado o código en sí mismo tiene o se le ha concedido el permiso para acceder y la otra sería si el usuario que está ejecutando el código tiene o se le ha concedido el permiso para acceder.

#### 1.3.7.4 Criptografía

.NET Framework ofrece soporte para realizar tareas de encriptación que pueden ser usadas en los procesos de autenticación y autorización a través de una serie de librerías. Algunos de los algoritmos de encriptación que soporta son:

- DES
- SHA
- AES
- RC2
- RSA y DSA
- MD5 ...

A parte de las tareas de encriptación, estas librerías también pueden ser usadas para tareas de creación de claves, creación de firmas digitales, generación de números aleatorios.

#### 1.3.7.5 Dominios de Aplicación

A través de estos dominios de aplicación el CLR es capaz de proporcionar óptimos niveles de aislamiento a la aplicación mientras ésta se está ejecutando impidiendo entre otras cosas que el código no pueda acceder o saltar de forma arbitraria a otras direcciones de memoria. Lo que se hace es separar una parte de una aplicación de otra parte. Además para contribuir aún más a ese aislamiento de las aplicaciones, el dominio de aplicación aporta también otra función adicional que consiste en que cuando la aplicación deja de estar ejecutándose, los ensamblados que cargó dejan de estarlo. Una ejecución puede contener diversos dominios de aplicación. La ejecución de un código en un dominio no puede afectar directamente a otras aplicaciones puesto que el código se carga en un dominio de aplicación y se ejecuta de acuerdo con la política de seguridad de ese dominio.

Para el trabajo no se utilizó ningún tipo de seguridad del .NET Framework en vez de esto se utilizó la seguridad integrada de Windows y la seguridad basada en formularios de ASP.NET (ver Capitulo 5). Pero se listaron para mostrar que el .NET Framework también cuenta con sus propios tipos de seguridad.

# CAPITULO 2

# ASP .NET

En la versión 5.0., Visual Basic incursionó como herramienta de desarrollo para la Web, a través de los llamados documentos ActiveX. Los documentos ActiveX brindaban la posibilidad de migrar una aplicación Windows en una forma relativamente sencilla, a los efectos de que la misma tomara algún beneficio de las características del explorador. Básicamente este último servia como un contenedor de la aplicación, pero las ventajas obtenidas no iban más allá de esto. El resultado concreto era que no se obtenía una integración real entre la Web y la aplicación[9], y debido a ello, esta tecnología no tuvo prácticamente adeptos, descartándose paulatinamente.

En la versión 6.0 de Visual Studio, Microsoft añadió 2 nuevas características al mismo, las cuales estaban directamente relacionadas con el desarrollo para Internet:

1. Diseñador de páginas DHTML.

2. Aplicaciones IIS.

La primera tecnología permite explotar los recursos del cliente (explorador) mediante la creación de aplicaciones que utilizan un conjunto de objetos ofrecidos por Internet Explorer. Microsoft básicamente integro al explorador de un conjunto de bibliotecas factibles a ser utilizadas desde Código Script el cual se incluye en la página HTML. A esta tecnología se le llama HTML dinámico, dado que gran parte del conjunto de objetos brinda la posibilidad de ejecutar tareas en forma dinámica, sin necesidad de realizar una nueva petición al servidor. Si bien la característica es de gran utilidad para algunos desarrollos, la realidad concreta es que el conjunto de objetos propuestos por Microsoft y Netscape cuenta con diferencias importantes, lo que hace difícil la construcción de aplicaciones compatibles con ambos exploradores[10].

[9] La integración no se logra debido a que la aplicación sólo esta contenida y no utiliza ninguna característica del explorador.
[10] Erich R. Bühler, **Visual Basic .NET Guía de Migración y Actualización**, Edit. Mc. Graw Hill, Madrid 2002, Pág. 684.

La segunda opción propuesta por el producto permite al desarrollador crear una aplicación ejecutable, la cual debe ser instalada en el servidor de Internet, y es vista por el navegante como una página estándar, aunque en realidad se trata de una página de resolución dinámica (ASP). Básicamente, cuando el servidor recibe una petición del explorador, la biblioteca es ejecutada, y su respuesta es enviada nuevamente al cliente. Esto permite mayormente emplear los recursos del servidor para realizar un gran conjunto de funciones. Si bien Microsoft se encontraba por buen camino[9], acababa de constituirse como los primeros pasos hacia una integración real con Internet y otros dispositivos vinculables al mismo. Sin embargo, la tecnología de aplicaciones IIS cuenta con algunas restricciones significativas, y una de las más importantes reside en que la página a ser resuelta en forma dinámica debe ser incluida dentro del ejecutable, por lo que se hace muy difícil la posterior modificación de las mismas. Por otra parte, el modelo completo de objetos ofrecidos por el servidor no siempre esta disponible para una aplicación IIS, y adicionalmente la tarea de construcción de la interfaz grafica es difícil.

Visual InterDev ofrece la posibilidad de implementar verdaderas soluciones Web, pero a un alto coste: el de tener que rediseñar la aplicación. Este producto hace posible escribir páginas con ciertas secciones a ser resueltas en forma dinámica por el servidor (igual que las aplicaciones IIS), pero adicionalmente brinda controles para facilitar la creación de la interfaz gráfica, así como también para enlazar los mismos a datos. Lamentablemente, la tecnología propuesta cuenta con tres puntos en contra: el primero reside en que las aplicaciones no son compiladas sino interpretadas, por lo que el resultado final tiene siempre un rendimiento inferior al de un ejecutable real. El segundo, en que las funcionalidades residen dentro de cada página como “islas” independientes, lo que dificultaba la reutilización del código. En ciertas ocasiones se refiere a esta característica como código spaghetti, debido a que el mismo esta generalmente repetido en varias secciones de la página, y mezclado con información de presentación.

El tercer punto reside en que las herramientas  para la depuración de código brindadas por el producto no siempre funcionan de la forma esperada, requiriéndose una inversión adicional de tiempo para llevar a cabo exitosamente dicha tarea.

Fue entonces cuando Microsoft comenzó a trabajar sobre estos puntos, a los efectos de llegar a un modelo más conveniente y consistente, el cual incluyera lo mejor de cada tecnología. El resultado final se dio a conocer a comienzos del año 2002 y se llamó ASP .NET.

ASP .NET incluye varias mejoras con respecto a su antecesor directo (ASP), las cuales aceleran los tiempos de desarrollo y ofrecen un modelo más consistente.

## 2.1 ASP y ASP.NET

ASP (Active Server Page) ha brindado durante varios años la posibilidad de llevar a cabo soluciones para Internet en forma relativamente sencilla. Básicamente, una página ASP es un documento de texto escrito por Visual InterDev o en el Bloc de notas (se puede utilizar cualquier editor de texto) el cual contiene secciones a ser reemplazadas por el servidor en el momento en que un cliente realiza una petición a la misma. ASP transforma al servidor de Internet en una especie de supermáquina de escribir, la cual es capaz de reemplazar un trozo, o generar la totalidad de su contenido.

El desarrollador lleva adelante la tarea mediante la inclusión de marcas o secciones a ser resueltas por el servidor, las cuales generalmente incluyen lenguajes tales como VbScript, JavaScript ó JSCript. El resultado final obtenido por el explorador es generalmente una página con etiquetas HTML. La ventaja principal con que cuenta esta tecnología es que es fácil de usar, ya que basta con conocer HTML, Visual Basic ó Java para poder escribir una página de este tipo. También ofrece varias de las ventajas del modelo en tres capas[11], como la reducción de los costes de instalación y/o actualización del producto, dado que la solución sólo reside en el servidor. No obstante, ASP cuenta con algunas desventajas, como la poca flexibilidad con la que cuenta para reutilizar un trozo de código común a varias páginas, y ello es debido a que cada una de éstas actúa como una “isla” independiente dentro de la solución.

[11] Erich R. Bühler, **Visual Basic .NET Guía de Migración y Actualización**, Edit. Mc. Graw Hill, Madrid 2002, Pág. 688.

En el caso de que una función o procedimiento requiera ser utilizado en varias secciones de una misma aplicación, o bien se copia y pega ella en las diferentes partes, o bien se encapsula en una biblioteca COM.

Como desventaja adicional, también se obtiene que la lógica (código) y la presentación (controles visuales) son siempre almacenadas dentro del mismo documento, lo que dificulta seriamente su mantenimiento. Ello es debido a que ASP no permite hacer uso de un archivo que contenga la información de presentación y otro para la implementación, o por lo menos no en forma sencilla[10].

Como última desventaja, y quizá una de las más importantes, el desarrollador debe someterse a utilizar dos ambientes diferentes, uno para crear aplicaciones para Windows (Visual Basic, etc.) y otro para escribir soluciones para la Web (Visual Interdev, etc.).

Por otro lado ASP .NET es la tecnología propuesta por Microsoft para enfrentar los desafíos de interconexión entre dispositivos y sitios Web. Visual Studio .NET ofrece la posibilidad de crear aplicaciones para Web, haciendo uso del mismo entorno utilizado para desarrollar aplicaciones para Windows.

Las técnicas brindadas para dibujar un formulario de Windows no difieren en mucho de las ofrecidas para realizar la misma tarea en una página Web. Se cuenta con un nuevo tipo de proyecto llamado Aplicación Web ASP .NET, el cual hace uso de formularios Web o Web Forms, los cuales cumplen un comportamiento similar al de los formularios estándares en el modelo Windows. Un formulario Web es exhibido de igual forma que uno Windows, pero, en realidad, el primero es gestionado internamente como una página de servidor activo y etiquetas HTML. Para crear la interfaz basta con crear un nuevo proyecto de este tipo, y posteriormente arrastrar los controles del cuadro de herramientas y así crear una interfaz gráfica para la Web.

Una de las características más importantes de ASP .NET es que pueden emplearse dos archivos independientes, uno para la presentación y otro para contener la implementación o

lógica de los diferentes eventos, lo que permite que ambas partes puedan ser tratadas de forma independiente. Por otra parte, el modulo que contiene el código es ahora compilado como ensamblado, adquiriendo así todas las características de los mismos (ver Figura 2.1). Una página de servidor activo ASP .NET es vista por el modelo como un conjunto de datos de presentación, más un ensamblado conteniendo la lógica.

El modelo cuenta con varias mejoras sustanciales con el manejo de la persistencia de información (estado), ya que agrega la posibilidad de que los diferentes componentes que integran una página puedan mantener en forma automática su contenido a través de las diversas peticiones de la misma.

Cuando se produce un error en una página de servidor activo ASP .NET, el programa se encarga de incluir como respuesta de la misma un conjunto de información con respecto al motivo que lo provocó, lo que hace más fácil su detección y eliminación.

En resumen ASP .NET es un vasto conjunto de tecnologías que permiten desarrollar aplicaciones para la Web, y que integran en forma natural todas las características del modelo .NET.

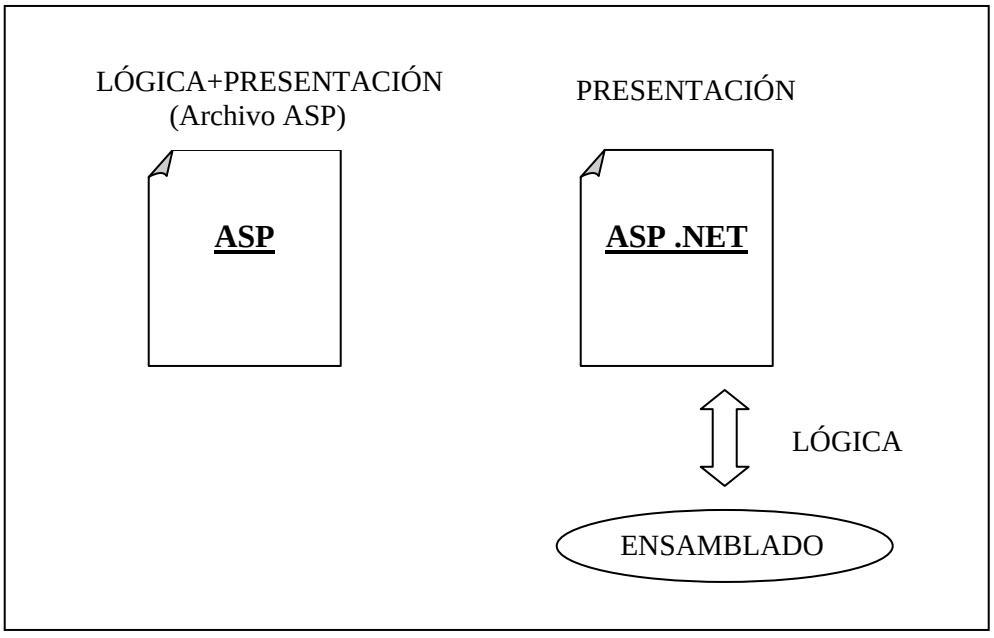



Figura 2.1 Diferencia entre ASP y ASP .NET

## 2.2 Anatomía de un Proyecto ASP .NET

Al crearse una Aplicación Web ASP .NET por defecto se incluyen los siguientes módulos:

- **AssemblyInfo.<Extensión dependiendo el lenguaje>:** Contiene información sobre el proyecto, como el título, la descripción, compañía, etc.

- **Carpeta de referencias:** Se pueden ver los diferentes ensamblados o espacio de nombres requeridos para que la misma se ejecute correctamente.

- **<Nombre de la aplicación>.vsdisco:** Llamado módulo de descubrimiento, y es de suma utilidad cuando se emplean Servicios Web.

- **Web.Config:** Contiene información sobre diferentes preferencias a ser aplicadas al proyecto en el momento de la ejecución del mismo. El archivo esta dividido internamente en varias secciones, y cada una de ellas permite configurar un aspecto diferente del funcionamiento de la aplicación. Modificando una de ellas es posible cambiar la página que será exhibida en caso de que se produzca un error, asi como también cambiar la cultura a utilizar, u otros aspectos aun más complejos.

- **Styless.css:** Define el estilo global (fuente, color, etc.) a emplear por los diferentes elementos de los formularios Web.

- **Global.asax:** Este archivo es conocido también bajo el nombre de archivo de aplicación de ASP .NET. El mismo contiene procedimientos, los cuales son iniciados en el ámbito de toda la aplicación (proyecto), sin importar la página que solicite el usuario. Y existe un archivo oculto llamado Global.asax.<Extensión dependiendo el lenguaje> que contiene el código a ser compilado.

- **WebForm1.aspx.:** El archivo WebForm1.aspx contendrá los datos de presentación, mientras que el WebForm1.aspx.<Extensión dependiendo el lenguaje> contendrá la implementación de los diferentes eventos.

## 2.3 Proceso de Solicitud de una Página ASP .NET

Cuando una página es solicitada al servidor de Internet a través de la Web, esta observa la extensión de la misma, y en el caso de que sea ASPX, deriva la tarea a una biblioteca llamada aspnet_isapi.dll, la cual es la encargada de gestionar las páginas de contenido ASP .NET. Esta toma la misma y posteriormente crea un objeto dentro de la infraestructura .NET que la representa, el cual es en realidad derivado de la clase Page. Como último paso ejecuta sus métodos y eventos, y luego envía el código HTML resultante al servidor, que luego lo traslada al cliente (ver Figura 2.2).

Es por ello que ASP .NET considera cada página dinámica como un objeto de tipo Page, el cual tiene su propia representación en la infraestructura.

Hay que tomar en cuenta que cada vez que se invoca la página, ya sea su primer acceso o debido a un evento, la misma debe ser regenerada (recursos y contenido), por lo cualquier modificación sobre ella tiene que ser efectuada nuevamente.

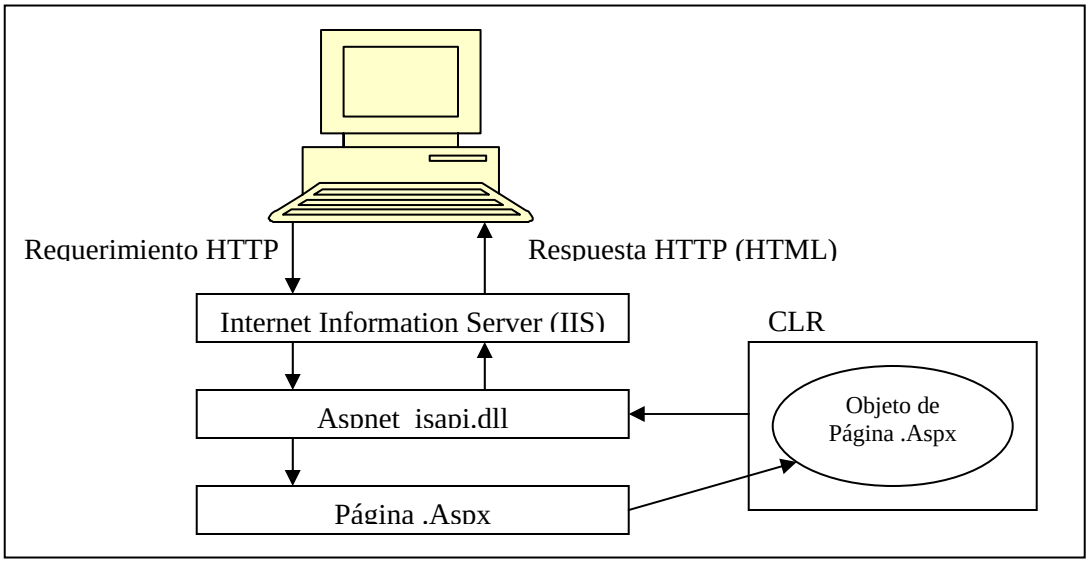



Figura 2.2 Proceso de solicitud de una página aspx

## 2.4 Elementos nuevos en ASP .NET

### 2.4.1 Formularios Web

Desde las primeras versiones del producto, la palabra formulario se ha utilizado como sinónimo de ventana de Windows, y específicamente como metáfora para representar la tela en donde dibujar los diferentes controles. En la plataforma .NET el concepto se ha ampliado, y ahora incluye dos tipos: formularios para Windows y formularios Web (ver Figura 2.3), el primer tipo constituye una ventana de Windows. El segundo tipo corresponde a una nueva aproximación tomada por Microsoft, la cual es utilizada exclusivamente para la tecnología de ASP .NET.

La idea principal consiste en dibujar los diferentes integrantes de la aplicación en forma similar a lo que se hacia con el primer tipo (arrastrando y soltando los controles sobre el mismo). La diferencia fundamental radica en que posteriormente el entorno trasladara dicha información a un modulo ASP .NET conteniendo HTML bien formado, en vez de llamar a funciones de Windows para recrear la ventana.

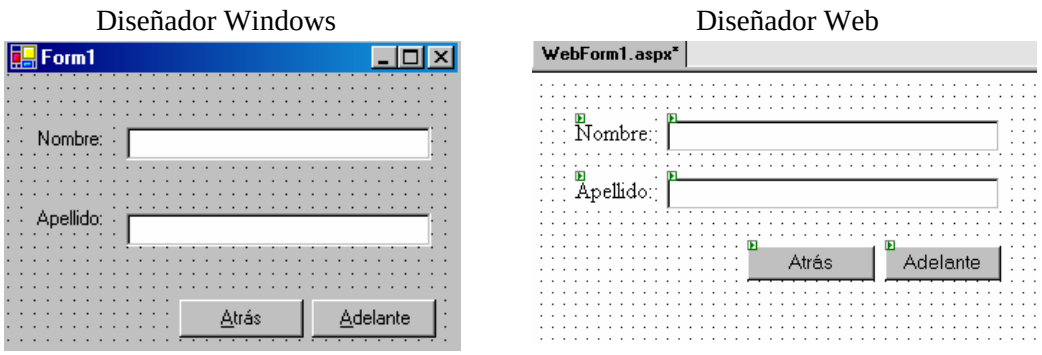



Figura 2.3 Formularios de la plataforma .NET

### 2.4.2 Eventos en Formularios Web

Los eventos en ASP .NET brindan la sensación al usuario de que se producen en el mismo lugar donde se sitúa la interfaz gráfica; sin embargo, la realidad es que ambas partes están ubicadas en diferentes lugares. La integración de una aplicación para Windows reside en el mismo ordenador, y utiliza funciones internas del sistema operativo para crear sus ventanas y sus controles. A diferencia, las aplicaciones ASP .NET crean la interfaz en el servidor a través de una página, la cual posteriormente es enviada al explorador o cliente a través del protocolo HTTP, y empleado como formato de archivo HTML.

Existe una funcionalidad ofrecida por HTTP, la cual hace posible enviar al servidor no solamente información de la petición de un recurso, si no también el contenido de la página actual que tiene el explorador. Esto es muy común cuando se desea enviar a otra página valores de la página activa. Para dicho fin, HTTP provee una alternativa diferente, la cual permite cargar en el encabezado del mensaje la página solicitada, y en el cuerpo en contenido de los diferentes controles. A esto se le llama método Post, dado que permite enviar información al servidor. Este tipo de funcionalidades son gestionados por HTML mediante la utilización del elementos o marcas especificas, las cuales permiten delimitar aquellos controles a ser enviados del explorador al servidor (por ejemplo la etiqueta Form).

Cada vez que un evento es iniciado en un formulario Web, el mismo no es ejecutado localmente dentro del explorador, si no que es enviado al servidor que es donde reside la implementación del mismo. Cada evento produce un evento Post, el cual hace que se lleve acabo una ida al servidor, enviando el contenido de los controles de la página. Una vez que la información es recibida por el servidor, el mismo identifica al recurso que inicio el evento, y posteriormente localiza la implementación del mismo y la ejecuta. Por ultimo, crea nuevamente la página y la envía al explorador, incluyendo las modificaciones necesarias realizadas por el código ejecutado.

Hay alguna tareas que realiza ASP .NET con el fin de minimizar las idas y vueltas al servidor de Internet, y así redundar en un mejor rendimiento. Una de las más importantes

radica en que muchos de los eventos son dejados en espera hasta que otro evento sea iniciado, con el cometido de que ambos sean procesados en forma conjunta. A esto se le llama eventos en espera, y es una característica exclusiva de ASP .NET.

## 2.5 Controles en ASP .NET

### 2.5.1 Controles de Formularios Web

Los controles de formularios Web son componentes que dan como resultado uno o más controles HTML ante la petición de un explorador. Todos ellos heredan sus características de diferentes clases, las cuales residen dentro del espacio de nombres (namespace) System.Web.UI.WebControls, el cual contiene todos aquellos elementos que pueden componer la interfaz gráfica.

Los controles Web están pensados para abstraer al desarrollador de los componentes o etiquetas HTML que serán empleadas al momento de plasmar la interfaz, ya que en muchas ocasiones los mismos se ajustaran en forma dinámica de acuerdo al explorador que el usuario este utilizando. Estos controles están pensados para ofrecer un modelo similar al de Windows, en el cual se configuran propiedades, y se utilizan métodos. Además de que pueden iniciar eventos a ser capturados del lado del servidor.

Estos controles pueden ser divididos en cuatro grandes grupos de acuerdo con su utilidad[12]:

1) Estándares
2) Complejos.
3) Validación
4) Vinculables a datos

[12] Erich R. Bühler, **Visual Basic .NET Guía de Migración y Actualización**, Edit. Mc. Graw Hill, Madrid 2002, Pág. 720-735.

1) Controles estándares

Los controles estándares son aquellos que se transforman en un control HTML al ser enviado del servidor al explorador, o en su defecto simulan uno en el caso de que no exista su contraparte. Cada uno de ellos es en forma automática convertido en una marca HTML, la cual puede depender del explorador que este realizando la petición de la página.

Los controles estándares son:

| NOMBRE DEL CONTROL | UTILIDAD PRINCIPAL |
|---|---|
| TextBox | Capturar texto del usuario, o exhibirlo de sólo lectura. |
| Button | Ejecutar código mediante el evento clic. |
| Hyperlink | Invocar otra página. Este control se transforma en una etiqueta HTML de hipervínculo, y no soporta la ejecución del evento clic del lado del servidor. |
| Label | Mostrar un texto dentro de la página. El control es trasladado a un elemento HTML llamado span. |
| Image | Exhibir una imagen dentro de un formulario Web. |
| DropDownList | Brindar un conjunto de opciones al usuario, de las cuales deberá seleccionar sólo una. |
| CheckBoxList, RadioButtonList | Facilita la creación de una colección visual de casillas de verificación u opción. |
| Panel | Permite contener un conjunto de controles o exhibir un texto alineado. |
| LinkButton | Ofrece la misma funcionalidad que un botón, pero exhibiendo un hipervínculo. |

2) Controles Complejos

Los controles complejos ofrecen en general mayores funcionalidades, ya que resuelven muchas de las tareas empleando un gran conjunto de etiquetas HTML y recursos exclusivos del explorador, así como también del servidor.

Los controles complejos son:

| NOMBRE DEL CONTROL | UTILIDAD PRINCIPAL |
|---|---|
| Calendar (Calendario) | Ofrece las características de un calendario estándar. |
| XML | Permite aplicar transformaciones XLS a un documento XLM, sin necesidad de hacer uso de código. |
| CrystalReportViewer (Visor de informes Crystal) | Ofrece la posibilidad de mostrar un informe utilizando un explorador. |
| AdRotator (Rotor de avisos) | Brinda la posibilidad de rotar un conjunto de imágenes. |

3) Controles de validación

Una tarea muy común es la de validar que el contenido introducido por el usuario en los diferentes campos de la página sea correcto, lo que generalmente incluye que no estén vacíos y que cumplan con ciertas pautas preestablecidas. En las aplicaciones Web gran parte del código escrito es en realidad utilizado para verificar los valores de los diferentes controles, a los efectos de conocer si se encuentran dentro de las condiciones esperadas. Con el fin de eliminar la necesidad de hacer uso de este tipo de técnicas, ASP .NET ofrece una aproximación diferente, la cual se basa en utilizar un conjunto de controles Web, los cuales permiten verificar que el contenido introducido dentro de un control cumpla con ciertos criterios.

Un componente de validación se enlaza con el control a validar en forma “parásita”, y posteriormente se le asigna el texto que este deberá de exhibir en el caso de que el contenido no siga las pautas establecidas.

En tiempo de ejecución, el componente de validación verificará el contenido del control asociado, y en el caso de que el mismo no coincida con el o los patrones dados, exhibirá el mensaje de error previamente establecido, y se configurara como no valido.

Los controles de validación son:

| NOMBRE DEL CONTROL | UTILIDAD PRINCIPAL |
|---|---|
| RequieredFieldValidator | Se utiliza para comprobar que un control tiene contenido |
| CompareValidator | Facilita la comprobación de que dos controles contengan los mismos valores. |
| RangeValidator | Permite verificar que el contenido de un control asociado al mismo está dentro de un rango numérico previamente establecido. |
| RegularExpressionValidator | Las expresiones regulares permiten definir un patrón para un texto, como la conformación de una dirección de correo electrónico, postal, etc. |
| CustomValidator | Permite aplicar una validación personalizada asociando una función, la cual puede residir en Código Script del lado del cliente, o como evento del lado del servidor. |

4) Controles Vinculables a Datos

Si bien algunos controles estándares permiten enlazarse y exhibir un conjunto de información obtenida de un origen de datos, en realidad no ofrecen mayores beneficios de personalización. A diferencia, los controles vinculables a datos ofrecen una mayor

flexibilidad sobre los mismos, dado que han sido exclusivamente construidos para dicha tarea. Como característica adicional, parte de los mismos son totalmente funcionales, lo que quiere decir que se puede interactuar con las filas exhibidas por ellos (eliminación, modificación, etc.), y su posterior sincronización con el origen de datos.

Los controles vinculables a datos son:

| NOMBRE DEL CONTROL | UTILIDAD PRINCIPAL |
|---|---|
| DataGrid | Es uno de los controles más importantes que ofrece ASP.NET, brinda la posibilidad de crear una aplicación con características de enlace a datos. El mismo se debe vincular a un origen de datos para exhibir y en dado caso manipular información. |
| DataList | Es similar al DataGrid, salvo que permite personalizar aún más la forma en que los datos serán exhibidos. |
| Repeater | Es similar al DataList, ya que permite exhibir un conjunto de filas en una página Web, pero como de sólo lectura, aunque ofrece mayor flexibilidad para exhibir información. |

### 2.5.2 Controles HTML

La aproximación tomada por los controles HTML es diferente a los controles Web, dado que existe una relación de uno a uno con la marcación que será utilizada en el momento de construir este en la página. A su vez, éstos no detectan el tipo de explorador que realiza la petición, por lo que no adecuan el resultado final a ser enviado al mismo. No obstante, estos controles cuentan con un punto importante, que radica en la relación de uno a uno, la cual facilita la utilización de Código Script del lado del cliente.

Los controles HTML y Web pueden ser utilizados en forma conjunta, debido a que ambos finalmente son transformados en un documento de marcas.

Por defecto, un control HTML no puede contener Código Script del lado del servidor, y para que esto sea posible el mismo tiene que ser convertido a un control HTML de servidor. Una vez que el control ha sido transformado a uno de servidor, este se comportara de forma similar a uno Web desde el punto de vista de sus eventos. Sin embargo, el mismo no gana sus características adicionales (sus métodos y propiedades siguen siendo los mismos).

## 2.6 Características Especificas de ASP .NET

Existen algunas funcionalidades disponibles en forma exclusiva para el modelo de aplicaciones de ASP .NET, las cuales no pueden ser empleadas en otro tipo de proyectos (aplicación para Windows, biblioteca de clases, etc.). Entre estas características están:

- Clase httpResponse: Por defecto cuando una página está siendo construida en el servidor, la misma es alojada en memoria y posteriormente enviada al cliente (explorador) que realizo la petición a ella. Este objeto es capaz de: enviar una respuesta al explorador, a través de la escritura directa a la página de retorno; brindar la posibilidad de conocer si el cliente todavía se encuentra a la espera de la petición realizada; y la de redirigir en forma automática a otra página a los usuarios.

- Clase httpRequest: Asi como el objeto Response centra sus utilidades sobre la información a ser enviada al explorador, el objeto Request permite obtener aquellos datos que son recibidos por éste. En ASP este objeto era de suma utilidad, pero en ASP .NET muchas de las tareas son realizadas en forma automática, por lo que se prescinden gran parte del mismo.

Los objetos Request y Response pueden ser utilizados en forma directa, sin la necesidad de que los mismos tengan que ser definidos.

## 2.7 Gestión de Estado

La gestión de estado es una característica fundamental ofrecida por los exploradores y servidores de Internet relativamente modernos. La idea fundamental consiste en hacer persistir la información del usuario a través de diferentes técnicas. Las mismas pueden variar de un proveedor a otro, aunque se puede decir que existe un cierto estándar al respecto.

Como se vio anteriormente, cada ida y vuelta al servidor genera nuevamente la página, por lo que el servidor no almacena por defecto información para identificar si una determinada petición corresponde al mismo usuario que ya hubiese realizado algún otro requerimiento. Debido a ello este protocolo es llamado “sin-estado”, ya que el mismo no almacena por defecto información con respecto a la conexión, como es el caso de un enlace a base de datos, en donde el motor se encarga de guardar las características de seguridad y otros datos del usuario conectado. Aquí cada petición es tratada como una independiente, y una nueva solicitud no difiere en nada de una anterior o posterior.

Para poder mantener información en el servidor de Internet ASP .NET proporciona:

- Variables de aplicación (Application): Este objeto permite mantener valores globales en el ámbito de todas las peticiones (por ejemplo, número de visitantes a un sitio). Básicamente, la información asignada al objeto es almacenada en la memoria del servidor de Internet, y mantenida siempre que el servicio de IIS no sea reiniciado. Para hacer persistencia de un valor, basta con indicarle a éste el nombre que identificará al elemento. Estos objetos no requieren ser declarados, y debido a ello brindan una libertad total, ya que estos son creados tan pronto como un valor y clave son establecidos.

- Variables de sesión (Session): Este objeto sirve para mantener un valor exclusivo para un usuario, pero solamente mientras dure la conexión del mismo sitio. Las variables de sesión son utilizadas generalmente para guardar las preferencias del

usuario mientras éste navega a través de las diferentes páginas de la aplicación. Igual que las variables de aplicación estas no son declaradas y son creadas tan pronto como un valor es asignado, y destruida una vez cumplido un tiempo máximo de inactividad.

- Variables de cookie (Cookie): En muchos casos, se busca que la información del usuario persista incluso después de que el mismo finalice con la sesión. Los modelos planteados anteriormente resultan de utilidad para mantener la información de manera temporal, pero muy restrictivos a la hora de tener que mantener estos por un tiempo indeterminado. Muy comúnmente un sitio necesita guardar las preferencias del usuario, a los efectos de tomar estas en futuras visitas del mismo. Para dicho fin, se cuenta con una tecnología llamada cookie.

  Una cookie es un conjunto de datos que son guardados dentro del explorador. Básicamente, cuando el servidor de Internet recibe una petición a una página, este puede indicar en el paquete de retorno al explorador que el mismo deberá hacer persistencia a un conjunto de valores. Posteriormente la próxima vez que el mismo se conecte al sitio el servidor recibirá dentro del paquete la información que fue previamente guardada por él.

# CAPITULO 3

# BASES DE DATOS

En las instituciones (escolares, empresariales, etc.) se maneja una gran cantidad de datos, por lo que se hace necesario disponer de Hardware y Software que permitan acceder a la información de una manera rápida, sencilla y fiable.

Tradicionalmente, la información se almacenaba en conjuntos de ficheros. Estos ficheros no guardaban ninguna relación entre sí, los datos podían repetirse de unos a otros, lo que se suponía la existencia de información redundante y, en algunos casos, inconsistente. A veces se precisaba cambiar la estructura de los registros de algunos ficheros, por ejemplo para agregar nuevos campos a los registros. Esto implicaba que todos los programas de aplicación que se servían de esos ficheros debían modificarse y adaptarse con la correspondiente pérdida de tiempo y dinero. Existía dependencia entre los ficheros que almacenaban la información y los programas que utilizaban esos ficheros.

A finales de los años sesenta surgieron las bases de datos. En una base de datos se almacenan los datos que necesite una institución, y los programas que manejan esos datos no se han de preocupar del almacenamiento físico de los mismos: ningún cambio en la estructura de los datos afectará a los programas de aplicación que los utilicen.

En las bases de datos existe una visión conceptual de los datos que no tiene por qué ser la misma que la visión física de estos. Es decir, existe una independencia de los datos con respecto a los procedimientos que los utilizan, de modo que ningún cambio en la estructura de datos afectará a los programas que los usen. Los datos se pueden utilizar por diferentes aplicaciones y usuarios. La base de datos ha de permitir métodos para consultar los datos, incluir nuevos datos, modificar los datos existentes y eliminar los datos que ya no sirven.

### 3.1 Propósitos de la Organización de una Base De Datos

Una base de datos, independientemente del modo en que esté organizada, debe cumplir los siguientes propósitos para que sea considerada como tal.

- **Versatilidad para representar la información:** Los datos podrán utilizarse de múltiples maneras. Los usuarios que accedan a la base de datos usarán éstos de diferentes maneras dependiendo de la información que precisen. Así un programador utiliza un conjunto de datos de distinto modo que un usuario que necesite únicamente realizar una consulta. El sistema de administración de datos ha de permitir derivar datos y relaciones a partir de los ya existentes.

- **Desempeño:** Las bases de datos han de atender con la rapidez adecuada las peticiones de datos que se hagan, según el uso que se vaya a hacer de ellos.

- **Redundancia mínima:** En los sistemas de procesamiento de datos tradicionales existía un alto nivel de redundancia, lo que hacia que los mismos datos apareciesen repetidos en varios archivos, incrementando el coste de almacenamiento y ocasionando inconsistencia entre los datos repetidos, ya que, en un momento dado, podían tener valores diferentes. Un objetivo de las bases de datos es eliminar la redundancia, siempre y cuando esto no implique un aumento de su complejidad ni una reducción en su rendimiento. La redundancia se elimina con el uso de claves que identifiquen cada registro (clave primaria) para evitar su duplicidad.

- **Capacidad de acceso:** Los usuarios de bases de datos continuamente están pidiendo información acerca de los datos almacenados. Si realizan una consulta y necesitan una respuesta rápida, el sistema de base de datos ha de tener capacidad para responder inmediatamente. Esta capacidad depende de la organización física de los datos. Los diseñadores de bases de datos deben lograr una capacidad de acceso rápido. Con una organización física bien diseñada, los accesos a la base de datos serán rápidos.

- **Integridad:** Los datos de una base de datos pueden ser usados por muchos usuarios y de diferentes maneras, el sistema de base de datos ha de asegurar que los datos que se almacenan sean los que se tienen que almacenar; que los valores

almacenados cumplan ciertas restricciones definidas con anterioridad, y que los fallos producidos en el sistema no destruyan los datos. Con este fin, los procedimientos de almacenamiento, inserción y actualización han de asegurar que el sistema pueda recuperarse en caso de fallo sin que se pierda información.

- **Seguridad y privacidad:** La información almacenada en la base de datos es de gran valor. Esto explica la necesidad de disponer de procedimientos que protejan los datos frente a fallos del sistema, evitando su pérdida y facilitando su recuperación, o frente a usos indebidos o intencionados por parte de personas no autorizadas. La privacidad hace referencia a que los datos de la base de datos sean accesibles para unos usuarios e inaccesibles para otros, para quines no están autorizados.

- **Afinación:** Con el tiempo, el volumen de datos va aumentando, por lo que la organización física de éstos ha de ser buena, y es posible que haya que ajustarla para atender a las nuevas necesidades de almacenamiento. La afinación hace referencia al ajuste de la organización física de los datos con el objeto de mejorar la rapidez para acceder a ellos. Para llevar a cabo una correcta afinación es precisa una independencia física de los datos que no suponga modificaciones en los programas de aplicación u otras representaciones de datos, y sí, en cambio, una mayor rapidez de acceso a los datos.

- **Interfaz con el pasado y el futuro:** Con el paso del tiempo, las necesidades de la empresa van cambiando, se precisan nuevos datos, etc., por tanto, se hace necesario actualizar el sistema de procesamiento. Cuando se decide instalar un software de base de datos es importante que éste pueda trabajar con los datos ya almacenados. Igualmente, una base de datos de datos ha de estar preparada para los futuros cambios de los datos y de los medios de almacenamiento. Cuando se introduzcan modificaciones en la organización física de los datos, no deben afectar a los programas de aplicación que ya hay en uso: es lo que se llama independencia física de los datos. Y cuando se hacen cambios en la estructura

lógica de los datos, por ejemplo, agregar nuevos campos a un registro, éstos no deben afectar a los programas de aplicación que utilicen esos datos: esto es la independencia lógica de los datos.

### 3.2 Arquitectura de las Bases de Datos

Los usuarios de una base de datos deben tener una visión lo más abstracta posible de los datos almacenados en ella; no tienen necesidad de saber cómo están organizados y almacenados los datos.

La base de datos debe presentar al usuario una visión de los datos de tal manera que sea capaz de interpretarlos y manejarlo. Se pueden señalar tres niveles, según la visión que se tenga de los datos en una base de datos:

- Nivel interno. Es la representación más cercana al almacenamiento físico de los datos. Permite describirle tal y como están almacenados en el ordenador. Este nivel se describe mediante el esquema interno. En este nivel se describen los ficheros que contienen la información, su organización, ubicación, la forma de acceso a sus registros, los tipos de registros, su longitud, los campos que lo componen, los índices, etc.

- Nivel conceptual. Es la representación de los datos que intervienen en el problema. Este nivel se representa mediante el esquema conceptual. Todos los datos se completan a partir de los requerimientos de los usuarios, definiendo una visión global de los mismos. No se tiene en cuenta la organización física ni los métodos de acceso.

- Nivel externo. Es el nivel más cercano a los usuarios. Representa la visión individual de un usuario o grupo de usuarios de la base de datos. En él se describe la parte de los datos que se interesa a un usuario o a un grupo de usuarios.

Para una base de datos especifica hay un esquema interno y conceptual pero puede haber esquemas externos, cada uno definido para uno o para varios usuarios (ver Figura 3.1).

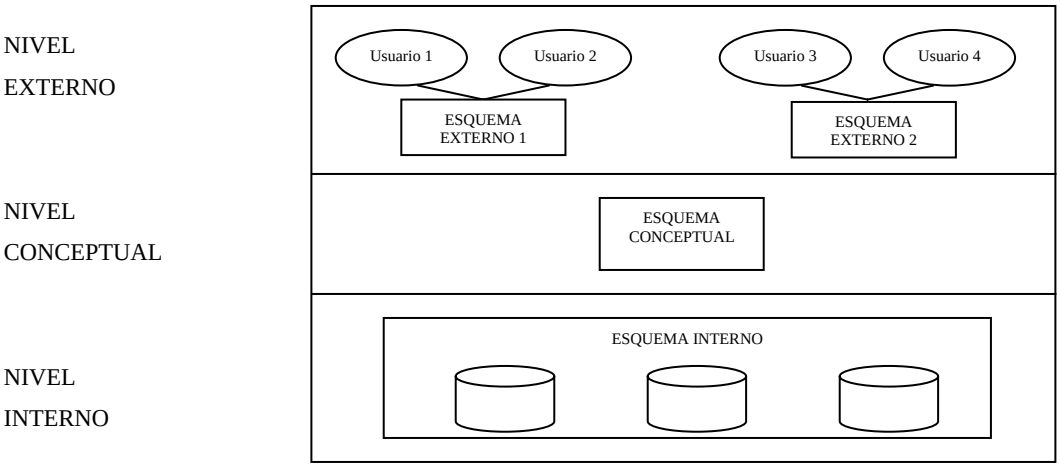



Figura 3.1. Niveles de abstracción en la descripción de una base de datos

### 3.3 Sistemas Gestores de Base de Datos

Un sistema de gestión de base de datos (SGBD) es un conjunto de programas que permiten administrar y gestionar la información de una base de datos. Proporcionan a los usuarios de las bases de datos facilidades para realizar las siguientes tareas:

- Definición de los datos en los distintos niveles de abstracción.
- Manipulación de datos, es decir, inserción, actualización, borrado y consulta.
- Mantenimiento de la integridad de la base de datos. Con la integridad se hace referencia a los datos en sí, sus valores y sus relaciones: los datos que se almacenen en la base han de satisfacer las restricciones definidas en el esquema de ésta.
- Control de la privacidad y seguridad de los datos en la base de datos, posibilitando el acceso a los mismos sólo a los usuarios autorizados.

### 3.3.1 Componentes

Para realizar las funciones que se acaban de describir y algunas más, el SGBD necesita, además de herramientas software, personal que gestione de manera adecuada la información almacenada en la base de datos. Estos componentes son los siguientes:

- El gestor de la base de datos

  Es un conjunto de programas transparentes al usuario que se encargan de garantizar la privacidad, la seguridad, la integridad de los datos, el acceso concurrente a ellos, así como de interactuar con el sistema operativo. El gestor proporciona una interfaz entre los datos almacenados, los programas que manejan esos datos y los usuarios. Cualquier operación que queremos realizar "contra" la base de datos ha de estar procesada por el gestor.

  El gestor almacena en el diccionario de datos la descripción de la base de datos, los usuarios permitidos y las autorizaciones pertinentes. Existe un usuario administrador encargado de centralizar estas tareas.

- El diccionario de datos

  El diccionario de datos en una base de datos es donde se almacena toda la descripción de la base datos, información referente a la estructura de los datos, relaciones entre ellos, gestión e implantación de la base de datos. Este diccionario debe contener todo lo que cualquier usuario quiere saber sobre la base de datos:

  - Las descripciones externa, conceptual e interna de la base de datos.
  - Las transformaciones entre los tres niveles.
  - Las restricciones sobre los datos.
  - El acceso a los datos.
  - Las descripciones de las cuentas de usuarios.

- Las autorizaciones de cada usuario.
- Los esquemas externos de cada programa, sus usuarios y qué autorizaciones tienen.

- El administrador de la Base de datos

El administrador de la base de datos (DBA o ABD) es una persona (o un grupo de personas) responsable de la seguridad y el control de los datos, los cuales se pueden utilizar por cualquiera que tenga autoridad para ello. Si un programador quiere crear un nuevo objeto de base de datos o modificar la estructura de un objeto existente, debe pedir autorización al administrador, que cederá los privilegios para que pueda modificar las estructuras de datos de la manera que crea más conveniente para el sistema.

Son tareas del administrador:

- La definición del sistema lógico de la base de datos mediante el uso de secuencias DDL que representen las características del problema que se va a tratar.
- La definición del esquema físico, de las estructuras de almacenamiento de los datos y de los métodos de acceso.
- La definición de los subesquemas o visiones de usuario de la base de datos.
- La concesión de autorización para el acceso a los datos mediante la asignación de privilegios a los usuarios.
- El mantenimiento de la seguridad de los datos almacenados en la base de datos a través de la especificación de los procedimientos necesarios para poner en marcha estrategias de recuperación de datos en caso de que algún fallo de lugar a su pérdida.
- El mantenimiento del esquema lógico y físico de la base de datos.

### 3.3.2 Los Lenguajes

El SGBD ha de proporcionar lenguajes para definir y manipular los datos de la base. Podrán utilizar estos lenguajes los administradores y los usuarios. Son los siguientes:

- Lenguaje de definición de Datos (DDL). Se utiliza para definir el esquema conceptual y los distintos subesquemas externos de la base de datos.
- Lenguaje de manipulación de datos (DML). Mediante este lenguaje podemos manipular los datos de la base de datos, podemos insertar datos, modificar los ya existentes, eliminar y recuperar datos almacenados.
- Lenguaje de Control de datos (DCL). Se utiliza para controlar el acceso a la información de la base de datos definiendo privilegios y tipos de acceso, así como para el control de la seguridad de los datos. De esta tarea se encarga el administrador.

Se trata de lenguajes que aportan una gramática sencilla, fácil de entender por usuarios no expertos. No todos los lenguajes DDL y DML son iguales para todas las bases de datos, pues dependen del modelo de datos que tenga el SGBD. También los SGBD pueden procesar peticiones DML formuladas en programas escritos en otros lenguajes con C, Cobol, etc.

## 3.4 Modelos de Bases de Datos

Una vez realizado el diseño conceptual de la base de datos, el esquema resultante ha de traducirse a un modelo lógico de datos. Los modelos más difundidos son el modelo de datos jerárquico, el modelo en red y el modelo relacional.

### 3.4.1 Modelo de Datos Jerárquico

El modelo jerárquico se sirve de árboles para la representación lógica de los datos. Un árbol está compuesto por una jerarquía de elementos llamados nodos. Cada nodo representa un tipo de registro llamado segmento con sus correspondientes campos.

En este modelo sólo existen relaciones 1:M (uno a muchos) que están representadas por las ramas del árbol, es decir, un padre puede tener varios hijos, pero un hijo sólo puede tener un padre. Por tanto, este modelo presenta muchos problemas a la hora de representar las relaciones N:M (muchos a muchos).

Algunos inconvenientes del modelo jerárquico son los siguientes:

- No se permite más de una relación entre dos segmentos.
- No se admiten relaciones del tipo N:M.
- No se permite que un segmento hijo tenga más de un padre.
- El árbol se recorre en un cierto orden.
- Para acceder a cualquier segmento es necesario hacerlo por el segmento raíz.

Son ejemplos de bases jerárquicas el IMS del IBM y el SYTEM 2000 de Intel (año 2000).

### 3.4.2 Modelo de Datos en Red.

Este modelo utiliza estructuras de datos en red, también conocidas como estructuras plex. Las entidades se representan como nodos, y las relaciones como líneas que unen los nodos. En una estructura red cualquier componente puede vincularse con cualquier otro. Es posible describirla en términos de padres e hijos, pero, a diferencia del modelo jerárquico, un hijo pude tener varios padres.

El modelo de red más importante es el que propuso DBTG (Data Base Task Croup, grupo de trabajo sobre base de datos del comité COBOL) de CODASYL (Conference on Data System Languages). Los conceptos básicos en el modelo CODASYL son:

- Tipo de registro: representa un nodo.
- Elemento: es un campo de datos. Ejemplo: DNI.
- Agregados de datos: conjunto de datos con nombre. Ejemplo: Fecha (día, mes, año).
- Conjunto: un conjunto relaciona dos tipos de registro. Uno de ellos se llama propietario y el otro, miembro.

El conjunto permite representar relaciones 1:M o 1:1. En una relación 1:M, el tipo de registro propietario es el que contiene muchos miembros. Puede darse el caso de que en un registro miembro tenga más de un propietario. Para representar una relación N:M se necesita un registro de enlace o link que conecte los dos tipos de registro.

Ejemplos de bases de datos en red son el DMS 1100 de UNIVAC el EDMS de Xerox, el PHOLAS de Philips y el DBOMP de IBM.

### 3.4.3 Modelo de Datos Relacional

Un SGBD de enfoque relacional utiliza tablas bidimensionales para la representación lógica de los datos y las relaciones entre ellos. Está basado en la teoría matemática de las relaciones. Fue Codd quien desarrolló en IBM-San José el modelo de datos relacional[13]. Se trata de un modelo lógico de datos que hace referencia a la representación lógica de la información por tanto, no es directamente aplicable a la representación física.

Algunas de las ventajas del modelo relacional son las siguientes:

- Ofrece una visión conceptual sencilla del conjunto de datos existente en la base.

[13] Maria Jesús Ramos, Alicia Ramos y Fernando Montero, **Desarrollo de Aplicaciones en entornos de 4ta. Generación y con Herramientas Case**, Edit. Mc. Graw Hill, España 2000, Pág. 10.

- Puede atenderlo y usarlo cualquier usuario, no es preciso que sea un experto informático.
- Los usuarios no necesitan saber dónde se encuentran físicamente los datos.
- Se puede ampliar el esquema conceptual sin modificar los programas de aplicación.

El elemento principal del modelo relacional es la relación. Cada relación se representa mediante una tabla.

Ejemplos de base de datos relacionales son SYSTEM R Y QBE de IBM, ORACLE de RSI, INFORMIX, DBASE IV y Microsoft SQL Server.

#### 3.4.3.1 El Modelo Entidad-Relación

En el proceso de diseño de una base de datos se obtiene el esquema conceptual, en el que se definen todos los datos que intervienen en el problema y sus relaciones. El modelo conceptual más conocido es el modelo entidad-relación (E/R). Fue propuesto por Peter Chen en 1976 para la representación gráfica de los problemas que forman parte del mundo real. Utiliza un conjunto de símbolos y reglas para representar los datos y las relaciones entre ellos.

Una vez obtenido el modelo E/R del problema que se trata de resolver, el paso al modelo relacional, jerárquico o de red es sencillo. La estructura lógica de una base de datos se puede representar gráficamente usando este modelo.

#### 3.4.3.2 Elementos del Modelo Entidad-Relación

- **Entidad:** Una entidad es un objeto acerca del cual se recoge información de interés para la base de datos. Por ejemplo: ALUMNO se recogen los datos de los alumnos de un centro de estudios; por ejemplo, nombre, dirección o población.

Existen entidades fuertes y entidades débiles. Las entidades fuertes son aquellas que no dependen de otra entidad para su existencia. Por ejemplo, la entidad EMPLEADO. Las entidades débiles son las que dependen de otra entidad para su existencia. Es un ejemplo la entidad HIJO_DE_EMPLEADO, que depende de la entidad EMPLEADO.

Las entidades se representan gráficamente con rectángulos (Peter Chen 1976).

- **Relación:** Una relación es una asociación entre dos o más entidades. Suele tener un nombre que la identifica al resto de las relaciones, ya que cada relación tiene un significado específico. El grado de una relación define el número de entidades que participan en ella. Las relaciones se representan gráficamente con rombos.

- **Cardinalidad:** Define el número máximo y el número mínimo de ocurrencias de cada tipo de entidad que interviene en una relación. Para representar la cardinalidad se encierran entre paréntesis los valores máximo y mínimo.

  Los valores para la cardinalidad son: (0,1),(1,1),(0,N),(1,N) y (N,M). Por ejemplo, si un EMPLEADO pertenece a un DEPARTAMENTO y sólo a uno, la cardinalidad para la entidad DEPARTAMENTO es (1,1). Si a un DEPARTAMENTO pertenecen uno o más empleados, la cardinalidad para la entidad EMPLEADO es (1:N).

- **Atributo:** Un atributo es cada una de las propiedades o características de una entidad o una relación.

| Entidad | Atributos |
|---|---|
| Alumno | NCA<br>Nombre<br>Dirección<br>Teléfono |

Cada atributo se identifica por un nombre y por todos los posibles valores que puede tener. El atributo o conjunto de atributos que identifica a una entidad denominada atributo identificador principal. Por ejemplo, en la entidad ALUMNO, el atributo identificador principal puede ser el numero de clave del alumno (NCA).

Se representa gráficamente con un círculo unido a una línea que parte de la entidad o relación de la que forma parte.

#### 3.4.3.3 Estructura del Modelo Relacional

El modelo relacional es un modelo simple y potente que sirve para representar problemas. El formalismo y base matemática sobre la que está definido constituyen las bases sobre las que se sustenta la teoría de las bases de datos relacionales. La sencillez del modelo ha facilitado la construcción de lenguajes de consulta e interfaces de fácil manejo. El elemento principal del modelo relacional es la relación. Una base de datos relacional está formada por un conjunto de relaciones. A continuación se exponen los elementos del modelo relacional.

**Relación**

La relación se representa mediante una tabla. Una tabla es la estructura de datos que se utiliza para almacenar todos los datos de una base de datos relacional. Representa una entidad o una asociación de entidades. Una relación está formada por un conjunto de atributos llamados columnas y un conjunto de tuplas llamadas filas.

- Atributo (columna). Se trata de cada una de las columnas de la tabla. Las columnas tienen un nombre y pueden guardar un conjunto de valores. Una columna se identifica siempre por su nombre, nunca por su posición. El orden de las columnas en una tabla es irrelevante.

- Tupla (fila). Representa una fila de la tabla (ver Figura 3.2).

| NoEmple | Apellidos | Salario | Numdepart | FechaAlta |
|---|---|---|---|---|
| 13407877B | Milagros Suela | 20000 | 10 | 18/11/1990 |
| 41667891C | José María Cabello | 20000 | 20 | 29/10/1992 |

Figura 3.2. Tabla EMPLEADO con dos filas.

En una tabla no se admiten filas duplicadas. El orden de las filas en la tabla es irrelevante. De las tablas se derivan los siguientes conceptos:

- Cardinalidad: Es el numero de filas de la tabla. En el ejemplo anterior es dos.

- Grado: Es el numero de columnas de la tabla. En el ejemplo anterior el grado es cinco.

- Valor: Viene representado por la intersección entre una fila y una columna.

- Valor NULL: Representa la ausencia de información.

**Dominios**

Es el conjunto de valores que puede tomar cada atributo. Los valores contenidos en una columna pertenecen a un dominio que previamente se ha definido. Existen dos tipos de dominios:

- Dominios generales. Son aquellos cuyos valores están comprendidos entre un máximo y un mínimo.

- Dominios registrados. Son los que pertenecen a un conjunto de valores específico.

**Claves**

Toda fila debe estar asociada con una clave que permita identificarla. A veces la fila se puede identificar por un único atributo, pero otras veces es necesario recurrir a más de un atributo. La clave debe cumplir dos requisitos:

- Identificación unívoca. En cada fila de la tabla el valor de la clave ha de identificarla de forma unívoca.

- No-redundancia. No se puede descartar ningún atributo de la clave para identificar la fila.

En una fila puede existir más de un conjunto de atributos que cumpla los requisitos anteriores. Este conjunto se conoce como claves candidatas. Uno de esos conjuntos será elegido como clave primaria que identificará la fila. En el modelo relacional existen varios modelos de claves: candidatas, primarias y ajenas:

- Clave primaria (primary key). Es la columna o conjunto de columnas que permiten identificar cada ocurrencia de la tabla. No puede tener valores nulos.

- Clave ajena (foreing key). Está formada por una o más columnas de una tabla cuyos valores corresponden con los de la clave primaria de otra tabla. Las claves ajenas representan las relaciones entre tablas. El modelo relacional especifica la regla de integridad referencial para las claves ajenas que establece que los valores de la clave ajena son nulos o que han de coincidir con los valores de la clave primaria de la otra relación.

**Vistas**

Una vista es una tabla ficticia cuyas filas se obtienen a partir de una o de varias tablas que sirven de base. Lo que se almacena de una vista es su definición. Mediante las vistas se pueden obtener los datos que nos interesen de una o varias tablas.

**Paso del modelo E/R al modelo relacional**

Una vez obtenido el esquema conceptual mediante el modelo E/R, hay que definir el modelo lógico de datos. Los pasos son:

- Toda entidad se transforma en una tabla.
- Todo atributo se transforma en columna dentro de una tabla.
- El identificador único de la entidad se convierte en clave primaria.
- Toda relación N:M se transforma en una tabla que tendrá como clave primaria la concatenación de los atributos clave de las entidades que asocia.

El modelo relacional fue el que se usó en este trabajo debido a que el programa Microsoft SQL Server 2005 utiliza este modelo.

## 3.5 Arquitectura Cliente/Servidor

Independientemente de la arquitectura física, toda aplicación informática costa de al menos tres niveles funcionales, que son:

- Nivel de presentación. Este nivel trata con la interfaz entre el usuario y el sistema. El nivel de presentación es el responsable de aceptar los datos de entrada del usuario y mostrar al usuario la información.

- Nivel de la lógica de negocio. Éste es el nivel que añade realmente significado o valor a los datos subyacentes. Es aquí donde se validan los datos, se realizan los cálculos y tienen lugar otras rutinas de procesamiento de la información.

- Nivel de acceso a los datos. Este nivel es el responsable del almacenamiento físico y la extracción de los datos.

La forma en que estos niveles funcionales están distribuidos a través del hardware físico ayuda a definir la arquitectura. Antes incluso de que el término cliente-servidor fuera empleado para describir una arquitectura informática específica, ya existían los clientes y los servidores. El cliente es la entidad que solicita alguna información o servicio, mientras que el servidor es la entidad que proporciona la información o servicio al cliente.

Con la aparición de los sistemas de gestión de bases de datos relacionales (RDBMS, Relational Database Management System) fue, la tecnología clave que hizo posible la arquitectura informática cliente-servidor. El RDBMS servía como almacén centralizado para los datos de la organización. El RDBMS estaba diseñado para gestionar el acceso multiusuario a un conjunto compartido de datos. Toda la gestión de bloqueos y conexiones es llevada a cabo por el RDBMS, que también se encarga de la seguridad. El lenguaje SQL (Structured Query Languaje, Lenguaje Estructurado de Consulta) fue creado como lenguaje universal de programación para solicitar datos específicos a un RDBMS.

La arquitectura cliente-servidor fue realmente una combinación de las mejores funciones del entorno basado en host y del entorno de red de área local basado en PC. Esta arquitectura utiliza la potencia del PC para realizar la presentación de los datos, junto con el complicado procesamiento relativo a la lógica de negocio que añade valor a dichos datos. El RDBMS proporciona un área de almacenamiento centralizada para los datos, así como los servicios para gestionar el acceso compartido y concurre a dichos datos. La arquitectura cliente-servidor puede tomar muchas formas, dependiendo de cómo se elija separar los niveles de presentación, lógica de negocio y datos.

Internet está teniendo un profundo impacto en la forma de desarrollar sistemas. Internet ha permitido a los desarrollares crear complejas aplicaciones cliente-servidor accesibles desde cualquier parte del mundo a través de una conexión a Internet.

Los componentes básicos de una arquitectura cliente-servidor en Internet son un navegador Web, un servidor Internet y un conjunto de protocolos de transmisión y presentación de datos (TCP/IP, HTTP y HTML). El navegador Web actúa como cliente y es responsable, de aceptar los datos de entrada del usuario y de mostrar al usuario la información. El servidor Internet responde a las solicitudes de información de los clientes y procesa dichas solicitudes adecuadamente. Los protocolos TCP/IP, HTTP y HTML definen un estándar universal para la transmisión y presentación de información a través de Internet.

En su forma más simple, las aplicaciones cliente-servidor basadas en Internet pueden recordar a la arquitectura basada en mainframe o en host, con el navegador Web proporcionando capacidades de terminal no inteligente y siendo toda la lógica de negocio y el acceso a los datos proporcionados por un servidor central.

Internet puede ser también la plataforma para arquitecturas complejas cliente-servidor de n niveles. En una arquitectura de n niveles, el servidor Internet actúa simplemente como una pasarela que dirige las solicitudes del cliente a los objetos de negocio apropiados y transmite las respuestas de éstos de vuelta hacia los navegadores web. Esta arquitectura es similar a cualquier otra arquitectura de n niveles, en el sentido de la lógica de negocio de la aplicación está separada en un conjunto de objetos de negocio que pueden reutilizarse de una aplicación a otra. Internet proporciona una plataforma que permite realmente personalizar la arquitectura física de los sistemas para adaptarlos a las necesidades de cualquier organización.

En un sistema cliente–servidor de dos niveles típico, la aplicación cliente debe conectarse directamente a un RDBMS, como SQL Server. Esto quiere decir que cada estación de trabajo cliente debe ser cargada con bibliotecas y controladores específicos de cada fabricante para poder establecer conexiones con la base de datos. Las aplicaciones cliente

son también responsables de iniciar la sesión en el RDBMS y de mantener las conexiones, así como de gestionar los mensajes de error y los otros tipos de mensajes devueltos por el RDBMS.

El nivel de la lógica de negocio puede residir en el cliente, en el servidor o en ambos, en los sistemas de dos niveles.

El modelo cliente-servidor de dos niveles hace que surjan algunos problemas inherentes. Dichos problemas difieren dependiendo de la localización del nivel de la lógica de negocio.

A medida que las organizaciones crecen, suelen sufrir cambios. Estos cambios normalmente se reflejan en cambios de la lógica de negocio utilizada en las aplicaciones informáticas. Cuando el nivel de la lógica de negocio reside en la aplicación cliente, es difícil implementar dichos cambios. Cada vez que la lógica de negocio cambia, debe modificarse la aplicación cliente y redistribuirla a todas las estaciones de trabajo clientes. En una organización donde haya cientos o miles de dichas estaciones de trabajo, este proceso puede resultar todo un problema.

La aparición de los procedimientos almacenados solventó algunos de estos problemas. Los procedimientos almacenados permitieron a los programadores colocar el nivel de la lógica de negocio en el servidor. Si se permite que los procedimientos almacenados realicen las funciones de la lógica de negocio, las aplicaciones cliente no necesitan cambiar cuando las reglas de negocio cambian. Los procedimientos almacenados, sin embargo, pueden no proporcionar el nivel de sofisticación requerido por la lógica de un negocio concreto. Los lenguajes de programación de procedimientos almacenados no suelen proporcionar las mismas capacidades disponibles en los lenguajes de programación de aplicaciones.

La arquitectura cliente/servidor ha ido evolucionando en conjunción con los avances hardware y software de la informática, como son:

- La aparición de nuevas técnicas de almacenamiento.

- La mejora de las comunicaciones por red.
- La mejora de la tecnología de base de datos.

En un sistema cliente/servidor se distinguen dos partes: un servidor y un conjunto de clientes. Generalmente, un servidor es un gran ordenador que actúa como depósito de los datos y permite llevar a cabo todas las funciones de un sistema gestor de base de datos (SGBD). Los clientes son estaciones de trabajo o PC que solicitan servicios al servidor. Para poder comunicarse, éstos deben estar interconectados a través de una red LAN (ver Figura 3.3).

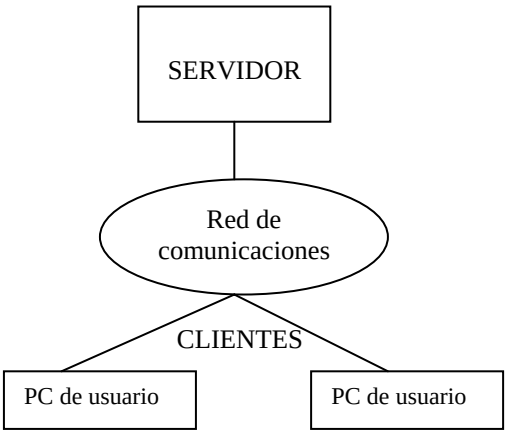



Figura 3.3. Arquitectura cliente/servidor.

El software adecuado para una arquitectura cliente/servidor posee varios componentes que se pueden asociar al cliente o al servidor. Son los siguientes:

- Software de gestión de datos. Lleva a cabo la manipulación y gestión de datos requeridos por las aplicaciones. Normalmente reside en el servidor.

- Software de interacción con el usuario y presentación. Implementa las funciones que se asocian a una interfaz gráfica del usuario (GUI). Suele residir en el cliente.

- Software de desarrollo. Se usa para desarrollar aplicaciones. Suele residir en el cliente.

Además de estos componentes, hay otros elementos software que existen en el cliente y en el servidor. Se trata de software de sistemas operativos en red, de aplicaciones de bases de datos, de comunicaciones, etc., que facilitan la conexión cliente/servidor.

La arquitectura cliente/servidor permite las configuraciones siguientes:

- Basada en anfitrión: La máquina cliente y la máquina servidora son la misma. Los usuarios se conectan directamente a la máquina donde se encuentra la base de datos.

- Cliente/servidor: La base de datos reside en una máquina servidora y los usuarios acceden a la base de datos desde su máquina cliente a través de una red.

- Procesamiento distribuido: La base de datos está repartida en más de una máquina servidora. Los usuarios no tienen por qué conocer la ubicación física de los datos con los que trabajan y acceden simultáneamente a varios servidores (ver Figura 3.4).

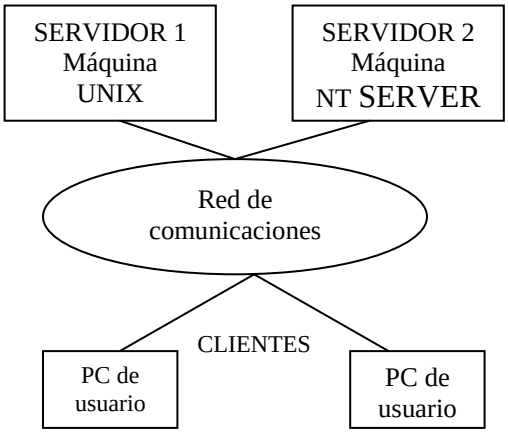



Figura 3.4. Base de datos con varios servidores y clientes.

- Basada en servidores de aplicaciones: esta configuración permite el uso de aplicaciones en redes de área amplia (WAN) e Internet. Hace posible que las aplicaciones se ejecuten en máquinas cliente que no requieren ninguna administración. Cualquier PC que ejecute un navegador Internet puede acceder a las aplicaciones. Éste es un tipo de arquitectura a tres niveles, pues se dispone de un servidor de base de datos, uno o varios servidores de aplicaciones y máquinas clientes (ver Figura 3.5).

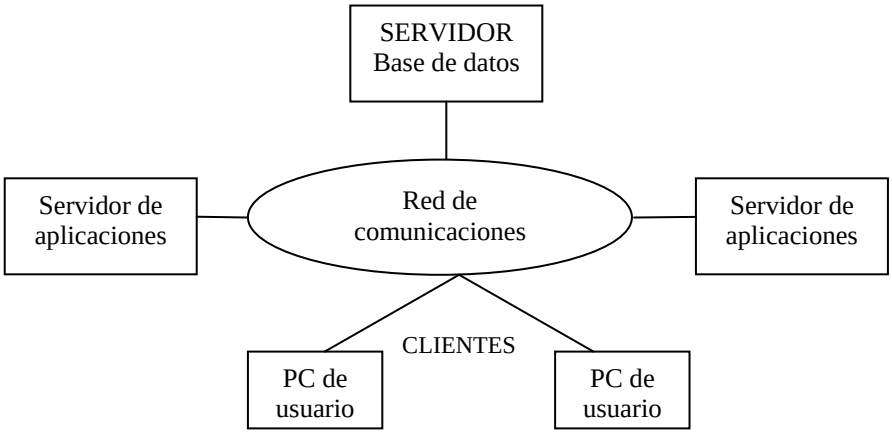



Figura 3.5. Basada en servidores de aplicaciones

La arquitectura cliente/servidor se diferencia de la arquitectura basada en servidores de aplicaciones en los siguientes puntos:

La arquitectura cliente/servidor requiere que las aplicaciones se instalen en cada puesto de trabajo, lo que ocasiona gastos de instalación y un aumento de los costes de administración. Además, impone grandes exigencias a la red, lo que imposibilita el uso de las aplicaciones en redes de área amplia (WAN) e Internet.

En la arquitectura basada en servidores de aplicaciones, éstas se instalan en puestos de trabajo que no requieren ninguna administración. Cualquier PC puede acceder al servidor de aplicaciones con un navegador web que soporte Java o Plataforma Microsoft .NET.

# CAPITULO 4

# ADO .NET

Inicialmente, el acceso a datos se realizaba llamando a funciones propias de cada manejador de base de datos, lo que hacia difícil la programación debido a que se debían de conocer las implementaciones que ofrecía cada uno de ellos. Con el paso de las versiones, Microsoft instauró diferentes modelos de objetos y tecnologías, con el fin de brindar el acceso a datos de forma más sencilla.

La primera versión de objetos se denominó DAO (Data Access Object), y estuvo originalmente ideada para interactuar con bases de datos de Microsoft Access, aunque posteriormente fue adaptada para consumir información desde orígenes remotos mediante ODBC.

La necesidad de integrar los lenguajes con motores de datos relacionales generó que Microsoft desarrollara un segundo conjunto de objetos denominado RDO (Remote Data Access), el cual estaba pensado principalmente para acceder a éstos en forma eficiente. Sin embargo, dicha aproximación daba como resultado que se debía apelar a uno u otro modelo dependiendo del motor que iba a ser utilizado.

Como siguiente paso, Microsoft comenzó a trabajar en un conjunto de nuevas tecnologías y objetos, entre los cuales se encontraba OLE-DB (el reemplazo de ODBC) y ADO (Activex Data Object). Ellas ofrecían una interfaz única para la programación de acceso a datos, sin importar el origen al cual se estuviera accediendo. Para construir el mismo se tomó lo mejor de las tecnologías DAO y RDO, y se obtuvo como resultado el conjunto de objetos de ADO.

ADO también ofrece algunas características avanzadas, con el fin de brindar cierta flexibilidad en la construcción de aplicaciones distribuidas, y específicamente para la gestión de cursores desconectados. Los cursores desconectados se basaban en que cualquier integrante de una aplicación podía realizar una consulta sobre un origen de datos y posteriormente desconectarse del mismo, sin que ello evitara el continuar trabajando con los resultados obtenidos. Esta característica ofrece dos grandes ventajas:

1. Era posible realizar modificaciones sin estar haciendo uso constante de una conexión, lo cual resguardaba parte de los recursos del servidor de datos. Por otra parte, la tecnología brinda el soporte necesario para reconectarse posteriormente al origen, y así efectuar los cambios realizados localmente.

2. Brinda la posibilidad de utilizar la misma estructura como medio de transporte entre el origen y la aplicación, o entre los componentes que integrasen ésta, lo cual hace posible emplear el mismo objeto en toda la aplicación.

Sin embargo, dicha aproximación plantea un problema, el cual reside en que una vez obtenidos los datos de las diferentes tablas, estos son tratados en forma desasociada. Esto es un problema cuando se debe consultar la información con la misma estructura de parentesco en el cual el origen la gestiona. Este inconveniente es realmente importante cuando se necesita acceder a ellas en forma desconectada, con la misma apariencia que se tiene en forma conectada. Debido a ello, Microsoft remodeló el conjunto de objetos, con el fin de solucionar este y otros problemas, lo que dio como resultado ADO .NET.

ADO .NET plantea una evolución natural de los modelos anteriores, ya que se basa principalmente en la manipulación de información de uno o más orígenes de datos, y su posterior gestión en forma desconectada. La idea primaria radica en utilizar conexiones únicamente bajo demanda, lo que quiere decir que la aplicación estará por defecto desconectada, y que solamente cuando se requieran los servicios del motor de datos ésta recurrirá al mismo.

Adicionalmente, ADO .NET emplea el lenguaje de marcación extendida XML como formato interno para almacenar o trasladar la información del mismo, o a través de los integrantes de la solución, lo cual hace que los resultados puedan ser fácilmente consultados a través de Internet por diferentes sistemas operativos o lenguajes.

## 4.1 ADO y ADO .NET

Todos los lenguajes utilizan alguna tecnología para acceder a los datos. Si el mismo tiene características de orientación a objetos, entonces las funcionalidades se proveerán a través de clases con métodos en vez de bibliotecas con funciones. A grandes rasgos, la forma por la cual una aplicación accede y manipula datos no varía en mucho de un lenguaje a otro, e incluso de un sistema operativo a otro. No obstante, la complejidad y sintaxis viene dada por cada tecnología.

A medida que han avanzado los años, las técnicas para acceso a datos se han tomado más simples y funcionales. En los últimos años la tecnología de acceso a bases de datos que más se ha utilizado de Microsoft ha sido ADO la cual esta diseñada para hacer acceso conectado a orígenes de datos, aunque es posible utilizar cursores desconectados, el diseño original del conjunto de componentes estaba pensado para trabajar en una forma conectada a la base de datos. De alguna manera esta directamente vinculada con el modelo físico de los datos, dado que en general los cursores que se utilizan se apoyan en la estructura de una o más tablas combinadas obtenidas por consultas a las bases de datos. Básicamente el componente más importante en ADO es el RecordSet el cual es un contenedor de datos y esta formado internamente como una tabla que contiene todos los datos necesarios para trabajar.

Ese conjunto de datos se puede obtener de más de una tabla origen pero esta formado o se estructura a través de sentencias de SQL (Lenguaje Estructurado de Consultas) que establecen relaciones entre las tablas a través de mecanismos de unión. Una vez obtenido el conjunto de resultados ese conjunto esta aplanado en el sentido que no se conocen las relaciones que se tienen entre ellos y salvo recorrerlos uno a uno, no se puede manipular información jerárquica.

Todos los tipos de datos que utiliza ADO en general están directamente relacionados con los tipos COM. Por lo tanto, si hay otros tipos de datos muy particulares en una base de datos deben compatibilizarse o sea hacer una conversión para hacerlos estructurados como

COM y de ahí que los mecanismos de accesos a ciertas bases de datos se pongan mas lentos.

Un problema de ADO es poder atravesar con un recordset los firewall (murallas de fuego) dado que esto se hace a través de mecanismos y abriendo puertos, cosa que habitualmente un firewall no permite. La transmisión de datos binarios a través de los mismos firewall se hace muy compleja dado que estos en general no permiten trasladar información desde un lado a otro bajo este formato.

Por otra parte ADO .NET fue pensado como un acceso a datos desconectado absolutamente de la base de datos. En principio en ADO .NET se puede modelar la estructura del conjunto de datos desde el código, sin necesidad de acceder a la base de datos. Esto significa que se puede definir en código una estructura de datos sin que tenga relación alguna con ninguna tabla en particular dentro de una o mas bases de datos. Se puede dibujar en memoria a través de código un conjunto de datos que podría funcionar como un repositorio de información.

En el caso de ADO .NET el DataSet es el conjunto común de datos que reemplazaría al recordset La diferencia es que el DataSet puede contener múltiples tablas en su interior, esto quiere decir que un DataSet puede contener datos de una tabla de una base de datos mas datos de otra tabla distinta de esa base u otra y asi sucesivamente no necesariamente se deben de obtener todas de una sentencia de SQL. Además se puede establecer en el conjunto del DataSet relaciones y por lo tanto navegar a través de esas mismas relaciones desde un conjunto de datos hacia el otro

Todos los tipos de datos que maneja el DataSet están directamente vinculados con la especificación que determina en topología de datos la definición de XML, lo cual lo hace mucho más portable tanto es asi que entonces no es necesario hacer conversiones de tipos. Entonces es mucho más sencillo poder pasar información inclusive entre distintas plataformas. XML al igual que HTML se considera texto plano desde el punto de vista de transmisión de la señal en la red con lo cual puede pasar mucho mas fácilmente los firewall.

El mismo principio que se esta utilizando para toda la definición del Microsoft .net framework

### 4.2 Propósito de ADO .NET

ADO .NET plantea una evolución natural a todos los modelos anteriores a él, ya que se basa en la manipulación de información obtenida de orígenes de datos en forma desconectada. La idea principal de ADO .NET es la de emplear las conexiones únicamente bajo demanda; esto quiere decir que las aplicaciones estarán por defecto desconectadas del origen de datos, y solamente en el momento de requerir los servicios del mismo se establecerá una sesión a éste. Esta aproximación permite al motor de datos resguardar un gran número de recursos.

Existen dos tipos de ejecución factibles de ser realizados sobre un origen de datos:

- Acciones unidireccionales.
- Acciones bidireccionales.

El primer caso involucra aquellas acciones que no retornan filas, como una actualización, modificación o eliminación de uno o varios registros. Para este caso la resolución en forma desconectada es relativamente sencilla, ya que en el momento de requerir la acción, el mismo podrá establecer una conexión con el origen, enviar la sentencia SQL y posteriormente desconectarse del mismo.

El segundo caso involucra aquellas acciones que dependen de ambos lados, como aquellas que retornan filas al ordenador. Evidentemente, la solución de este último bajo el modelo desconectado requiere de una mayor complejidad. Al igual que en el caso anterior, se abre la conexión con el origen. Para lograr esto último, ADO .NET cuenta con un conjunto de objetos que sirven de buffer ó bolsa para guardar en el cliente las filas obtenidas de la consulta del servidor (ver Figura 4.1). Como consecuencia, las acciones realizadas sobre las

filas (modificaciones, eliminaciones, etc.) serán gestionadas en forma local, o lo que es igual, utilizando un juego privado de datos.

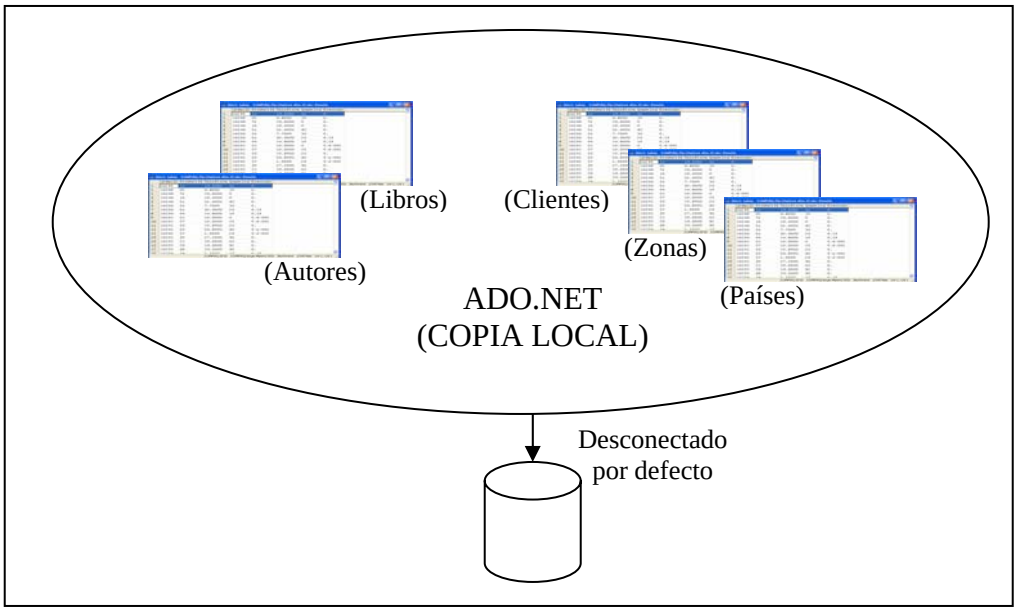



Figura 4.1. Modelo Desconectado de ADO.NET

En ADO ya se contaba con una característica similar denominada Cursores Desconectados, pero el proceso de conexión y desconexión debía realizarse en forma explícita. En ADO .NET los objetos están naturalmente desconectados del origen, y son estos últimos los que permiten en forma local el filtrar, ordenar y hasta agregar, eliminar o modificar a una o varias filas. Las acciones serán realizadas sobre la copia privada, sin que ello afecte a la información guardada en el motor de datos. ADO .NET cuenta con métodos que permiten posteriormente sincronizar la información del cursor local con el origen, y de esa forma hacer efectivos los cambios.

No obstante, hay dos puntos que se deben tener en cuenta cuando se emplea este modelo:

- Posibles conflictos
- Manejo de restricciones y relaciones en forma local

El primer punto hace referencia a los posibles conflictos que pueden ocurrir, como por ejemplo si dos usuarios modifican la misma fila utilizando su copia privada, o si uno de

ellos la elimina y el otro la modifica. En ambas situaciones los conflictos serían detectados en el momento de hacer efectivo los cambios, es decir, al sincronizar la información local con el origen.

El segundo punto a tener en cuenta tiene relación con las restricciones y vínculos establecidos entre las diferentes tablas. En muchas ocasiones es necesaria la existencia de una determinada fila en una tabla para la inserción de otra. A diferencia de ADO, ADO .NET cuenta con la posibilidad de copiar las relaciones y restricciones entre las diferentes tabla, a los efectos de que estas sean aplicadas también a la copia local. Des esta forma se obtiene el mismo comportamiento que en la modalidad conectada, pero en el juego de datos privado. ADO .NET maneja también un pool de conexiones (conjunto limitado de conexiones que se reutilizan constantemente para dar servicio a los diferentes clientes) que permite optimizar los recursos de apertura y cierre de conexiones. De esta forma, cada vez que una conexión es abierta, en realidad se utiliza una del pool, en vez de pasar por el costoso proceso de crear una nueva.

Adicionalmente, toda la información gestionada por ADO .NET puede ser almacenada en formato de documentos XML, ya sean cursores, reglas y hasta los tipos de datos de cada columna (campos). Esto beneficia al modelo, ya que este tipo de documentos es aceptado ampliamente para transferencia de datos a través de diferentes redes, y cuenta con la aprobación del consorcio W3. A su vez, el mismo es intensamente utilizado para intercomunicar información entre aplicaciones o sistemas operativos, dado que el mismo esta conformado exclusivamente por caracteres ASCII. Gracias a esto, los documentos pueden ser enviados a través del Web sin los inconvenientes que provocaban los formatos binarios como los que ofrecía COM. También puede ser consumido o generado por cualquier lenguaje en forma manual, o mediante algún modelo de objetos.

Otra característica radica en los datos soportados. En las tecnologías previas el traslado de la información se realizaba mediante la utilización de los estándares COM. Debido a ello, los tipos de datos retornados por el motor debían ser ajustados a alguno de los ofrecidos por el mismo. En la plataforma .NET los tipos de datos pueden ser manipulados en su formato

original, lo que permite al motor ofrecer más tipos y a la aplicación consumir éstos empleando a su tipo natural.

### 4.3 Las Clases de ADO .NET

ADO .NET incluye varias clases para el tratamiento de datos, las cuales están incluidas en varios espacios de nombres (ver Figura 4.2).

Debajo de Data se encuentran todas las clases de ADO .NET. Dentro del espacio OleDb se almacenan principalmente aquellas vinculadas a la obtención de información de un proveedor de datos genérico. Se entiende por genérico cualquier origen que requiera la utilización de un manejador (Driver) OleDB compatible.

Con el fin de optimizar el acceso a SQL Server y brindar funcionalidades específicas del mismo, Microsoft ofrece también un espacio llamado SqlClient, el cual almacena clases optimizadas para la gestión de conexiones con dicho motor. En un futuro, los diferentes proveedores ofrecerán su propio espacio de nombres conteniendo clases específicas, a los efectos de eliminar la utilización de OleDB, y así ofrecer un mejor rendimiento.

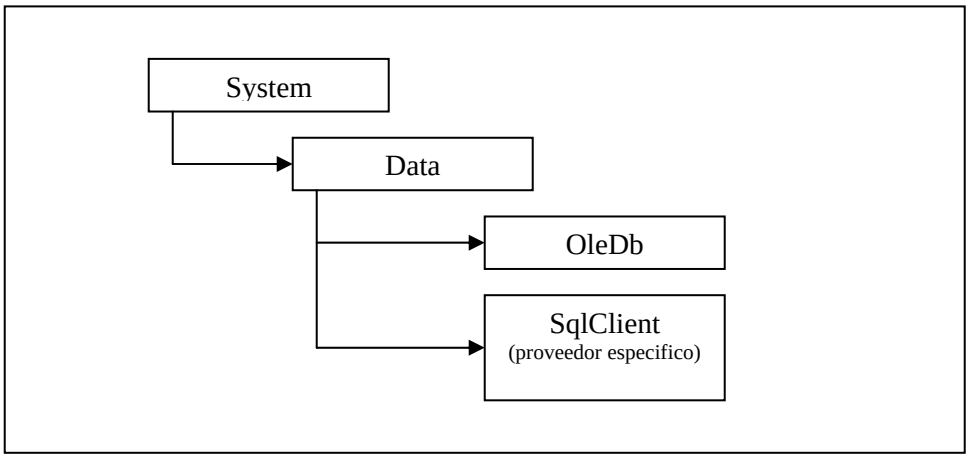



Figura 4.2. Espacios de nombres de ADO.NET

Las clases de ADO .NET se separan en dos grandes grupos, teniendo en cuenta la tarea que las mismas desempeñan en el modelo:

- Consumidores de datos.
- Gestores de información.

Dentro del primer grupo se encuentran las clases que se encargan exclusivamente de realizar una conexión y obtener los datos. Como ejemplo están las clases localizadas dentro de OleDb y SQLClient.

Dentro del segundo grupo se hallan aquellas que gestionan la información una vez obtenida. Esto permite a ADO .NET emplear las mismas clases para gestionar los datos, sin importar si éstos fueron obtenidos a través de un proveedor específico (SQLClient) o genérico (OleDb). De acuerdo con ello, quienes obtienen la información podrán estar relacionados con un motor de datos específico, pero quienes la gestionan posteriormente serán comunes a todas ellas.

### 4.3.1 Proveedores de Datos y Objetos Conectados

Se le llama proveedor a toda aquella tecnología que cumpla la función de establecer una conexión a un origen o fuente de datos.

Existen realmente pocas clases en ADO .NET que pueden interactuar en forma constante con una conexión abierta, y ello es debido a que la mayor parte de las funcionalidades del modelo están relacionadas con la conexión bajo demanda.

Las clases precedidas por OleDB ó Sql, dependiendo de que espacio de nombres se este utilizando que se encuentran dentro de este grupo son:

| NOMBRE | FUNCIÓN |
|---|---|
| Connection | Es la clase encargada de contener la información y establecer la sesión a un origen de datos. Para ello se utiliza una cadena de conexión, y el formato de la misma dependerá de cada motor, por lo que generalmente se tendrá que consultar la |

| | |
|---|---|
| | documentación del mismo para saber cómo construir la cadena. |
| Command | Esta clase representa una consulta o procedimiento almacenado a ser enviado a un origen de datos. Para que esta clase pueda realizar su función es necesarios que este asociada a una conexión (Connection). |

### 4.3.2 Proveedores de Datos y Objetos Desconectados

En versiones anteriores a .NET, la transferencia de información, ya sea entre un origen de datos y una aplicación como entre componentes, se llevaba acabo mediante un objeto llamado RecordSet, el cual esencialmente representaba un conjunto de filas y columnas. Cada columna contenía el nombre y tipo de campo, mientras que cada fila almacenaba la información propia de cada registro. El RecordSet podía contener un conjunto de filas de un origen. Cada cursor o RecordSet era siempre el resultante de la ejecución de un comando. En el caso de consultas que realizaran cruce entre tablas, el conjunto de filas consiguientes eran representadas como un único cursor que contenía a los resultados de ambas tablas. De esta forma, los mismos se ofrecían siempre como una única tabla, o como un conjunto de tablas no relacionadas entre sí (ver Figura 4.3).

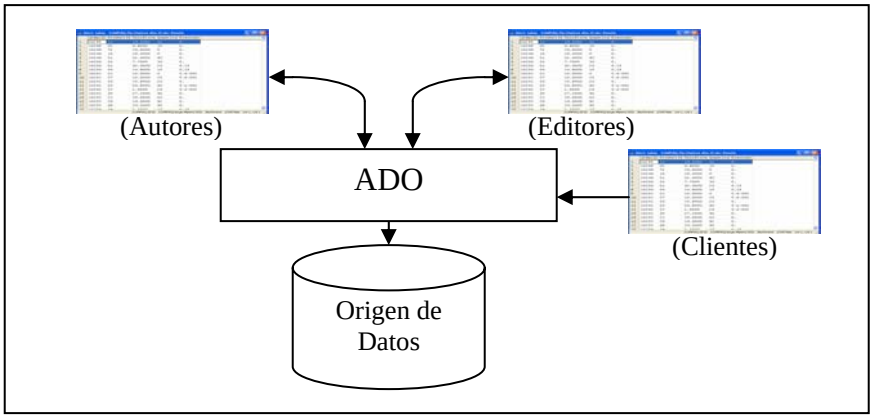



Figura 4.3. Funcionamiento del RecordSet

En ADO.NET el objeto RecordSet no existe, pero en su reemplazo se ofrece uno llamado DataSet. El mismo es el que se utiliza para gestionar los cursores en ADO.NET, así como también para el intercambio de datos entre aplicaciones o componentes. El DataSet permite almacenar filas de uno o varios resultados, pero, a diferencia del primero, en éste es posible establecer las relaciones entre sus integrantes. Sin embargo, la diferencia más importante entre un RecordSet y un DataSet es que este último representa una vista local o en memoria de las filas obtenidas de un origen de datos.

Un objeto DataSet obtiene un conjunto de filas de una o más tablas, y posteriormente las aloja en memoria. Una vez realizado este proceso, el mismo procede a desconectarse del origen de datos. El DataSet se encarga posteriormente de la gestión local de las filas obtenidas, así como también de las posibles acciones a ser realizadas sobre las mismas (modificaciones, eliminaciones, etc.). Todas las acciones serán entonces aplicadas sobre la copia privada o local de datos, sin que ello interfiera en el origen. Posteriormente, se necesitará sincronizar la información con este último, a los efectos que los cambios puedan hacerse efectivos.

Para esta tarea, se involucra un segundo objeto llamado DataAdapter, el cual funciona de intermediario o puente sincronizador entre la copia privada y el origen de datos (ver Figura 4.4).

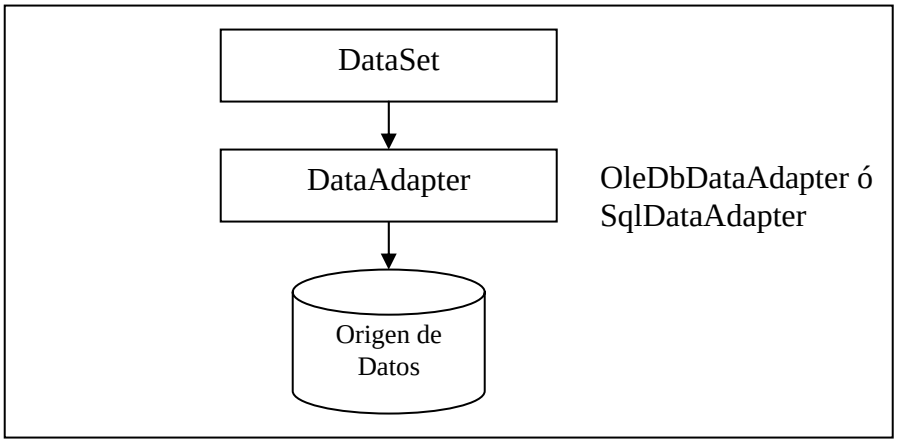



Figura 4.4. Funcionamiento del DataSet

Un DataAdapter establece una conexión, y luego envía al origen de datos una acción SQL por cada registro modificado, agregado o eliminado en la copia local, y luego se desconecta. Esta tarea no necesariamente debe realizarse al finalizar cada unas de las acciones, sino que la misma puede ser llevada a cabo durante periodos determinados de tiempo. El DataSet puede también desechar la copia local y obtener un nuevo juego de datos más actualizado.

El DataSet beneficia en forma directa a las aplicaciones Web (ASP.NET), las cuales son desconectadas por naturaleza. En general, no se hace necesario el conocer dónde está localizado el origen de datos hasta el momento en que la información debe ser sincronizada. En muchas ocasiones, el proceso involucra solamente lectura de datos o acciones que nunca serán enviadas al motor, por lo que el DataSet será utilizado simplemente como estructura de intercambio entre componentes o aplicaciones.

Dado que los cursores son internamente gestionados en formato XML, el proceso de traslado de información entre componentes de una red puede ser realizado haciendo uso de los protocolos estándares de comunicación, sin que se requieran consideraciones especiales de seguridad.

# CAPITULO 5

# SEGURIDAD EN APLICACIONES ASP.NET

## 5.1 Nociones de Seguridad

De manera predeterminada, la mayoría de los sitios Web permite el acceso anónimo. Esto quiere decir que cualquiera que tenga conexión a Internet puede acceder y ver las páginas de su sitio. Estos usuarios no necesitan ser autenticados, o validar su identidad; pueden acceder a cualquier archivo que esté disponible en su servidor. La seguridad en Web está diseñada para restringir el acceso a determinados archivos y hacerlo posible sólo a cierto grupo de usuarios.

Este proceso demuestra los elementos básicos de una transacción de seguridad Web. El primer paso es la autenticación, que es el proceso de identificar al usuario que requiere información. El usuario se identifica mediante sus credenciales, que pueden venir de muchas maneras (casi siempre un nombre de usuario y una contraseña). La autenticación asegura que una persona es quien dice ser. Si el sistema de seguridad no puede identificar a un usuario a partir de sus credenciales, la autenticación falla y no permite el acceso al usuario desconocido. Si las credenciales son correctas, se le permite entrar al sistema y se le da una identidad correcta y conocida.

Una vez que el usuario se le da la identidad, el sistema determina a cuáles recursos tiene acceso. Este proceso se le conoce como autorización. El sistema autorizará de acuerdo con los permisos que se le asignaron a su identidad. En Web, ciertos usuarios pueden tener acceso a ciertos archivos, pero otros usuarios tendrán acceso a otros.

Finalmente, el último paso es la personificación. Como una medida adicional de seguridad se añadió esta característica (ver Figura 5.1).

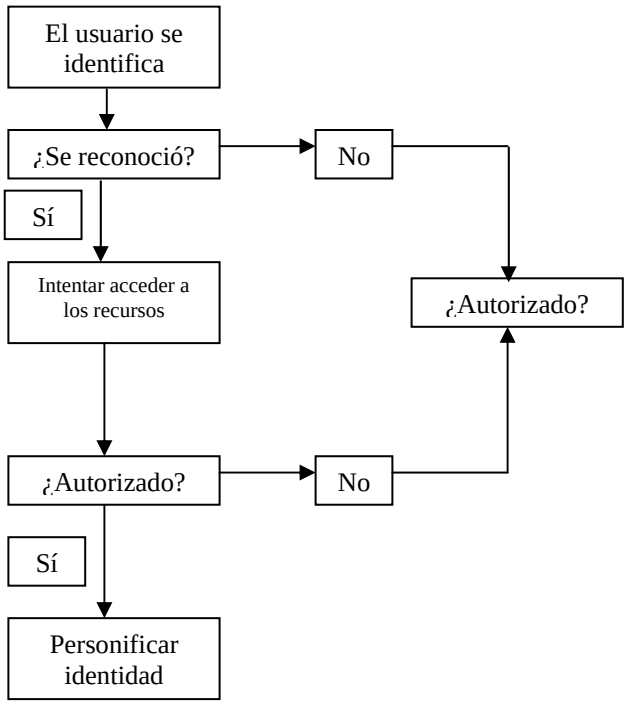



Figura 5.1 El protocolo típico de seguridad

Básicamente, la seguridad en ASP .NET se establece con dos métodos distintos. Puede autenticar y autorizar credenciales comparándolas con las identidades del sistema operativo (mediante IIS), o puede compararlas con permisos en un origen de datos (como se hace mediante web.config). El primer método requiere muy poco código o modificación de las páginas ASP.NET, pero proporciona menos control sobre la autenticación. El segundo, por el contrario, requiere de mayor codificación, pero da mayor flexibilidad.

El sistema operativo Windows admite la seguridad basada en roles. Un rol define un tipo de identidad. Por ejemplo, si está a cargo de mantener un sistema de cómputo e instalar software y hardware, está en el rol o papel del administrador. Si alguien sólo quiere hacer una atarea de la escuela o navegar por Internet, está en el papel de invitado (esto es, un invitado a los recursos de la computadora). Los roles definen cuánto control tiene alguien sobre la computadora y cuáles permisos están disponibles para dicha persona.

Por lo general, los roles tienen varias identidades asociadas. Varias personas pueden dar mantenimiento a una computadora, lo cual quiere decir que existen varios administradores.

En Windows, estas identidades son conocidas como usuario, la cuenta aspnet_wp es la identidad predeterminada que utiliza un cliente cuando accede a una máquina mediante el servidor Web local. Es una cuenta anónima, lo que significa que no requiere ninguna contraseña para acceder a su máquina cuando utiliza esta identidad. Como tal, los permisos que se le proporcionan a esta identidad son mínimos.

## 5.2 Autenticación

La autenticación en ASP .NET se establece mediante Proveedores de Autenticación: módulos que contienen el código para autenticar solicitudes de clientes Web. Estos proveedores se controlan mediante el archivo web.config, con las etiquetas authentication.

ASP .NET ofrece métodos, o modos, para autenticar a los usuarios, cada uno de ellos implementando mediante un proveedor de autenticación distinto: Windows, Passport y Formularios. La autenticación Windows se hace con IIS, lo que requiere de poca o ninguna modificación a sus páginas. Las autenticaciones Passport y Formularios son similares: la segunda se implementa en el servidor; la primera mediante un servicio de suscripción de Microsoft.

### 5.2.1 Autenticación Windows

Cuando los clientes hacen solicitudes a las páginas ASP.NET, lo primero que encuentran en el programa servidor Web de Microsoft, Internet Information Server o IIS. En este punto, IIS puede autenticar al usuario, o delegar las tareas de autenticación a la aplicación ASP .NET. Cuando IIS maneja la autenticación, puede comunicarse directamente con el sistema operativo (Windows NT o 2000) para validar las credenciales del usuario (ver Figura 5.2.).

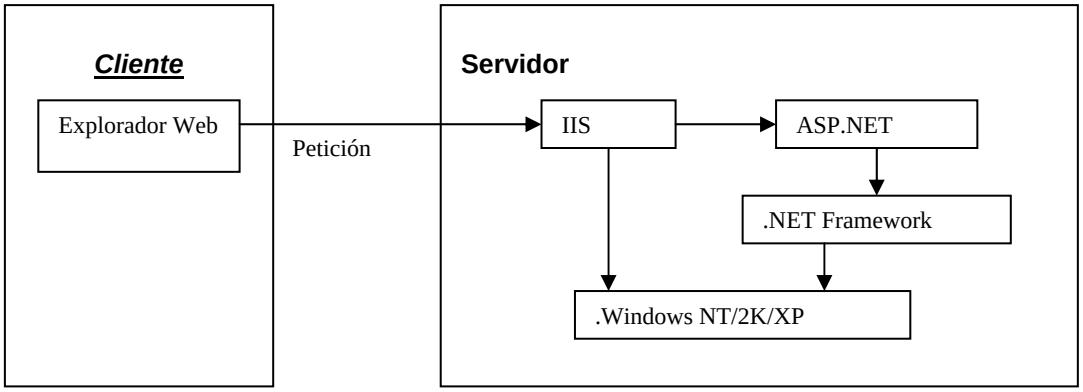



Figura 5.2 La arquitectura de seguridad de ASP.NET

IIS no requiere autenticación de manera obligatoria. Puede dejar a los usuarios a la aplicación sin autenticarlos, y confiar en las páginas ASP.NET para establecer cualquier autenticación. Cualquier usuario al que IIS deje pasar sin una identificación correcta se conoce como usuario anónimo.

Las identidades que IIS asigna son definidas por cuentas de usuario de Windows, de aquí el nombre autenticación Windows. Los clientes Web son restringidos de la misma forma que los usuarios y roles de Windows. Por ejemplo, si un cliente Web accede a un rol administrativo, tiene permisos para modificar el sistema de archivos de la manera que desee, incluyendo borrar o mover de lugar los archivos.

IIS tiene tres tipos diferentes de autenticación Windows: Básica, Resumen y NTLM (también conocida como Autenticación Integrada de Windows).

La autenticación básica es la forma más sencilla de autenticación y la más cercana a no hacer nada. Es una forma estándar de la industria para pedir las credenciales del usuario, e incluso es parte de la especificación de HTTP del Consorcio de Word Wide Web. El proceso de la autenticación básica es:

1. El cliente solicita un recurso restringido al servidor.
2. El servidor Web responde con “401 Unauthorized”.
3. El explorador Web recibe este mensaje y le pide al usuario su identificación, normalmente un nombre y una contraseña.

4. El explorador Web intenta entonces acceder a los recursos del servidor con estas credenciales.
5. Si las credenciales fallan, el proceso vuelve a iniciar en el paso número 2.
6. Cuando la autenticación tiene éxito, el explorador Web ya tiene acceso a los recursos solicitados.

Con la autenticación básica, el nombre de usuario y la contraseña proporcionadas viajan por la red hacia el servidor como texto normal lo cual es poco seguro.

La autenticación resumida trabaja de manera a la básica, pero el nombre de usuario y la contraseña se cifran antes de enviarse por la red. Este mecanismo de cifrado, conocido como hashing, altera la información de forma que no es posible descifrarla. Es un proceso en un sólo sentido:

1. El cliente solicita un recurso restringido del servidor, y éste envía una solicitud de autenticación.
2. El servidor Web responde con un “401 Unauthorized”.
3. El explorador Web recibe este mensaje y pide la identificación del usuario. A continuación, añade cierta información única a los datos suministrados y los cifra. Esta información única asegura que nadie podrá copiar la información cifrada y tener acceso después.
4. El explorador web envía la identificación cifrada y una copia adicional de la información única, sin cifrar, al servidor. El servidor utiliza entonces dicha información adicional para cifrar su propia copia de la identificación (es decir, la identificación que se almacena en la información de las cuentas de usuario en Windows).
5. El servidor compara la identificación recién cifrada con los valores enviados por el explorador Web.
6. Si la identificación falla –esto es, las dos versiones cifradas no coinciden- el proceso comienza de nuevo en el paso 3.

7. Cuando la autenticación se realiza sin problemas, el explorador Web puede acceder a lo recursos.

La autenticación NTLM, el usuario nunca verá una ventana que le pida su autenticación. En lugar de esto, en cuanto el explorador Web hace contacto con el servidor, envía el nombre de usuario y la contraseña cifrados que el usuario utilizó para entrar en su computadora. El servidor evalúa esta información para determinar si el usuario debe ser autorizado a pasar. Todo esto es imperceptible para usuario.

Normalmente, la autenticación de Windows se utiliza cuando se desea un método fácil de implementar que no requiera nada de código ASP .NET. Todo lo que tiene que hacer es establecer el valor de ciertas propiedades en el IIS.

Tanto la autenticación resumida como  la de Windows requieren que sus usuarios estén ejecutando Internet Explorer. Asimismo, el modo resumido sólo funciona en cierto tipo de servidores.

### 5.2.2 Autenticación de Formularios

Con ASP .NET, se puede optar por autenticar no mediante IIS, sino mediante la aplicación ASP .NET con ayuda de un proceso conocido como autenticación de formularios. Esto da mayor control sobre las distintas posibilidades de autenticación de un sitio. Por ejemplo, se puede guardar la autenticación de usuario en una base de datos o un archivo XML en lugar de hacerlo mediante Windows.

Con este método, los usuarios son dirigidos a un formulario de inicio de sesión del sitio Web, donde deben introducir su identificación. Si la aplicación la acepta, a partir del esquema construido, ASP .NET creará una cookie de autorización en la computadora del explorador Web. Está cookie contiene ya asea la identificación, de cierta manera, o una cadena que puede utilizarse para adquirirla. Esta cookie se utiliza entonces en toda la aplicación para las autorizaciones. Este proceso es (ver Figura 5.3):

1. Un cliente solicita una página segura de su sitio.
2. Si la solicitud no contiene una cookie de autenticación correcta, el servidor Web redirige al usuario hacia el URL que se especifica en el atributo loginURL de la etiqueta Authentication en el archivo web.config. Este URL deberá contener un formulario de inicio de sesión para el usuario.
3. La identificación se introduce en el formulario y se envía.
4. Si la identificación es correcta, ASP .NET crea una cookie de autenticación en el cliente.
5. El usuario puede entonces ser restringido a la página que solicitó originalmente.

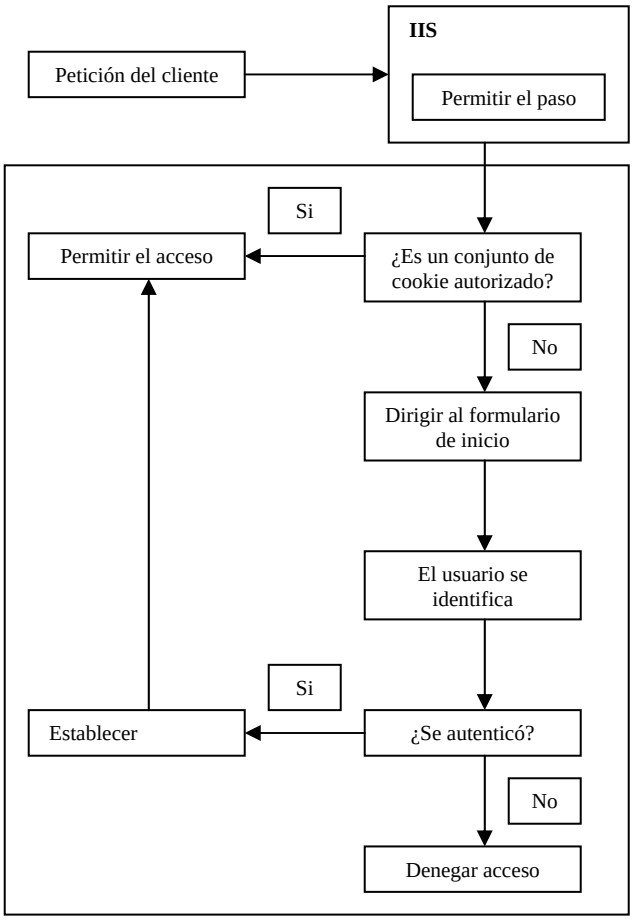



Figura 5.3. El flujo de trabajo de la autenticación de formularios

Una vez que se establece la cookie de autorización, todos los requerimientos subsiguientes serán autenticados de manera automática hasta que el usuario cierre el explorador Web o la sesión termine.

La autenticación de formularios es muy poderosa y ofrece a los desarrolladores muchísima flexibilidad al integrar la autenticación. La configuración se encuentra en el archivo web.config, este método de autenticación es fácil de implementar. Todo lo que se necesita es construir una página de inicio de sesión.

### 5.2.3 Autenticación con Microsoft Passport

La autenticación de Passport es un servicio centralizado de autenticación (no un servicio Web) proporcionado por Microsoft. Trabaja de manera muy similar a la autenticación de formularios, ambos crean cookies de autenticación que residen en el cliente, y se utilizan para la autorización. Cuando un usuario encuentra la autenticación de Passport en un sitio, es redirigido a la página de ingreso de Passport (http://www.pasport.com) que proporciona un formulario muy sencillo que llenar. Este formulario verifica la identificación del usuario contra el servicio Passport de Microsoft y determina si el usuario es reconocido. Si lo es, establece una cookie de autenticación, igual que la autenticación de formularios. Los pasos exactos son:

1. Un cliente solicita una página segura de su sitio.
2. Si la solicitud no contiene una cookie Passport de autenticación adecuada, su servidor Web redirigirá al usuario al servicio de inicio de sesión de Passport. El servidor Web también pasa cierta información cifrada sobre la solicitud al servicio Passport.
3. Se presenta al usuario un formulario de entrada en el sitio de inicio de sesión de Passport. Se introduce la identificación y se envía mediante un POST de formulario (utilizando el cifrado SSL).
4. Si se hace la identificación, el servicio Passport crea un boleto de autenticación y lo coloca en la cadena de consulta. Se vuelve a enviar al usuario al sitio original.

5. Cuando el servidor ve que hay un boleto de autenticación en la cadena de consulta, crea la cookie de autenticación para el usuario.

De este modo, el servicio de autenticación de Passport no crea por sí mismo la cookie de autenticación. Esto se deja al servidor Web de origen.

El servicio de Passport es una manera útil y sencilla para implementar en su sitio un esquema de seguridad en el cual puede confiar. Muchos sitios Web existentes, entre los que se encuentran casi todos los sitios de Microsoft, utilizan el servicio Passport. Después que un usuario tiene una identidad Passport adecuada, puede entrar a todos los sitios Passport con el mismo nombre de usuario y contraseña.

### 5.3 Autorización

El propósito de la autorización es determinar a cuáles recursos puede acceder un usuario autenticado. La autorización en ASP .NET trabaja de dos maneras. Puede basarse en que Windows le diga a cuáles recursos tiene acceso un usuario autenticado, lo cual se conoce como autorización de archivos, o puede basarse en el URL del sitio solicitado, lo que se conoce como autorización de URL.

Windows puede establecer opciones de seguridad para cada archivo y carpeta de una computadora. Estas opciones incluyen permitir a los usuarios leer o escribir archivos, crear aplicaciones y ver el contenido de carpetas. Estos permisos de archivo y de carpeta se guardan en Listas de Control de Acceso (ACL), en el sistema operativo Windows. Es fácil modificar dichas listas si se marca o desmarca las casillas de verificación para cada usuario de la ficha de Seguridad.

Con la autorización de archivos, ASP .NET se comunica con Windows para correlacionar las identidades de los usuarios con las ACLs. Cuando un usuario autenticado trata de acceder a un archivo en su servidor Web, Windows verifica el ACL para determinar si

dicho rol o identidad tiene permiso para ver este archivo. La autorización de archivos trabaja con la personificación para determinar los permisos ACL.

A veces las ACLs pueden ser difíciles de manejar. Si se tiene un sitio Web con 50 carpetas diferentes, cada una con varios archivos, definir las ACLs puede ser dificil. ASP .NET también permite la autorización de URL, que orienta las identidades de los usuarios especificadas a los diferentes URLs solicitados.

Existen dos identidades especiales en ASP .NET que permiten otro grado de agrupación de usuarios. Un signo de interrogación en el atributo users significa usuarios anónimos, aquellos que no han sido autenticados de alguna manera. Un asterisco indica a todos los usuarios, independientemente de su estado de autenticación.

Aunque este sistema de configuración jerárquica es muy útil, no proporciona mucho control detallado. Por ejemplo, los valores establecidos en web.config se aplicarán a la carpeta en la que reside este archivo al igual que a todas las carpetas que estén bajo ella. Si se desea cambiar los parámetros de seguridad, se necesita construir nuevos archivos web.config en cada carpeta que los necesite.

El archivo web.config permite controlar, con mayor granularidad y mediante la etiqueta location, los recursos a los que los usuarios están autorizados a acceder. Esta etiqueta puede ser implementada en cualquier archivo web.config y establece cuáles recursos están restringidos de acuerdo con la estructura de carpetas del sitio.

**5.4 Personificación**

La personificación le permite a ASP .NET ejecutar páginas con la identidad del cliente para el cual está operando. De manera predeterminada, la personificación está inhabilitada. A medida que el usuario va de la autenticación de IIS a la aplicación ASP.NET, el propio ASP .NET toma la identidad que IIS está configurando para utilizar (de manera predeterminada es la identidad “Máquina Local”). Por lo general, esta identidad puede

acceder a todos los archivos y carpetas. Deben utilizarse otras medidas de seguridad para manejar el acceso, como la autorización por URL (ver Figura 5.4).

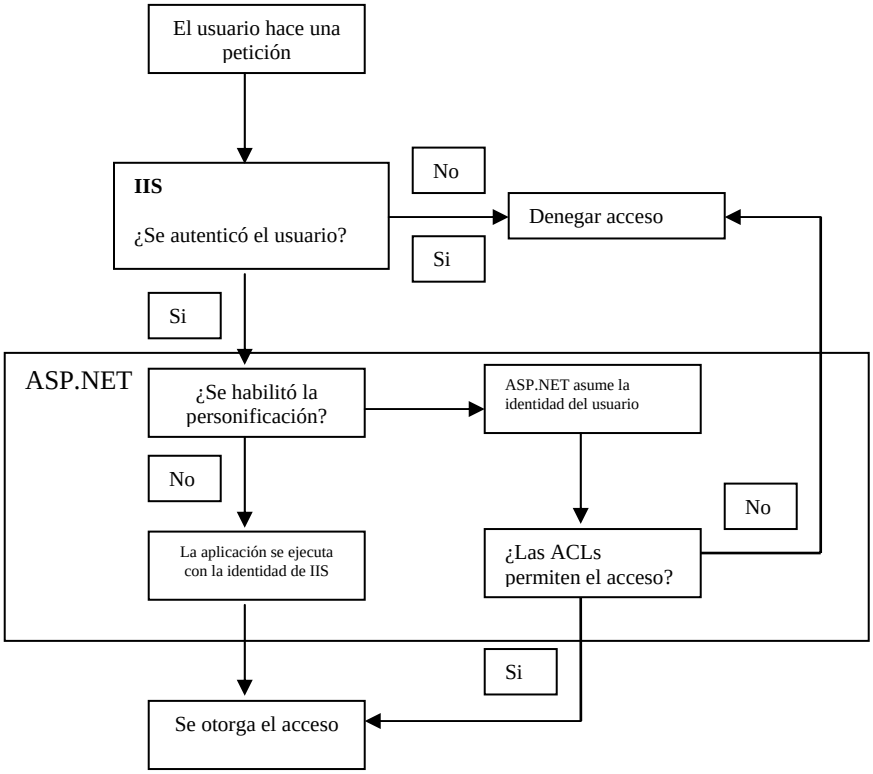



Figura 5.4 El flujo de proceso cuando se habilita la personificación

Cuando se habilita la personificación, ASP .NET toma el rol de la identidad que IIS le pasa. Si el usuario no está autenticado, ASP .NET personificará al usuario anónimo, y si el usuario está autenticado, ASP .NET tomará su identidad. Ahora que ASP .NET está personificando a otro usuario, Windows puede restringir el acceso a la aplicación como un todo utilizando ACLs.

Una aplicación ASP .NET es un usuario de los recursos del sistema. Accede a archivos y carpetas, memoria y demás. De manera predeterminada, la aplicación ASP .NET tiene permisos bastante liberales, y generalmente puede acceder a todo lo que el sistema puede ofrecer. Esto es necesario porque ASP .NET tiene que utilizar estos recursos para operar de manera correcta.

No obstante, se puede restringir el acceso a ciertos recursos, de acuerdo con la persona que esté utilizando la aplicación ASP .NET. Por ejemplo, un usuario anónimo no debería ser capaz de tomar ventaja de los permisos de la aplicación y acceder a todos los recursos del sistema. De este modo, una aplicación ASP .NET puede personificar a sus usuarios para poder restringir el acceso.

Cuando ASP .NET personifica a un usuario, actúa como si fuera él, en todo sentido. De hecho, el sistema operativo creerá que ASP .NET es precisamente ese usuario en particular que accede a los recursos. Las ACLs ahora pueden aplicarse y restringir el acceso tanto a ASP .NET como al usuario.

La personificación impide que los usuarios accedan a archivos para los cuales no tienen permiso. De manera predeterminada, ASP .NET asume que se utilizan otros métodos de autenticación y autorización, y la personificación no es necesaria. Por ejemplo, con la autenticación de Windows, se puede limitar a los usuarios para que accedieran a recursos incluso antes de que pudieran acceder a la página en cuestión. La personificación no es necesaria aquí porque el usuario nunca llega siquiera a intentar ejecutar el código. Pero si hay una posibilidad de que el usuario pueda acceder a un archivo que no debiera, la personificación es muy útil.

La personificación se utiliza si no se desea invertir tiempo en escribir código ASP .NET. Con la personificación solamente se necesita establecer la autenticación IIS y manejar las ACLs.

# CAPITULO 6

# MODELO DE APLICACIÓN DE TRES CAPAS

## 6.1 El Modelo de Tres Capas

La programación en múltiples capas es la técnica más efectiva en las aplicaciones, debido a la fácil administración que consiste en dividir los componentes de la aplicación en capas y la rapidez que esto implica en programas orientados a Cliente/Servidor.

El modelo de aplicación de tres capas, divide las aplicaciones en tres capas (en ocasiones, no muy distintas): una interfaz, una de lógica u objetos de negocios, y una de datos, programarlas por separado y luego unirlas ya sea en tiempo de ejecución o en el mismo código[14]. Este modelo es adecuado para desarrollar aplicaciones Web y no es muy difícil de implementar (ver Figura 6.1).

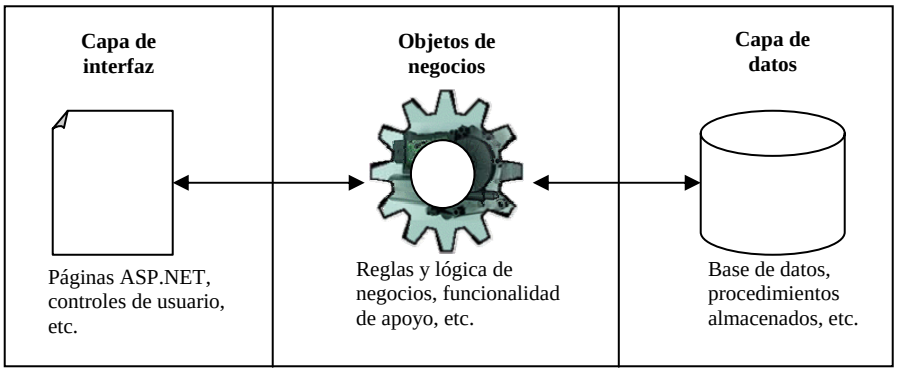



Figura 6.1. El modelo de tres capas

Es como una producción de teatro. La primera capa son los actores en escena. Éstos proporcionan la interfaz para la audiencia, que atrae su atención y los envuelve en la vivencia, etc. Ésta es la capa con la que se comunica la audiencia.

La segunda capa consta de las personas que dirigen y apuntan: la orquesta, los tramoyistas, los apuntadores, etc. Todas estas personas se comunican con los actores pero no son vistos directamente por la audiencia. Dirigen y dan material a los actores.

[14] Chris Payne, Aprendiendo ASP .NET en 21 dias, Edit. Prentice Hall, Pág. 510.

Finalmente, la tercera capa son las personas responsables de las escenas y el material: escritores, artistas, diseñadores de escenario, etc. Todas estas personas trabajan en conjunto para dar la sustancia a la producción. No son vistos por la audiencia excepto a través de los resultados que dan.

Este modelo de aprovechamiento de funciones está muy bien definido y sincronizado. La falta de una capa impediría el buen funcionamiento de la aplicación.

El mismo modelo se aplica en el desarrollo de aplicaciones Web. La ausenta de una capa dificulta la producción de una aplicación. En un sitio de comercio electrónico, la primera capa es la interfaz (formularios, carritos de supermercado, gráficos, etc.). La capa de negocios, que es la intermedia consta de la lógica para controlar los precios, costos de envíos, etc. La de datos que es la tercera, consta del inventario almacenado en una base de datos. Si cualquiera falta, otra deberá tomar su lugar.

### 6.1.1 Capa de Datos

La capa de datos representa el mecanismo por el cual se manipula y persiste la información. Consiste en un administrador de bases de datos relacional, y el esquema de datos propio de cada aplicación. Cuando hay varias aplicaciones presentes, los modelos de datos se complementan, evitando la duplicidad de información y aumentando las facilidades que brinda el sistema como un todo.

La capa de datos tiene como misión la administración de la información que maneja el sistema. Esto incluye el almacenamiento, la actualización y la consulta de todos los datos contenidos en el sistema. En la práctica, esta capa es esencialmente un servidor de bases de datos. Gracias a herramientas propias y de terceras partes para la abstracción de de base de datos, las aplicaciones pueden usar distintas bases de datos. También es posible agregar soporte para una nueva base de datos en un periodo de tiempo relativamente corto. La capa de datos puede estar en el mismo servidor que la lógica de negocio y presentación, o en un servidor independiente (incluso puede consistir en un cluster de servidores).

### 6.1.2 Capa de Lógica de Negocio

En la capa de lógica de negocio se modela el comportamiento del sistema, basándose en los datos provistos por la capa de datos, y actualizándolos según sea necesario. Esta capa describe los distintos procesos de negocio que tienen lugar en las organizaciones, desde el ciclo de aprobación de un documento hasta la política de descuentos para un pedido.

El comportamiento de la aplicación es definido por los componentes que modelan la lógica de negocio. Estos componentes reciben las acciones a realizar a través de la capa de presentación, y llevan acabo las tareas necesarias utilizando la capa de datos para manipular información del sistema. Tener la lógica de negocio separada del resto del sistema también permite una integración más sencilla y eficaz con sistemas externos, ya que la misma lógica utilizada por la capa de presentación puede ser accedida desde procesos automáticos que intercambian información con los mismos. En los sistemas, esta capa es construida utilizando tecnología de componentes.

### 6.1.3 Capa de Presentación

Esta capa contiene todos los elementos que constituyen la interfaz con el usuario. También, incluye todo aquello con lo que el usuario puede interactuar, como por ejemplo las pantallas de las aplicaciones, el modelo de navegación del sistema y los adaptadores para cada modo de acceso (browser, teléfono celular, etc.).

En una aplicación web, generalmente la capa de presentación se divide en dos: el lado servidor y el lado cliente. En el lado servidor ocurre toda la interacción con la lógica de negocio, y es también donde se genera la interfaz del usuario. En el lado cliente se presenta la interfaz generada en el servidor al usuario, de forma tal que éste pueda trabajar con ella. Los datos o acciones reunidas por el cliente son luego enviadas de vuelta al servidor para su procesamiento.

## 6.2 Los Componentes en el desarrollo de aplicaciones ASP .NET

Los componentes son objetos que pueden utilizarse una y otra vez en distintas aplicaciones. Usualmente representan objetos reales. Para fabricar un reloj toma varios componentes, como resortes, bisagras, cristal, madera y péndulos, y los ensambla. Las aplicaciones de ASP .NET son como cualquier otra: constan de muchas partes que se ensamblan para obtener una entidad completa.

Sin embargo, ahí una importante observación al respecto. El tener demasiados componentes podría contravenir el propósito debido a una complejidad innecesaria.

En términos de ASP .NET, los componentes son piezas de código reutilizables que pueden mejorar una aplicación o darle una nueva funcionalidad. Un componente se utiliza para describir un objeto, como un calendario o un libro. En una página ASP .NET, un cuadro de texto o un conjunto de datos de una base de datos podría considerarse como componentes.

Anteriormente se mencionaba la utilización de objetos de negocios como la capa intermedia entre las capas de presentación y datos. Estos objetos de negocios (o funcionales) son componentes que integran código apropiado para una aplicación. Es el código que se utiliza para dar funcionalidad sin una interfaz de usuario (También conocidas como lógica de negocios, reglas de negocios o lógica funcional). Por ende, los componentes que implementan una lógica de negocios se conocen como objetos de negocios.

Lo ideal es que la lógica esté separada de las páginas ASP .NET en un objeto de negocios. La página ASP.NET sólo debe usarse con fines interfaz y para el procesamiento propio del cliente de la aplicación.

En un sitio de comercio electrónico puede encontrarse un ejemplo común de un objeto de negocios, que tuviera que obtener los cargos de envío de distintas empresas acordes. Un desarrollador puede generar esta lógica en una página ASP .NET, pero seria difícil modificarla se los cálculos de envió cambian posteriormente (sin mencionar que no es muy

reutilizable). Una mejor opción seria generar un componente de envíos que pueda usarse en cualquier aplicación ASP .NET. Este obtendría la información de las bases de datos de las empresas de envíos y daría a la aplicación de comercio electrónico lo que necesitara. El componente podría utilizarse varias veces, y cualquier modificación se haría en un lugar en vez de hacerlo en varias páginas ASP .NET.

Una parte importante de los componentes es que se generan desde el inicio para los propósitos de cada desarrollo a diferencia de los de componentes como ADO .NET, XML, etc. Además de que oculta complejidad en el manejo de instrucciones, es decir, por ejemplo un componente de conexión de bases de datos no sólo va a permitir al usuario (un desarrollador) obtener los datos con una sola línea de código, sino que oculta toda la complejidad de trabajar con objetos de datos (ver Figura 6.2). El usuario no tiene que preocuparse por los comandos o las conexiones. No tiene que preocuparse de la captura de errores. El componente controla todo esto automáticamente.

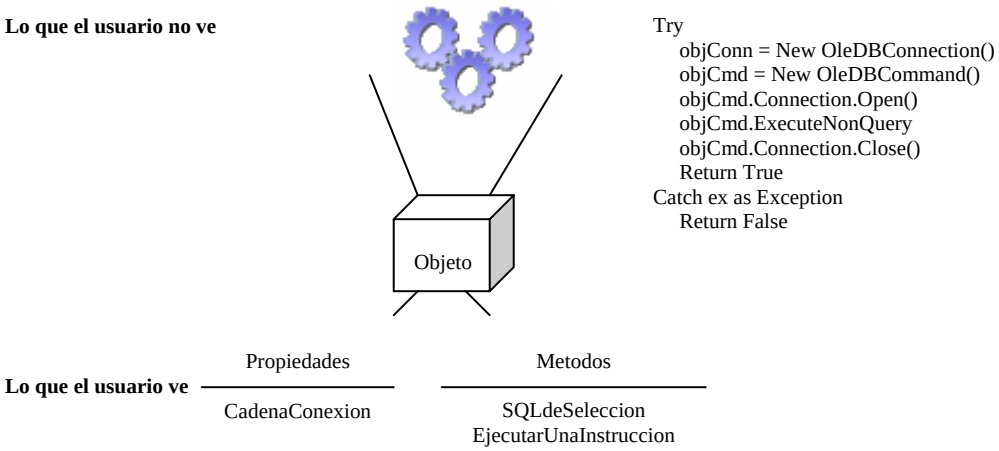



Figura 6.2 Ejemplo de un componente de Base de Datos

# CAPITULO 7

# CASO PRÁCTICO

## 7.1 Planteamiento del Problema

### 7.1.1 Antecedentes de los Laboratorios de Cómputo

El objetivo de los laboratorios de computo es hacer accesible el equipo de cómputo a los estudiantes de la UNAM sin importar su materia principal de estudio, para su uso libre, así como para cursos curriculares y extracurriculares. Esto con el fin de obtener el mayor aprovechamiento de la cultura informática en cada una de las disciplinas y fomentar la excelencia académica y profesional de la comunidad universitaria[15].

Los laboratorios de cómputo fueron creados pensando en satisfacer las necesidades de uso de equipo de cómputo para los alumnos y profesores de las dependencias que forman parte de la universidad.

Existen cinco laboratorios de cómputo actualmente, dos ubicados en la Facultad de Estudios Superiores Cuautitlán Campo 1 y tres ubicados en la Facultad de Estudios Superiores Cuautitlán Campo 4 como se muestra a continuación respectivamente:

- Laboratorio de Químico Biológicas (QB)
- Laboratorio de Diseño Gráfico (DG)
- Laboratorio de Ciencias Administrativas y Sociales (CAS)
- Laboratorio de Ingeniería Mecánica Eléctrica (IME)
- Laboratorio de Medicina Veterinaria y Zootecnista (MVZ)

Dentro del programa actividades de los laboratorios se encuentran las siguientes actividades[16]:

- Asignatura de iniciación al cómputo.
- Talleres de cómputo.

[15] Texto tomado del **Manual de procedimientos Centro de Cómputo UNAM** Pág. 1.
[16] Texto tomado del **Manual de procedimientos Centro de Cómputo UNAM** Pág. 2.

- Cursos sobre diversas herramientas de cómputo.
- Cursos especiales, entendiéndose estos, como aquellos que a solicitud de grupos de alumnos se abran sobre diferentes temas de herramientas de cómputo y software de aplicación.
- Cursos de capacitación y diplomados se consideraran dentro del programa y en tal sentido no se cobrará a la comunidad interna de la Facultad, excepción hecha a egresados y externos.
- Permitir a los alumnos el acceso controlado a los activos de los laboratorios, tales como computadoras y software de aplicación.

### 7.1.2 Estructura de los Laboratorios de Cómputo

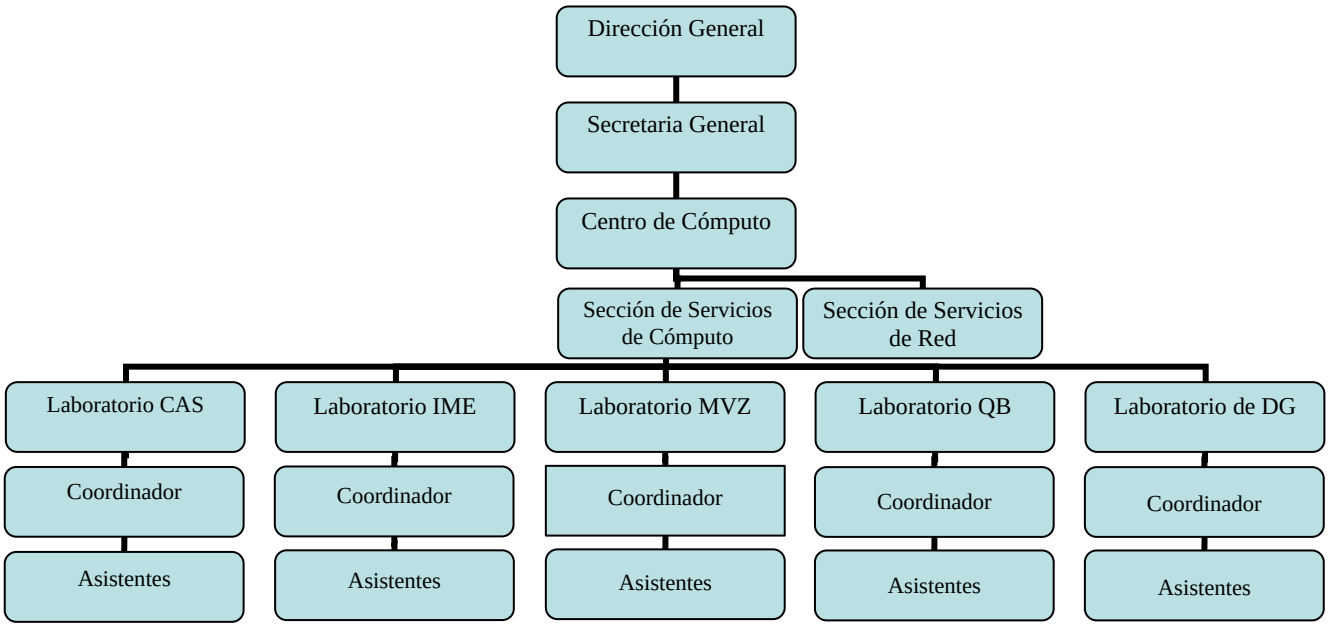



Figura 7.1. Estructura Organizacional de los Laboratorios

Las personas encargadas de los laboratorios de cómputo de la universidad son:

1. El coordinador del laboratorio cómputo que administra las operaciones del laboratorio.

2. Los asistentes de laboratorio (alumnos que prestan el servicio social) son los encargados de las operaciones diarias del laboratorio.

3. El departamento del centro de cómputo se encarga de las funciones administrativas generales de los laboratorios.

### 7.1.3 Análisis

Los laboratorios de cómputo cuentan con un sistema con las siguientes funciones:

- Servicio a usuarios: registro de entradas y salidas al laboratorio de los usuarios.
- Control de ingresos: registro de recibos de impresiones pagados por los usuarios.

El sistema ya no cumple con las necesidades de los laboratorios debido a que presenta deficiencias como son:

- No genera ningún tipo de reporte que pueda ser útil para los coordinadores de los laboratorios.
- Cuando se han registrado muchas entradas el sistema se vuelve lento y la información que genera es difícil de utilizar ya que esta guardada en archivos de texto.
- El sistema esta hecho en QBasic y ya no es posible actualizarlo o agregarle funcionalidad debido a que no se cuenta con la documentación del sistema para tal propósito.
- Como el sistema no esta en un entorno de red no se puede compartir la información con los diferentes laboratorios.

Tomando en cuenta estas necesidades se desarrolló un sistema llamado LabComp que administra las principales actividades que se llevan acabo en los laboratorios de cómputo de la Facultad de Estudios Superiores Cuautitlán. Este sistema permite:

- Concentración de la información en una base de datos.
- Posibilidad de agregarle nuevos módulos o modificar los existentes cuando así se requiera.
- Generación de reportes mediante el tratamiento de la información.
- Disposición de los saldos de los recibos de impresiones u otros servicios.
- Administración de los tiempos de los recursos de cómputo.

Se utilizó la plataforma Microsoft .NET debido a que proporciona las herramientas necesarias para la elaboración de aplicaciones Web de una forma sencilla y rápida mediante el uso de ASP .NET versión 2.0 para la construcción de la interfaz con el usuario, Visual Basic .NET versión 2.0 como lenguaje de programación, SQL Server Express versión 2005 para el manejo de bases datos y ADO .NET versión 2.0 para realizar la conexión entre la base de datos y el sistema. Se desarrolló según el modelo de aplicación de tres capas (ver Capitulo 6)**.**

Se usó el modelo de base de datos relacional y la arquitectura cliente/servidor. El primero debido a que es el que maneja Microsoft SQL Server Express versión 2005 y el segundo por que el número de computadoras que va a acceder al sistema es reducido y por lo cual no se necesita de un modelo más complejo.

La seguridad que utiliza la aplicación es mixta, es decir, implica la Autenticación Windows y la de Formularios (ver Capitulo 5).

El entorno de desarrollo que se utilizó fue el Visual Web Developer versión 2005 Express que es la herramienta gratuita que proporciona Microsoft para el desarrollo de aplicaciones Web.

## 7.2 Diseño del Sistema LabComp

### 7.2.1 Diseño de la Capa de Datos

El modelo entidad-relación de la base se muestra en la Figura 7.2.

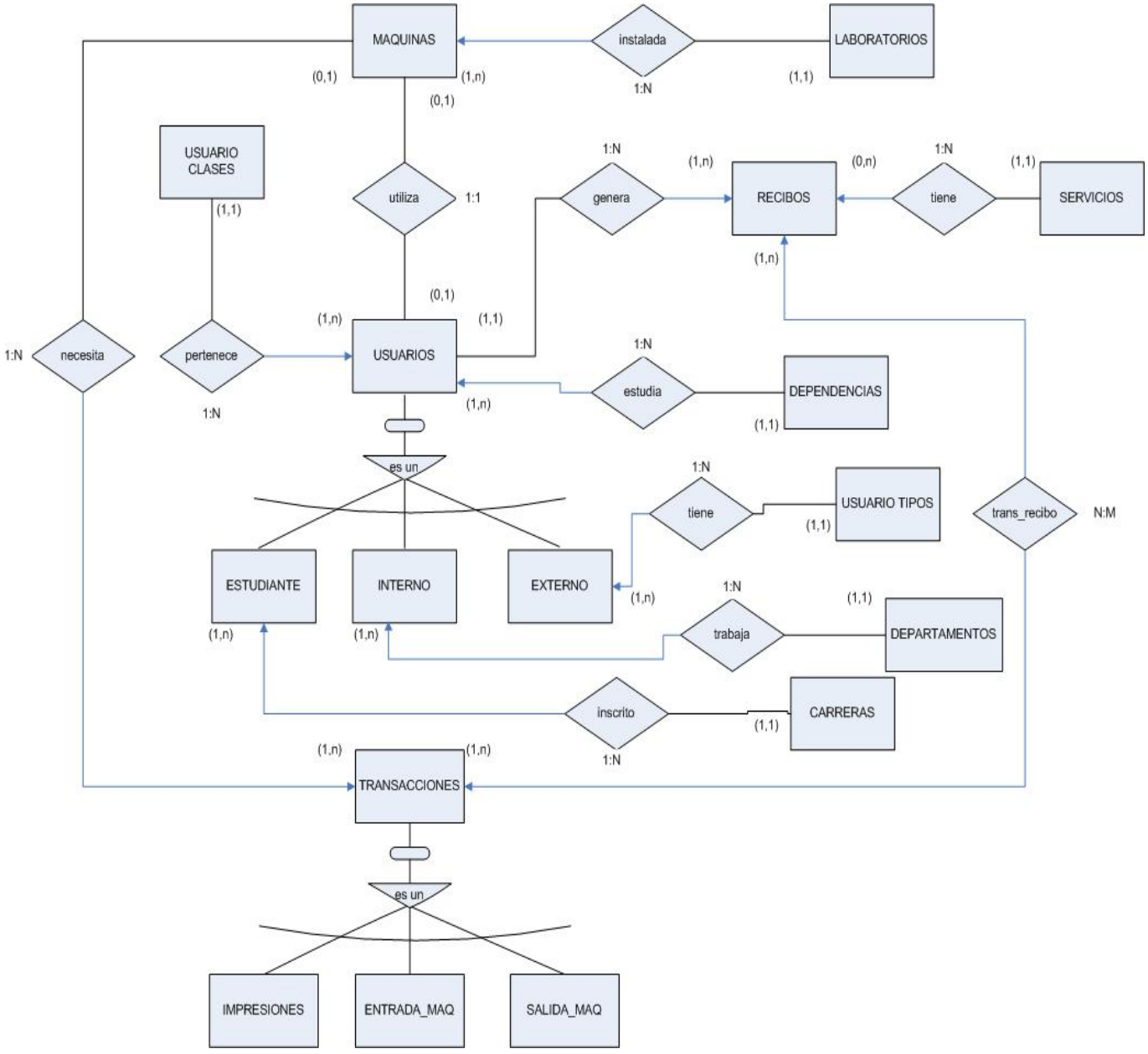



Figura 7.2 Modelo Entidad-Relación (LabComp)

El diagrama de la base de datos en SQL Server queda representado de acuerdo con lo que se muestra en la Figura 7.3.


Figura 7.3 Diagrama de la Base de Datos (LabComp)

Las tablas que conforman la base de datos así como su función dentro de la misma son:

**Tabla Usuarios:** Almacena los datos generales de un usuario.

| Campo | Tipo de dato | Descripción | Tipo de Llave |
|---|---|---|---|
| ID_USUARIO | Int | Identificador del usuario | Primaria |
| NOMBRE_USUARIO | Char(40) | Nombre completo del usuario | Ninguna |
| FECHA_REG_USUARIO | Datetime | Fecha de registro del usuario | Ninguna |
| ID_DEPENDENCIA | Int | Identificador de la dependencia | Foránea |
| ID_USUARIOCLASE | Int | Identificador de la clase del usuario | Foránea |
| ACTIVO | Bit | Muestra el estatus del usuario (activo o Inactivo) | Ninguna |

**Tabla Estudiante:** Almacena los datos de los estudiantes (alumnos).

| Campo | Tipo de dato | Descripción | Tipo de Llave |
|---|---|---|---|
| ID_ESTUDIANTE | Int | Identificador del estudiante | Primaria |
| ID_USUARIO | Int | Identificador del usuario al que pertenece | Foránea |
| NUM_CUENTA | Varchar(20) | Número de cuenta del estudiante | Ninguna |
| ID_CARRERA | Int | Identificador de la carrera a la que pertenece | Foránea |

**Tabla Interno:** Almacena los datos de las personas que laboran en la UNAM.

| Campo | Tipo de dato | Descripción | Tipo de Llave |
|---|---|---|---|
| ID_INTERNO | Int | Identificador del interno | Primaria |
| ID_USUARIO | Int | Identificador del usuario al que pertenece | Foránea |
| NUM_TRABAJADOR | Varchar(20) | Número de trabajador del interno | Ninguna |
| RFC | Char(10) | R.F.C. del interno | Foránea |
| ID_USUARIOTIPO | Int | Identificador del tipo de usuario que es | Foránea |
| ID_DEPARTAMENTO | Int | Identificador del departamento en el que labora | Foránea |

**Tabla Externo:** Almacena los datos de las personas que vienen de instituciones que no pertenecen a la UNAM.

| Campo | Tipo de dato | Descripción | Tipo de Llave |
|---|---|---|---|
| ID_EXTERNO | Int | Identificador del externo | Primaria |
| ID_USUARIO | Int | Identificador del usuario al que pertenece | Foránea |
| NUM_CREDENCIAL | Varchar(20) | Número de la identificación del externo | Ninguna |
| ID_USUARIOTIPO | Int | Identificador del tipo de usuario que es | Foránea |
| ID_DEPENDENCIA | Int | Identificador de la dependencia de la que proviene el externo | Foránea |

**Tabla UsuarioClases:** Catálogo de las clases que puede ser un usuario (Estudiante, Interno o Externo).

| Campo | Tipo de dato | Descripción | Tipo de Llave |
|---|---|---|---|
| ID_USUARIOCLASE | Int | Identificador del usuarioclase | Primaria |
| DESCRIPCION | Char(20) | Nombre de la clase de usuario | Ninguna |

**Tabla Dependencias:** Catálogo de las diferentes escuelas de donde puede provenir un usuario.

| Campo | Tipo de dato | Descripción | Tipo de Llave |
|---|---|---|---|
| ID_DEPENDENCIA | Int | Identificador de la dependencia | Primaria |
| DESCRIPCION | Char(100) | Nombre de la dependencia o escuela | Ninguna |
| ACTIVO | Bit | Muestra el estatus de la dependencia (Activo ó Inactivo) | Ninguna |
| ID_USUARIOSISTEMA | Int | Identificador del usuario que dio de alta la dependencia | Foránea |
| FECHA_CAPTURA | Datetime | Fecha en que se capturo la dependencia | Ninguna |

**Tabla Carreras:** Catálogo de las diferentes carreras que puede estar cursando un usuario (Estudiante).

| Campo | Tipo de dato | Descripción | Tipo de Llave |
|---|---|---|---|
| ID_CARRERA | Int | Identificador de la carrera | Primaria |
| DESCRIPCION | Char(100) | Nombre de la carrera | Ninguna |
| ABREVIATURA | Char(10) | Nombre corto de la carrera | Ninguna |
| ACTIVO | Bit | Muestra el estatus de la carrera (Activo ó Inactivo) | Ninguna |
| ID_USUARIOSISTEMA | Int | Identificador del usuario que dio de alta la carrera | Foránea |
| FECHA_CAPTURA | Datetime | Fecha en que se capturo la carrera | Ninguna |

**Tabla Departamentos:** Catálogo de los diferentes departamentos en los que puede estar trabajando un usuario (Interno).

| Campo | Tipo de dato | Descripción | Tipo de Llave |
|---|---|---|---|
| ID_DEPARTAMENTO | Int | Identificador del departamento | Primaria |
| DESCRIPCION | Char(20) | Nombre del departamento | Ninguna |
| ACTIVO | Bit | Muestra el estatus del departamento (Activo ó Inactivo) | Ninguna |
| ID_USUARIOSISTEMA | Int | Identificador del usuario que dio de alta el departamento | Foránea |
| FECHA_CAPTURA | Datetime | Fecha en que se capturo el departamento | Ninguna |

**Tabla UsuarioTipos:** Catálogo de las diferentes categorías que puede tener un usuario (Interno ó Externo), ya sea Administrativo, Académico, etc.

| Campo | Tipo de dato | Descripción | Tipo de Llave |
|---|---|---|---|
| ID_USUARIOTIPO | Int | Identificador del usuariotipo | Primaria |
| DESCRIPCION | Char(20) | Nombre del tipo de usuario | Ninguna |
| ACTIVO | Bit | Muestra el estatus del tipo de usuario (Activo ó Inactivo) | Ninguna |
| ID_USUARIOSISTEMA | Int | Identificador del usuario que dio de alta el | Foránea |

| | | | |
|---|---|---|---|
| | | tipo de usuario | |
| FECHA_CAPTURA | Datetime | Fecha en que se capturo el tipo de usuario | Ninguna |

**Tabla Recibos:** Almacena los datos de los recibos que son llevados por los usuarios, ya sea para impresiones o algún otro servicio.

| Campo | Tipo de dato | Descripción | Tipo de Llave |
|---|---|---|---|
| ID_RECIBO | Int | Identificador del recibo | Primaria |
| SERIE_RECIBO | Char(1) | Serie del recibo | Ninguna |
| NUMERO_RECIBO | Int | Número del recibo | Ninguna |
| FECHA_RECIBO | Datetime | Fecha en la que se elaboro el recibo | Ninguna |
| IMPORTE_RECIBO | Money | Cantidad monetaria que ampara el recibo | Ninguna |
| CANTIDAD_RECIBO | Int | Cantidad en unidad del servicio que ampara el recibo | Ninguna |
| SALDO_RECIBO | Int | Cantidad en unidad que le queda al recibo sin utilizar | Ninguna |
| SERVICIO | Int | Identificador del servicio que ampara el recibo | Foránea |
| ID_USUARIO | Int | Identificador del usuario al que pertenece el recibo | Foránea |
| TOTAL_RECIBO | Int | Histórico de la cantidad en unidad del servicio que ampra el recibo | Ninguna |
| REALIZADAS | Int | Cantidad en unidad del servicio que ha sido consumido o utilizado | Ninguna |
| ID_LABORATORIO | Int | Identificador del laboratorio en donde se capturo el recibo | Foránea |

**Tabla Servicios:** Catálogo de los servicios que prestan los laboratorios.

| Campo | Tipo de dato | Descripción | Tipo de Llave |
|---|---|---|---|
| ID_SERVICIO | Int | Identificador del servicio | Primaria |
| CONCEPTO_SERVICIO | Char(20) | Nombre del servicio | Ninguna |
| COSTO_SERV | Money | Costo del servicio | Ninguna |
| ACTIVO | Bit | Muestra el estatus del servicio (Activo ó | Ninguna |

| | | | |
|---|---|---|---|
| | | Inactivo) | |
| ID_USUARIOSISTEMA | Int | Identificador del usuario que dio de alta el servicio | Foránea |
| FECHA_CAPTURA | Datetime | Fecha en que se dio de alta el servicio | Ninguna |

**Tabla Transacciones:** Almacena los datos generales de las diferentes operaciones que se realizan en los laboratorios (Descarga de recibos, Entradas y Salidas).

| Campo | Tipo de dato | Descripción | Tipo de Llave |
|---|---|---|---|
| ID_TRANS | Int | Identificador de la transacción | Primaria |
| ID_USUARIO | Int | Identificador del usuario que realizo la transacción | Foránea |
| FECHA_TRANS | Datetime | Fecha en que se realizo la transacción | Ninguna |
| TIPO_SERVICIO | Int | Tipo de servicio que involucra la transacción (Servicios, Entrada ó Salida) | Ninguna |
| ID_MÁQUINA | Int | Identificador de la máquina (computadora) que se involucra en la transacción | Foránea |

**Tabla Impresiones:** Almacena el detalle de la transacción cuando se involucra la descarga de recibos para impresiones o algún otro servicio.

| Campo | Tipo de dato | Descripción | Tipo de Llave |
|---|---|---|---|
| ID_IMPRESION | Int | Identificador de la impresión | Primaria |
| ID_TRANS | Int | Identificador de la transacción a la que corresponde | Foránea |
| CANT_IMP | Int | Cantidad que se utilizo del servicio correspondiente | Ninguna |
| ID_SERVICIO | Int | Identificador del servicio que involucro la transacción | Foránea |

**Tabla Entrada_Maq:** Almacena el detalle de la transacción cuando se involucra la entrada al laboratorio por parte de un usuario.

| Campo | Tipo de dato | Descripción | Tipo de Llave |
|---|---|---|---|
| ID_ENTRADAMAQ | Int | Identificador de la entradamáquina | Primaria |
| ID_TRANS | Int | Identificador de la transacción a la que corresponde | Foránea |
| HORA_ENTRADA | DateTime | Hora en la que ingreso el usuario al laboratorio | Ninguna |

**Tabla Salida_Maq:** Almacena el detalle de la transacción cuando se involucra la salida del laboratorio por parte de un usuario.

| Campo | Tipo de dato | Descripción | Tipo de Llave |
|---|---|---|---|
| ID_SALIDAMAQ | Int | Identificador de la salidamáquina | Primaria |
| ID_TRANS | Int | Identificador de la transacción a la que corresponde | Foránea |
| HORA_SALIDA | DateTime | Hora en la que salio el usuario del laboratorio | Ninguna |
| CANT_HORAS | Varchar(10) | Diferencia entre la hora de entrada y salida del usuario | Ninguna |

**Tabla Utiliza:** Esta tabla contiene información que permite saber que usuarios se encuentran usando una máquina.

| Campo | Tipo de dato | Descripción | Tipo de Llave |
|---|---|---|---|
| ID_UTILIZA | Int | Identificador de utiliza | Primaria |
| ID_USUARIO | Int | Identificador del usuario que se encuentra utilizando una máquina | Foránea |
| ID_MÁQUINA | Int | Identificador de la máquina que se encuentra en uso | Foránea |
| ID_TRANS | Int | Identificador de la transacción que involucra | Foránea |

**Tabla Máquinas:** Catálogo de las máquinas (Computadoras) que se encuentran en los laboratorios de computo.

| Campo | Tipo de dato | Descripción | Tipo de Llave |
|---|---|---|---|
| ID_MÁQUINA | Int | Identificador de la máquina | Primaria |
| NUMERO_MAQ | Int | Número de la máquina | Ninguna |
| ID_LABORATORIO | Int | Identificador del laboratorio al que pertenece la máquina | Foránea |
| INFORMACION | Varchar(100) | Información sobre la máquina | Ninguna |
| BANDERA_MAQ | Int | Permite saber si una computadora esta desocupada (0), ocupada (1) o inhabilitada (2) | Ninguna |
| IP_MAQ | Varchar(15) | Dirección IP de la máquina | Ninguna |
| MOSTRAR_MAQ | Bit | Permite determinar si una máquina se muestra o no en el programa | Ninguna |
| ID_USUARIOSISTEMA | Int | Identificador del usuario que dio de alta la máquina | Foránea |
| FECHA_CAPTURA | Datetime | Fecha en la que se dio de alta la máquina | Ninguna |

**Tabla Laboratorios:** Catálogo de los laboratorios que integran la Facultad de Estudios Superiores Cuautitlán Campo 1 y 4.

| Campo | Tipo de dato | Descripción | Tipo de Llave |
|---|---|---|---|
| ID_LABORATORIO | Int | Identificador del Laboratorio | Primaria |
| NOMBRE | Varchar(100) | Nombre completo del laboratorio | Ninguna |
| ABREVIATURA | Varchar(10) | Nombre corto del laboratorio | Ninguna |
| ACTIVO | Bit | Muestra el estatus del laboratorio (Activo ó Inactivo) | Ninguna |
| ID_USUARIOSISTEMA | Int | Identificador del usuario que dio de alta el laboratorio | Foránea |
| FECHA_CAPTURA | Datetime | Fecha en la que se capturo el laboratorio | Ninguna |

**Tabla Trans_Recibo:** Registra todas las transacciones que afectan a un determinado recibo, así como la cantidad de cada una.

| Campo | Tipo de dato | Descripción | Tipo de Llave |
|---|---|---|---|
| ID_TRANRECIBO | Int | Identificador de tran_recibo | Primaria |
| ID_TRANS | Int | Identificador de la transacción | Foránea |
| ID_RECIBO | Int | Identificador del recibo | Foránea |
| CANTIDAD | Int | Cantidad que afecta el recibo | Ninguna |

**Tabla UsuariosSistema:** Almacena los datos de los coordinadores ó asistentes del laboratorio que pueden acceder al modulo de administración del sistema.

| Campo | Tipo de dato | Descripción | Tipo de Llave |
|---|---|---|---|
| ID_USUARIOSISTEMA | Int | Identificador del usuariosistema | Primaria |
| NOMBRE_COMPLETO | Varchar(50) | Nombre completo del coordinador o asistente | Ninguna |
| PUESTO | Varchar(30) | Cargo que ocupa en el laboratorio | Ninguna |
| ACTIVO | Bit | Muestra el estatus del usuariosistema (Activo o Inactivo) | Ninguna |
| ID_LABORATORIO | Int | Identificador del laboratorio al que pertenece | Foránea |
| USERNAME | Varchar(15) | Nombre para entrar al sistema | Ninguna |
| PASSWORD | Varchar(15) | Contraseña para entrar al sistema | Ninguna |
| ID_USUARIOCAPTURA | Int | Identificador del usuario que dio de alta al usuario del sistema | Ciclica |
| FECHA_CAPTURA | Datetime | Fecha en que se capturo el usuario del sistema | Int |

### 7.2.2 Diseño de la Capa de Negocios

En esta capa se diseñan los objetos de negocios las cuales como se describió anteriormente, son porciones reutilizables de código que pueden agregarse a las aplicaciones.

El sistema LabComp utiliza dos componentes que contiene la lógica de negocios. Estos componentes fueron realizados con el propósito de utilizar el modelo de tres capas y hacer

más sencilla la programación (debido a que se encapsula la complejidad)[17]. Estos componentes son:

- LabConnection – Contiene los métodos para hacer acceso a una base de datos.

- LabClass- Contiene las clases utilizadas por el sistema para interactuar con la capa de datos y la capa de presentación. También se encargan de las validaciones necesarias para el buen funcionamiento del sistema.

El componente LabConnection esta integrado de la siguiente forma:

| Clase | Método ó Función | Descripción |
|---|---|---|
| Conexión | GetRows(query) | Sirve para obtener una serie de columnas resultantes de una consulta a la base de datos (SELECT). |
| | ExecuteQuery(query) | Sirve para ejecutar una consulta del tipo INSERT, DELETE ó UPDATE a la base de datos. |
| | ExecuteSP(comando) | Sirve para ejecutar un procedimiento almacenado en la base de datos. |

La cadena de conexión se almacena en el archivo machine.config (archivo de configuración utilizado por el .NET Framework). Esto se hace con la finalidad de que si se desea instalar el sistema en otro servidor sólo se cambie la cadena de conexión en el archivo y no en código evitando que se compile todo el sistema de nuevo.

El componente LabClass esta integrado de la siguiente forma:

[17] Consultar Capitulo 6 inciso 6.2 de este trabajo

| Clase | Método ó Función | Descripción |
|---|---|---|
| Usuario | GetData(id) | Obtiene los datos de un usuario en especifico |
| | InsertUpdate() | Construye un comando para insertar o actualizar los datos de un usuario en base al ID (identificador único) de la clase. Si es 0 inserta y en caso contrario actualiza |
| | Delete() | Construye un comando para borrar un usuario |
| | CleanData() | Limpia los valores de la clase |
| | New() | Inicializa un objeto usuario en blanco |
| | New(id) | Inicializa un objeto usuario y carga sus datos |
| Carrera | GetData(id) | Obtiene los datos de una carrera en especifico |
| | InsertUpdate() | Construye un comando para insertar o actualizar los datos de una carrera en base al ID de la clase. Si es 0 inserta y en caso contrario actualiza. |
| | Delete() | Construye un comando para borrar una carrera |
| | CleanData() | Limpia los valores de la clase |
| | New() | Inicializa un objeto carrera en blanco |
| | New(id) | Inicializa un objeto carrera y carga sus datos |
| Departamento | GetData(id) | Obtiene los datos de un departamento en especifico |

| Clase | Método ó Función | Descripción |
|---|---|---|
| | InsertUpdate() | Construye un comando para insertar o actualizar los datos de un departamento en base al ID de la clase. Si es 0 inserta y en caso contrario actualiza. |
| | Delete() | Construye un comando para borrar un departamento |
| | CleanData() | Limpia los valores de la clase |
| | New() | Inicializa un objeto departamento en blanco |
| | New(id) | Inicializa un objeto departamento y carga sus datos |
| Dependencia | GetData(id) | Obtiene los datos de una dependencia en especifico |
| | InsertUpdate() | Construye un comando para insertar o actualizar los datos de una dependencia en base al ID de la clase. Si es 0 inserta y en caso contrario actualiza. |
| | Delete() | Construye un comando para borrar una dependencia |
| | CleanData() | Limpia los valores de la clase |
| | New() | Inicializa un objeto dependencia en blanco |
| | New(id) | Inicializa un objeto dependencia y carga sus datos |
| Entrada Máquina | GetData(id) | Obtiene los datos de una entrada máquina en especifico |

| Clase | Método ó Función | Descripción |
|---|---|---|
| | InsertUpdate() | Construye un comando para insertar o actualizar los datos de una entrada máquina en base al ID de la clase. Si es 0 inserta y en caso contrario actualiza. |
| | Delete() | Construye un comando para borrar una entrada máquina |
| | CleanData() | Limpia los valores de la clase |
| | New() | Inicializa un objeto entrada máquina en blanco |
| | New(id) | Inicializa un objeto entrada máquina y carga sus datos |
| Estudiante | GetData(id) | Obtiene los datos de un estudiante en especifico |
| | InsertUpdate() | Construye un comando para insertar o actualizar los datos de un estudiante en base al ID de la clase. Si es 0 inserta y en caso contrario actualiza. |
| | Delete() | Construye un comando para borrar un estudiante |
| | CleanData() | Limpia los valores de la clase |
| | New() | Inicializa un objeto estudiante en blanco |
| | New(id) | Inicializa un objeto estudiante y carga sus datos |
| Externo | GetData(id) | Obtiene los datos de un externo en especifico |

| Clase | Método ó Función | Descripción |
|---|---|---|
| | InsertUpdate() | Construye un comando para insertar o actualizar los datos de un externo en base al ID de la clase. Si es 0 inserta y en caso contrario actualiza. |
| | Delete() | Construye un comando para borrar un externo |
| | CleanData() | Limpia los valores de la clase |
| | New() | Inicializa un objeto externo en blanco |
| | New(id) | Inicializa un objeto externo y carga sus datos |
| Impresión | GetData(id) | Obtiene los datos de una impresión en especifico |
| | InsertUpdate() | Construye un comando para insertar o actualizar los datos de una impresión en base al ID de la clase. Si es 0 inserta y en caso contrario actualiza. |
| | Delete() | Construye un comando para borrar una impresión |
| | CleanData() | Limpia los valores de la clase |
| | New() | Inicializa un objeto impresión en blanco |
| | New(id) | Inicializa un objeto impresión y carga sus datos |
| Interno | GetData(id) | Obtiene los datos de un interno en especifico |

| Clase | Método ó Función | Descripción |
|---|---|---|
| | InsertUpdate() | Construye un comando para insertar o actualizar los datos de un interno en base al ID de la clase. Si es 0 inserta y en caso contrario actualiza. |
| | Delete() | Construye un comando para borrar un interno |
| | CleanData() | Limpia los valores de la clase |
| | New() | Inicializa un objeto interno en blanco |
| | New(id) | Inicializa un objeto interno y carga sus datos |
| Laboratorio | GetData(id) | Obtiene los datos de un laboratorio en especifico |
| | InsertUpdate() | Construye un comando para insertar o actualizar los datos de un laboratorio en base al ID de la clase. Si es 0 inserta y en caso contrario actualiza. |
| | Delete() | Construye un comando para borrar un laboratorio |
| | CleanData() | Limpia los valores de la clase |
| | New() | Inicializa un objeto laboratorio en blanco |
| | New(id) | Inicializa un objeto laboratorio y carga sus datos |
| Máquina | GetData(id) | Obtiene los datos de una máquina en especifico |

| Clase | Método ó Función | Descripción |
|---|---|---|
| | InsertUpdate() | Construye un comando para insertar o actualizar los datos de una máquina en base al ID de la clase. Si es 0 inserta y en caso contrario actualiza. |
| | Delete() | Construye un comando para borrar una máquina |
| | CleanData() | Limpia los valores de la clase |
| | New() | Inicializa un objeto máquina en blanco |
| | New(id) | Inicializa un objeto máquina y carga sus datos |
| | ObtenerUbicacion(ip) | Obtiene el laboratorio en que se encuentra una máquina determinada mediante su dirección IP |
| Recibo | GetData(id) | Obtiene los datos de un recibo en especifico |
| | InsertUpdate() | Construye un comando para insertar o actualizar los datos de un recibo en base al ID de la clase. Si es 0 inserta y en caso contrario actualiza. |
| | Delete() | Construye un comando para borrar una recibo |
| | CleanData() | Limpia los valores de la clase |
| | New() | Inicializa un objeto recibo en blanco |
| | New(id) | Inicializa un objeto recibo y carga sus datos |

| Clase | Método ó Función | Descripción |
|---|---|---|
| | ObtenerSaldoRecibo(id,idservicio) | Obtiene todos los recibos de un determinado servicio que pertenezcan a un usuario. Sólo obtiene aquellos recibos cuyo saldo sea mayor a cero |
| | DescargarRecibo(id,idservicio,total) | Realiza una resta del saldo del recibo con la cantidad utilizada |
| Salida Máquina | GetData(id) | Obtiene los datos de una salida máquina en especifico |
| | InsertUpdate() | Construye un comando para insertar o actualizar los datos de una salida máquina en base al ID de la clase. Si es 0 inserta y en caso contrario actualiza. |
| | Delete() | Construye un comando para borrar una salida máquina |
| | CleanData() | Limpia los valores de la clase |
| | New() | Inicializa un objeto salida máquina en blanco |
| | New(id) | Inicializa un objeto salida máquina y carga sus datos |
| Servicio | GetData(id) | Obtiene los datos de un servicio en especifico |
| | InsertUpdate() | Construye un comando para insertar o actualizar los datos de un servicio en base al ID de la clase. Si es 0 inserta y en caso contrario actualiza. |

| Clase | Método ó Función | Descripción |
|---|---|---|
| | Delete() | Construye un comando para borrar un servicio |
| | CleanData() | Limpia los valores de la clase |
| | New() | Inicializa un objeto servicio en blanco |
| | New(id) | Inicializa un objeto servicio y carga sus datos |
| Transacción | GetData(id) | Obtiene los datos de una transacción en especifico |
| | InsertUpdate() | Construye un comando para insertar o actualizar los datos de una transacción en base al ID de la clase. Si es 0 inserta y en caso contrario actualiza. |
| | Delete() | Construye un comando para borrar una transacción |
| | CleanData() | Limpia los valores de la clase |
| | New() | Inicializa un objeto transacción en blanco |
| | New(id) | Inicializa un objeto transacción y carga sus datos |
| Usuario Tipo | GetData(id) | Obtiene los datos de un tipo de usuario en especifico |
| | InsertUpdate() | Construye un comando para insertar o actualizar los datos de un tipo de usuario en base al ID de la clase. Si es 0 inserta y en caso contrario actualiza. |

| Clase | Método ó Función | Descripción |
|---|---|---|
| | Delete() | Construye un comando para borrar un tipo de usuario |
| | CleanData() | Limpia los valores de la clase |
| | New() | Inicializa un objeto usuario tipo en blanco |
| | New(id) | Inicializa un objeto usuario tipo y carga sus datos |
| Usuario Sistema | GetData(id) | Obtiene los datos de un usuario de sistema en especifico |
| | InsertUpdate() | Construye un comando para insertar o actualizar los datos de un usuario de sistema en base al ID de la clase. Si es 0 inserta y en caso contrario actualiza. |
| | Delete() | Construye un comando para borrar un usuario de sistema |
| | CleanData() | Limpia los valores de la clase |
| | New() | Inicializa un objeto usuario sistema en blanco |
| | New(id) | Inicializa un objeto usuario sistema y carga sus datos |

Con el uso de estos dos componentes la implementación del código queda separada de la interfaz del usuario, haciendo más fácil el mantenimiento al sistema y dándole modularidad en caso de que se quiera agregar nuevos módulos ó funciones.

### 7.2.3 Diseño de la Capa de Presentación

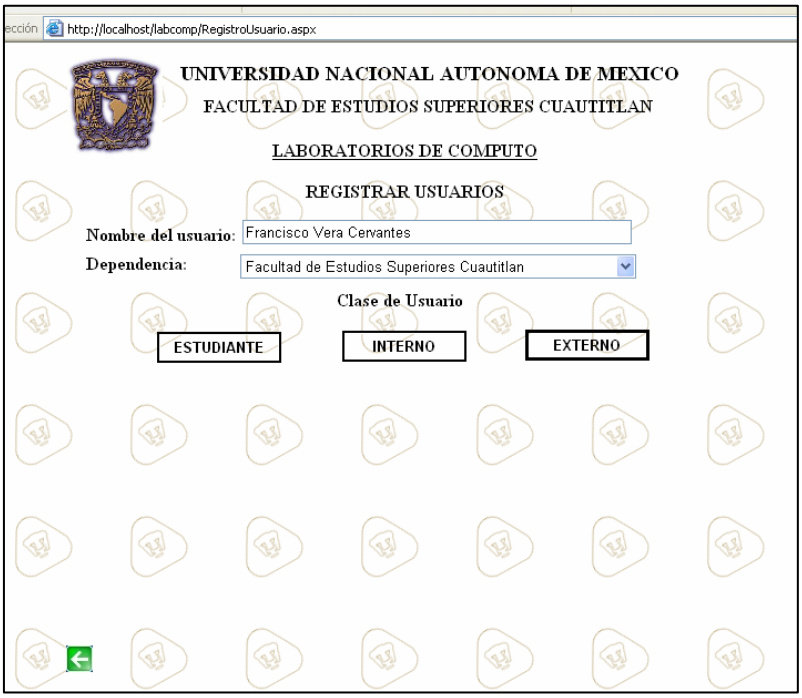
**Página Principal (Default.aspx):** Esta página esta compuesta por 5 opciones las cuales se explican a continuación:


1. **USUARIOS**

**Página Menú Usuario (MenuUsuario.aspx):** Esta página nos permite Registrar, Modificar y Listar usuarios. Todas las ventanas tienen un cuadro color verde con una flecha que señala hacia atrás, si uno desea regresar a la página anterior puede elegir este cuadro.

**Página Registrar Usuario (RegistroUsuario.aspx):** El primer paso es elegir el botón de Registrar aquí deberá teclear un nombre de usuario y seleccionar a que dependencia pertenece (la dependencia que da por primera vez el sistema es Facultad de Estudios Superiores Cuautitlan).

Dependiendo la clase de usuario se deben de capturar diferentes datos. Las opciones pueden ser Estudiante, Interno ó Externo.




**ESTUDIANTE:** Eligiendo el botón de Estudiante aparece un cuadro en donde se registra el Número de cuenta y selecciona la Carrera a la cual pertenece el estudiante. Si los datos están completos y correctos se puede registrar el Estudiante eligiendo la opción de Agregar Usuario si desea detener este registro debe oprimir el botón de Cancelar.[18]

[18] El sistema contempla una carrera por número de cuenta. Sin importar que el alumno este inscrito en más de una carrera.


**INTERNO:** Eligiendo el botón de Interno pueden registrarse trabajadores de la Facultad aquí se deben de llenar los campos de Numero de Trabajador, R.F.C., seleccionar el Tipo de Usuario hay dos tipos Administrativo o Académico y seleccionar su Departamento, si los datos están completos y correctos se puede registrar el trabajador oprimiendo la opción de Agregar Usuario si desea detener este registro debe oprimir el botón de Cancelar.

**EXTERNO:** Eligiendo el botón de Externo pueden registrarse Alumnos de otras dependencias aquí se deben de llenar los campos de Numero de Credencial y seleccionar Tipo de Usuario hay dos tipos Administrativo o Académico, si los datos están completos y correctos se puede registrar el usuario externo oprimiendo la opción de Agregar Usuario si desea detener este registro debe oprimir el botón de Cancelar.


**MODIFICAR USUARIO (MenuUsuario.aspx):** Para modificar usuarios es necesario ingresar al Menú Usuario y elegir el botón de Modificar.

**Página Modificar Usuarios (ModificaUsuario.aspx):** Esta página nos permite hacer modificaciones a los registros de usuarios que hayamos dado de alta en el sistema.


Para realizar una modificación existen cinco formas:

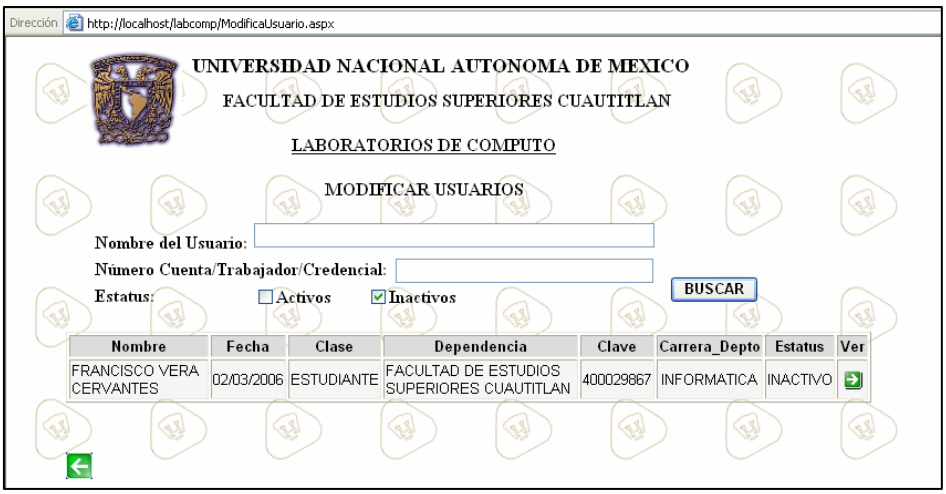
- Introducir un nombre de usuario y oprimir el botón **Buscar**. La búsqueda muestra todas las coincidencias que encuentre con ese nombre. Aparecerá una cuadricula con los datos del usuario, si desea hacer modificaciones a un registro sólo debe señalar el cuadro verde con la flecha que señala hacia delante, en la columna **Ver**.

**Página Modificar Usuarios (ModUsuario.aspx):** Aparece esta pantalla donde podrá hacer modificaciones a los datos del usuario así como cambiar su estado Activo o Inactivo (señalando Activo nuevamente). Si los datos están completos y correctos se puede modificar el usuario oprimiendo la opción de Guardar si desea detener esta modificación debe oprimir el botón de Cancelar este regresara al Menú de Usuario.

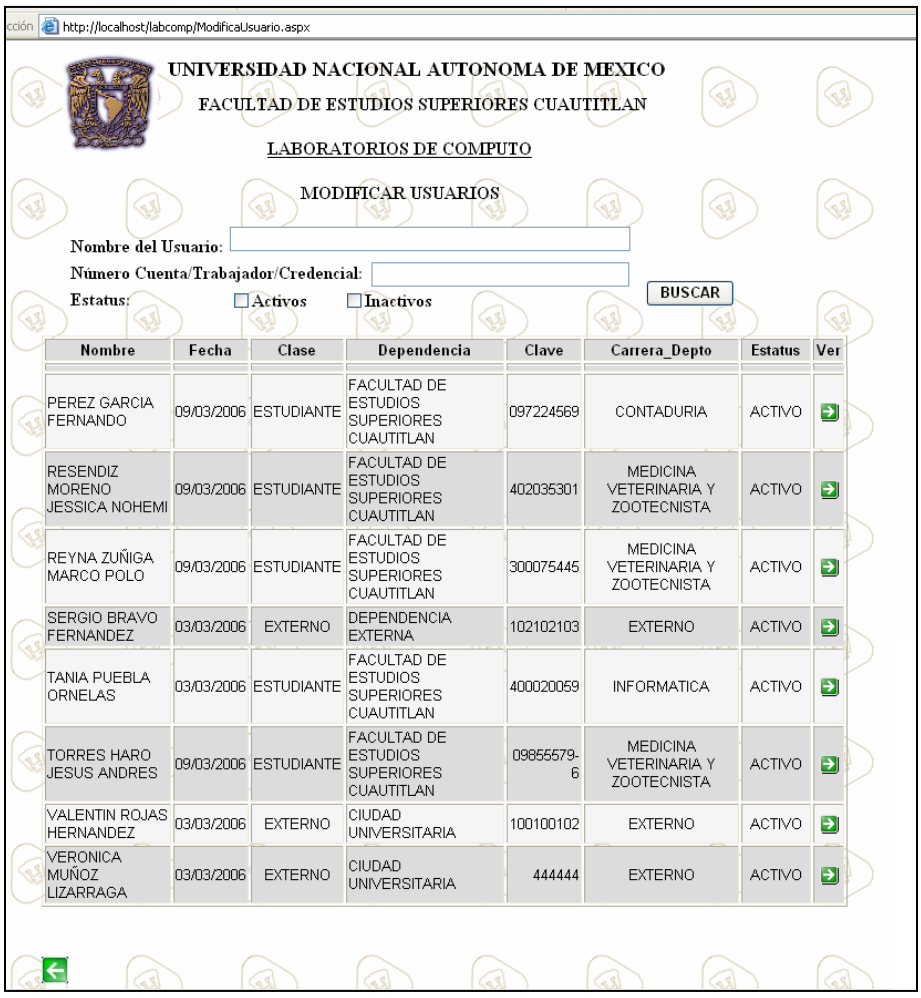

- Introducir un número de cuenta y oprimir el botón **Buscar**. La búsqueda muestra la coincidencia que encuentre con ese número. Aparecerá una cuadricula con los datos del usuario, si desea hacer modificaciones al registro sólo debe de señalar el cuadro verde con la flecha que señala hacia delante, en la columna **Ver**. Las modificaciones se realizan de la misma forma que cuando se busca por nombre.

- Señalando el cuadro de Activos y oprimir el botón **Buscar**. La búsqueda muestra las coincidencias que encuentre. Aparecerá una cuadricula con los datos del usuario, si desea hacer modificaciones al registro sólo debe de señalar el cuadro verde con la flecha que señala hacia delante, en la columna **Ver**. Las modificaciones se realizan de la misma forma que en las anteriores.


- Señalando el cuadro de Inactivos y oprimir el botón **Buscar**. La búsqueda muestra las coincidencias que encuentre. Las modificaciones se realizan de la misma forma que en las anteriores.



- En el caso de no introducir ningún dato y pulsar el botón **Buscar**. La búsqueda muestra todos los usuarios Activos e Inactivos. Las modificaciones se realizan de la misma forma que en las anteriores.




| Nombre | Fecha | Clase | Dependencia | Clave | Carrera_Depto | Estatus | Ver |
|---|---|---|---|---|---|---|---|
| PEREZ GARCIA FERNANDO | 09/03/2006 | ESTUDIANTE | FACULTAD DE ESTUDIOS SUPERIORES CUAUTITLAN | 097224569 | CONTADURIA | ACTIVO | |
| RESENDIZ MORENO JESSICA NOHEMI | 09/03/2006 | ESTUDIANTE | FACULTAD DE ESTUDIOS SUPERIORES CUAUTITLAN | 402035301 | MEDICINA VETERINARIA Y ZOOTECNISTA | ACTIVO | |
| REYNA ZUÑIGA MARCO POLO | 09/03/2006 | ESTUDIANTE | FACULTAD DE ESTUDIOS SUPERIORES CUAUTITLAN | 300075445 | MEDICINA VETERINARIA Y ZOOTECNISTA | ACTIVO | |
| SERGIO BRAVO FERNANDEZ | 03/03/2006 | EXTERNO | DEPENDENCIA EXTERNA | 102102103 | EXTERNO | ACTIVO | |
| TANIA PUEBLA ORNELAS | 03/03/2006 | ESTUDIANTE | FACULTAD DE ESTUDIOS SUPERIORES CUAUTITLAN | 400020059 | INFORMATICA | ACTIVO | |
| TORRES HARO JESUS ANDRES | 09/03/2006 | ESTUDIANTE | FACULTAD DE ESTUDIOS SUPERIORES CUAUTITLAN | 09855579-6 | MEDICINA VETERINARIA Y ZOOTECNISTA | ACTIVO | |
| VALENTIN ROJAS HERNANDEZ | 03/03/2006 | EXTERNO | CIUDAD UNIVERSITARIA | 100100102 | EXTERNO | ACTIVO | |
| VERONICA MUÑOZ LIZARRAGA | 03/03/2006 | EXTERNO | CIUDAD UNIVERSITARIA | 444444 | EXTERNO | ACTIVO | |

**LISTAR USUARIO (MenuUsuario.aspx):** Para ver los usuarios del sistema es necesario ingresar al Menú Usuario y elegir el botón de Listar.


**Página Listar Usuario (ListaUsuario.aspx):** Sirve para realizar búsquedas de los usuarios que se encuentran capturados en la base de datos.

Para realizar una búsqueda existen cinco formas:

- Introducir un nombre de usuario y oprimir el botón **Buscar**. La búsqueda muestra todas las coincidencias que encuentre con ese nombre estas se ven reflejadas en una cuadricula. La cuadricula sólo muestra 15 resultados por esa razón en su esquina superior derecha aparece un número con el cual podrá desplazarse a las siguientes páginas.


- Introducir un número de cuenta y oprimir el botón **Buscar.** La búsqueda muestra todas las coincidencias que encuentre con ese número de cuenta estas se ven reflejadas en una cuadricula. Hay un número en la esquina superior derecha de la

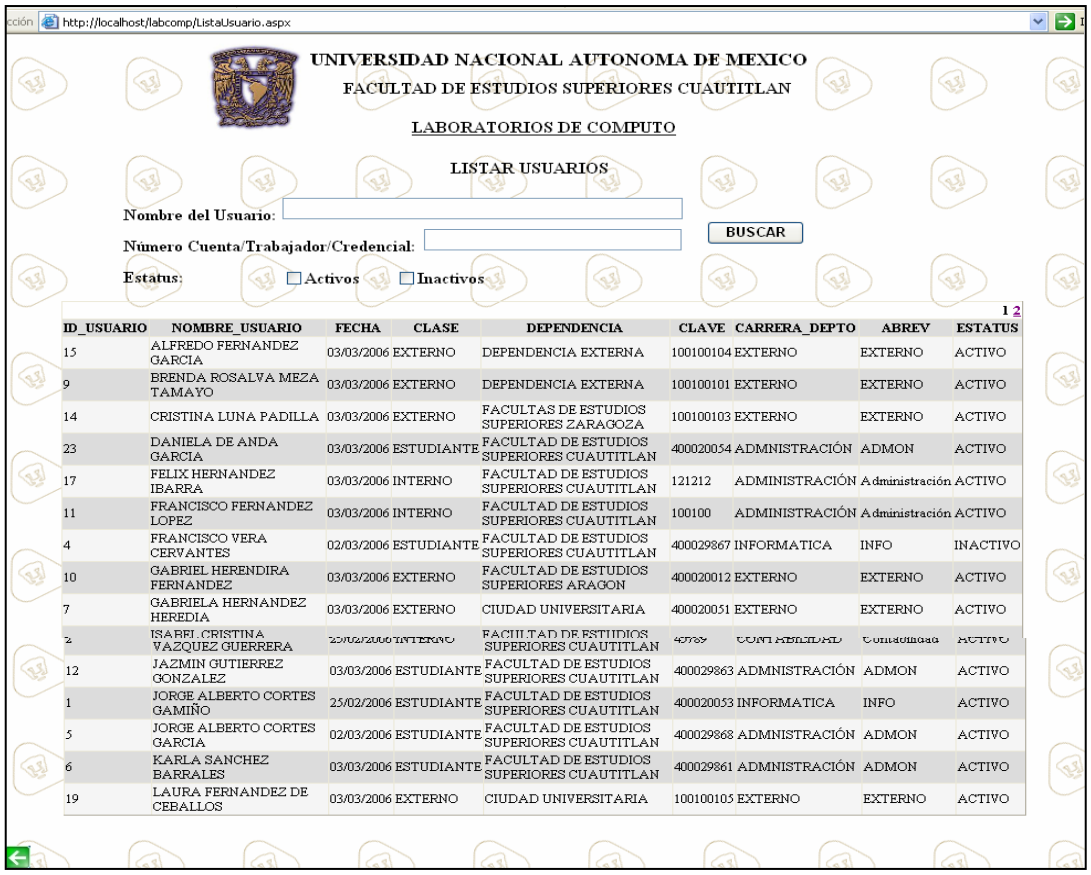
cuadricula con el cual podrá desplazarse a otras páginas con los resultados obtenidos.

- Seleccionar la opción de Activos y oprimir el botón **Buscar**. La búsqueda muestra todas las coincidencias que encuentra con el dato proporcionado estas se ven reflejadas en una cuadricula. El número en la esquina superior derecha desplaza a otras páginas con los resultados obtenidos.


| ID_USUARIO | NOMBRE_USUARIO | FECHA | CLASE | DEPENDENCIA | CLAVE | CARRERA_DEPTO | ABREV | ESTATUS |
|---|---|---|---|---|---|---|---|---|
| 15 | ALFREDO FERNANDEZ GARCIA | 03/03/2006 | EXTERNO | DEPENDENCIA EXTERNA | 100100104 | EXTERNO | EXTERNO | ACTIVO |
| 9 | BRENDA ROSALVA MEZA TAMAYO | 03/03/2006 | EXTERNO | DEPENDENCIA EXTERNA | 100100101 | EXTERNO | EXTERNO | ACTIVO |
| 14 | CRISTINA LUNA PADILLA | 03/03/2006 | EXTERNO | FACULTAS DE ESTUDIOS SUPERIORES ZARAGOZA | 100100103 | EXTERNO | EXTERNO | ACTIVO |
| 23 | DANIELA DE ANDA GARCIA | 03/03/2006 | ESTUDIANTE | FACULTAD DE ESTUDIOS SUPERIORES CUAUTITLAN | 400020054 | ADMNISTRACIÓN | ADMON | ACTIVO |
| 17 | FELIX HERNANDEZ IBARRA | 03/03/2006 | INTERNO | FACULTAD DE ESTUDIOS SUPERIORES CUAUTITLAN | 121212 | ADMINISTRACIÓN | Administración | ACTIVO |
| 11 | FRANCISCO FERNANDEZ LOPEZ | 03/03/2006 | INTERNO | FACULTAD DE ESTUDIOS SUPERIORES CUAUTITLAN | 100100 | ADMINISTRACIÓN | Administración | ACTIVO |
| 10 | GABRIEL HERENDIRA FERNANDEZ | 03/03/2006 | EXTERNO | FACULTAD DE ESTUDIOS SUPERIORES ARAGON | 400020012 | EXTERNO | EXTERNO | ACTIVO |
| 7 | GABRIELA HERNANDEZ HEREDIA | 03/03/2006 | EXTERNO | CIUDAD UNIVERSITARIA | 400020051 | EXTERNO | EXTERNO | ACTIVO |
| 2 | ISABEL CRISTINA VAZQUEZ GUERRERA | 25/02/2006 | INTERNO | FACULTAD DE ESTUDIOS SUPERIORES CUAUTITLAN | 45789 | CONTABILIDAD | Contabilidad | ACTIVO |
| 1 | JORGE ALBERTO CORTES GAMIÑO | 25/02/2006 | ESTUDIANTE | FACULTAD DE ESTUDIOS SUPERIORES CUAUTITLAN | 400020053 | INFORMATICA | INFO | ACTIVO |
| 5 | JORGE ALBERTO CORTES GARCIA | 02/03/2006 | ESTUDIANTE | FACULTAD DE ESTUDIOS SUPERIORES CUAUTITLAN | 400029868 | ADMNISTRACIÓN | ADMON | ACTIVO |
| 6 | KARLA SANCHEZ BARRALES | 03/03/2006 | ESTUDIANTE | FACULTAD DE ESTUDIOS SUPERIORES CUAUTITLAN | 400029861 | ADMNISTRACIÓN | ADMON | ACTIVO |
| 19 | LAURA FERNANDEZ DE CEBALLOS | 03/03/2006 | EXTERNO | CIUDAD UNIVERSITARIA | 100100105 | EXTERNO | EXTERNO | ACTIVO |
| 20 | LUIS HUMBERTO GUZMAN RIVAS | 03/03/2006 | ESTUDIANTE | FACULTAD DE ESTUDIOS SUPERIORES CUAUTITLAN | 400029866 | VETERINARIA | MVZ | ACTIVO |

- Seleccionar la opción de Inactivos y oprimir el botón **Buscar**. La búsqueda muestra todas las coincidencias que encuentra con el dato proporcionado estas se ven reflejadas en una cuadricula. El número en la esquina superior derecha desplaza a otras páginas con los resultados obtenidos.


- No introducir ningún dato y oprimir el botón **Buscar.** La búsqueda muestra todos los usuarios registrados del sistema. El número en la esquina superior derecha desplaza a otras páginas con los resultados obtenidos.

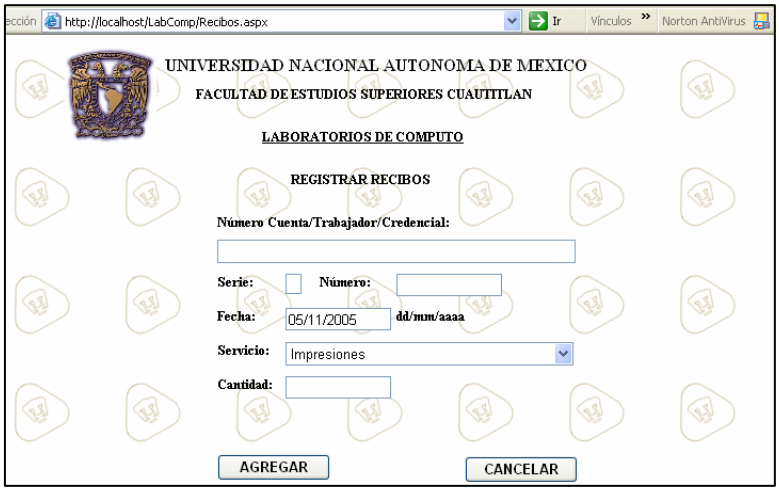


## 2. RECIBOS

Para registrar recibos al sistema es necesario ingresar al Menú Usuario y elegir el botón de Recibos.


**Página Menú Recibo (MenuRecibo.aspx):** Esta página nos permite Registrar, Modificar y Listar los recibos para los servicios activos del sistema (Impresiones, Escaneo, etc.).

**Página Registrar (Recibos.aspx):** En esta página se registran los recibos para los servicios activos del sistema.




Cuando un usuario entregue un recibo este será registrado de la siguiente manera:

- Introducir el Número de Cuenta (Estudiante), Número de Trabajador (Interno) ó el Número de Credencial (Externo).
- Introducir el Número de Serie y el Número marcado en el recibo.
- Capturar en Fecha la fecha del recibo esta debe ser así día/mes/año a cuatro cifras.
- Seleccionar el Servicio que ampara el recibo.
- Capturar en Cantidad el importe que ampara el recibo.
- Si los datos están completos y correctos se puede registrar el recibo oprimiendo el botón de Agregar (Regresará al Menú de Recibo), si desea detener este registro debe oprimir el botón de Cancelar se quedará en la misma página.


**MODIFICAR RECIBO (MenuRecibo.aspx):** Para modificar recibos es necesario ingresar al Menú Recibo y elegir el botón de Modificar.

**Página Modificar (ModificaRecibo.aspx):** En esta página se modifican los recibos para los servicios activos del sistema.


Para realizar una modificación es necesario capturar el número de Serie y Número del recibo. Si se desconocen estos datos es necesario ir a la opción de Listar recibos.

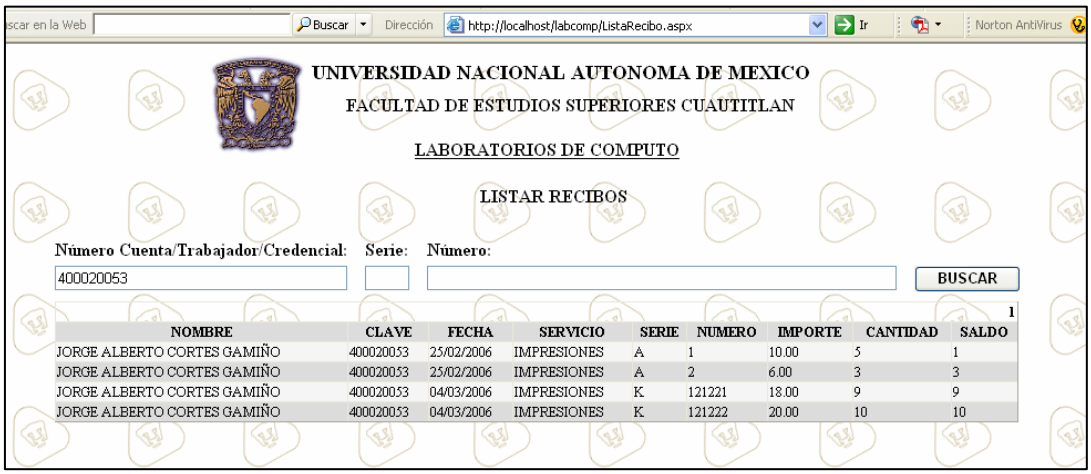
Al oprimir el botón de **Buscar.** La búsqueda muestra todas las coincidencias que encuentre con ese número. Hechas las modificaciones y dado que los datos están completos y correctos se puede modificar el recibo oprimiendo el botón de Guardar (Regresará al Menú de Recibo), si desea detener este registro debe oprimir el botón de Cancelar se quedará en la misma página.


**LISTAR RECIBO (MenuRecibo.aspx):** Para ver los recibos del sistema es necesario ingresar al Menú Recibo y elegir el botón de Listar.

**Página Listar (ListaRecibo.aspx):** En esta página se pueden ver los recibos de los servicios activos del sistema.


Para poder ver un recibo existen cuatro formas:

- Introducir el Número de Cuenta (Estudiante), Número de Trabajador (Interno) ó el Número de Credencial (Externo) y pulsar el botón **Buscar**. La búsqueda muestra todas las coincidencias que encuentre con ese número. Aparecerá una cuadricula con los datos del recibo en la parte superior derecha aparece un número de páginación ya que la cuadricula sólo muestra 15 resultados.



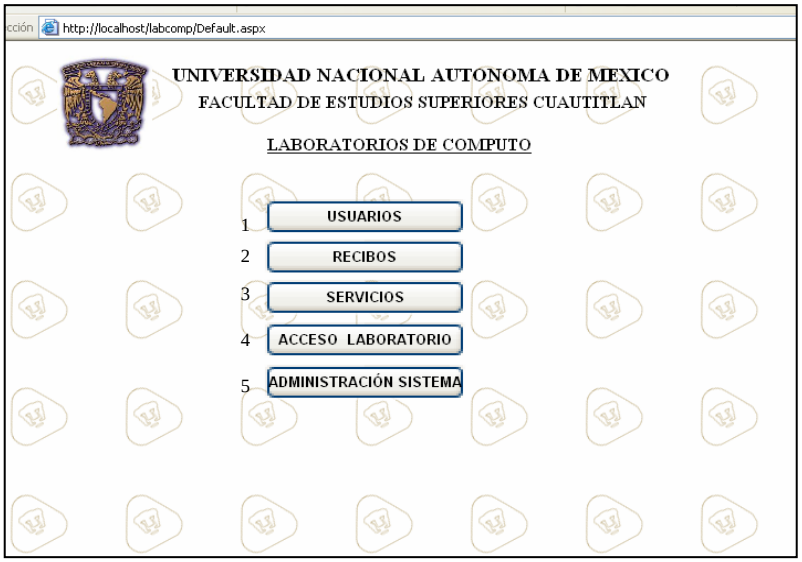
- Introducir sólo la letra de Serie del recibo y oprimir en botón **Buscar.** La búsqueda muestra todas las coincidencias que encuentre con ese dato. Aparecerá una cuadricula con los datos del recibo también aparece el número de páginación de la cuadricula.


- Introducir en Número del recibo y oprimir el botón de **Buscar.** La búsqueda muestra todas las coincidencias que encuentre con ese dato. Aparecerá una cuadricula con los datos del recibo y el número de páginación de la cuadricula.

- En el caso de no introducir ningún dato y oprimir el botón **Buscar.** La búsqueda muestra todos los recibos registrados del sistema. El número en la esquina superior derecha desplaza a otras páginas con los resultados obtenidos.


## 3. SERVICIOS

Para descargar los saldos de los recibos del sistema es necesario ingresar al Menú Usuario y elegir el botón de Servicios.



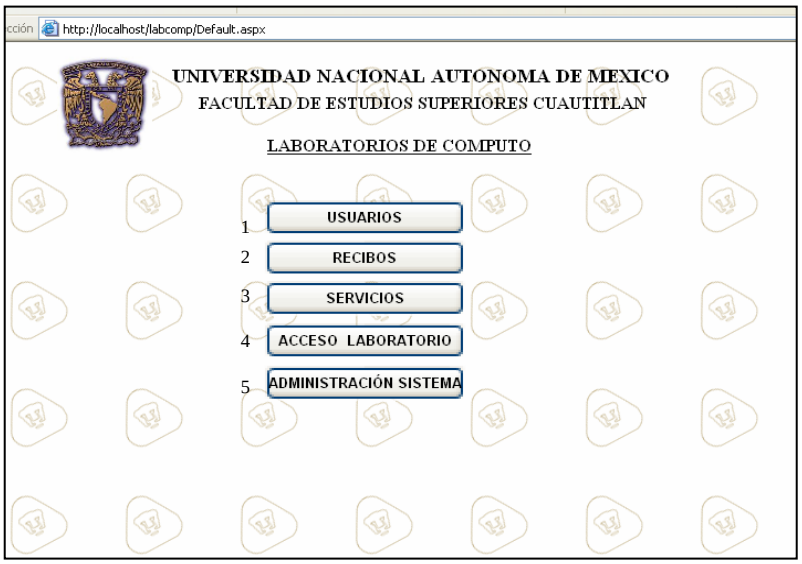
**Página Servicios (Servicios.aspx):** En esta página se descargan los recibos cuando se presta el servicio indicado por los mismos ejemplo: impresiones, escaneo, escaneo a color, etc. No se puede prestar el servicio cuando un usuario no tiene recibos registrados.


Para descargar un recibo se siguen los siguientes pasos:

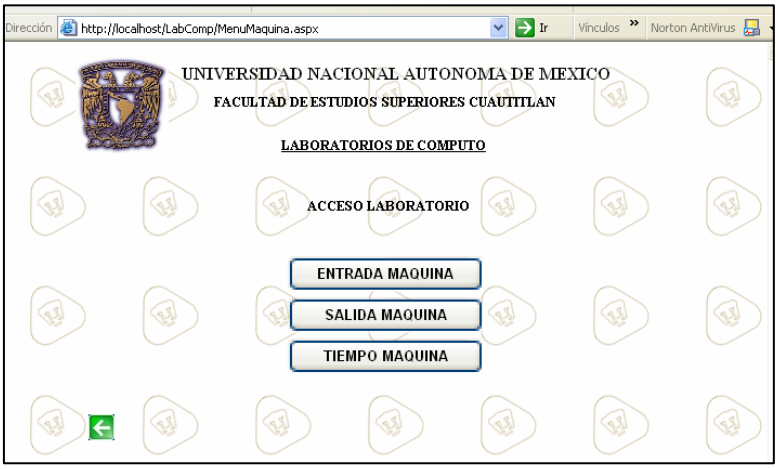
- Introducir el número de cuenta (Estudiante), número de trabajador (Interno) ó el número de credencial (Externo).
- Seleccionar el servicio que corresponde a lo que se desea realizar.
- Oprimir el botón **Buscar** se pueden presentar dos casos: Si el usuario tiene recibos con saldo aparecerá una cuadricula con los datos del recibo y su saldo de lo contrario aparecerá un mensaje señalando que el usuario no tiene recibos con saldo. Si el usuario tiene recibos aparece la cantidad real del servicio, lo único que se necesita es introducir la Cantidad a Realizar (descargar) del recibo o recibos y oprimir el botón **Descargar**. Regresa a la página de Menú Principal. Si se selecciona Cancelar te deja en la misma página.


## 4. ACCESO LABORATORIO

Para acceder a las instalaciones del laboratorio necesario ingresar al Menú Usuario y elegir el botón de Acceso Laboratorio.



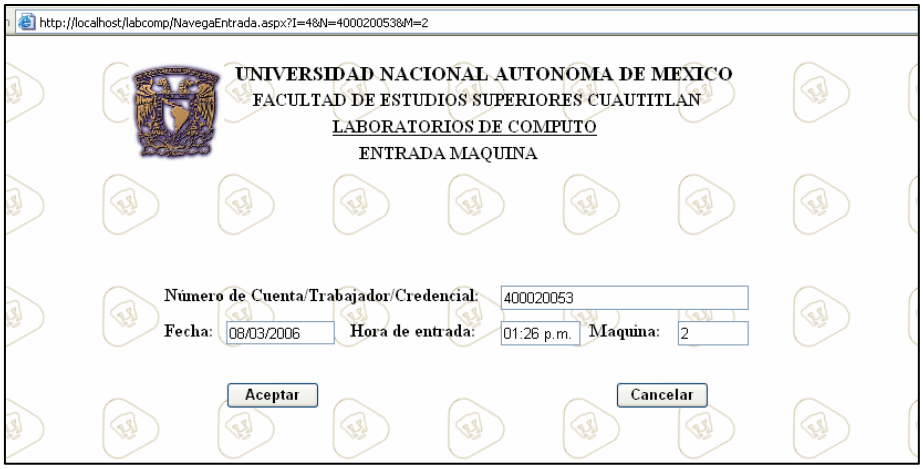
**Página Acceso Laboratorio (MenuMáquina.aspx):** Cuando un usuario va a acceder a las instalaciones del laboratorio, es necesario entrar a esta página donde se encuentran las tres opciones de Entrada Máquina, Salida Máquina y Tiempo Máquina las cuales se explican a continuación:



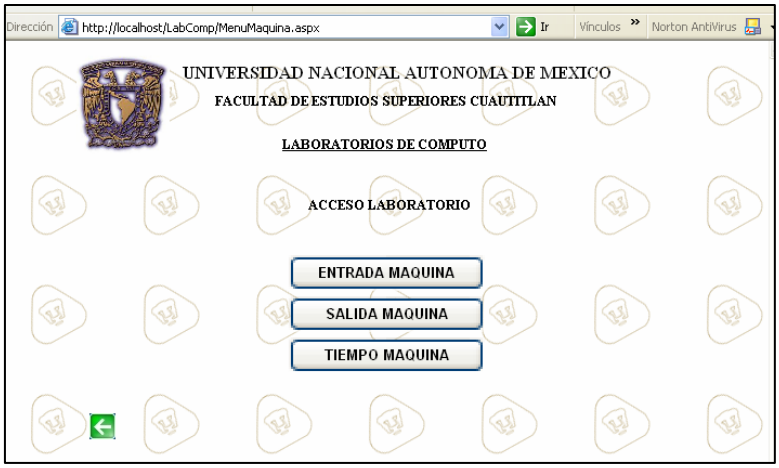

**Página Entrada Máquina (EntradaMaq.aspx):** En esta página se registra el momento en que una máquina es asignada a un usuario. El número de máquinas que se muestran en la cuadricula depende del laboratorio en que se este ingresando al sistema LABCOMP. El color de las imágenes varía dependiendo el estado de las máquinas del laboratorio. Máquinas de color azul aquellas que están desocupadas, de color rojo aquellas que están siendo utilizadas y de color negro aquellas que por algún motivo (mantenimiento, virus, etc.) están inhabilitadas o fuera de servicio.

Para ingresar al usuario hay que introducir su número de cuenta, trabajador ó credencial y seleccionar una imagen de la cuadricula inferior con el estado libre (imagen de color azul). Si se elige alguna de color rojo o negro el sistema no realiza ninguna acción.


**Página Navega Entrada (NavegaEntrada.aspx):** Después de haber seleccionado la imagen correspondiente a la máquina a ocupar. Se muestra la siguiente página proporcionando el detalle de la entrada. Si los datos de entrada son correctos se debe de oprimir el botón **Aceptar** nos regresara al la página de Entrada Máquina**.** En caso de que haya ocurrido un error de asignación de máquina se sugiere apretar el botón **Cancelar.**



**Salida Máquina**. Para liberar a las máquinas en el sistema es necesario ingresar a la página de Acceso Laboratorio y pulsar el botón Salida Máquina.



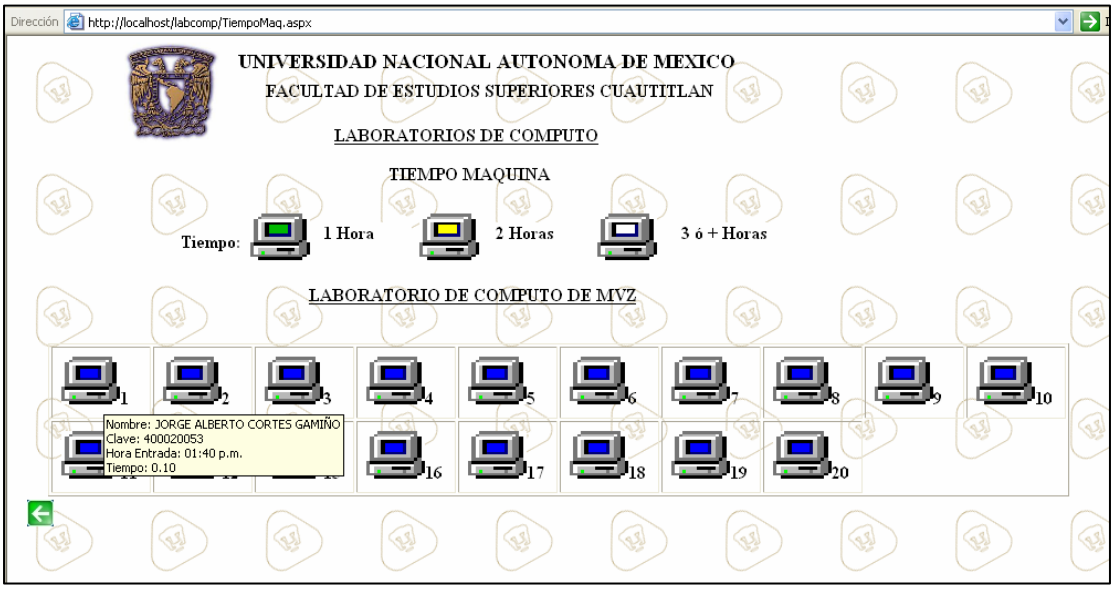

**Página Salida Máquina (SalidaMaq.aspx):** En esta página se registra el momento cuando una máquina es desocupada por el usuario. Para marcar la salida se selecciona la imagen con el estado ocupada (imagen de color rojo) que representa la máquina a desocupar. Al pasar el apuntador del mouse sobre dicha imagen despliega la información del usuario que la esta ocupando para verificar que efectivamente sea el usuario que va a salir del laboratorio.

**Página Navega Salida (NavegaSalida.aspx):** Después de haber seleccionado la imagen se muestra la página con el detalle de la salida. Si los datos de salida son correctos se debe de oprimir el botón **Aceptar** nos regresa ala página Salida Máquina. En caso de que haya ocurrido un error de salida máquina se puede apretar el botón **Cancelar** nos regresa a la página anterior**.**


**Tiempo Máquina**. Para conocer el tiempo que se han utilizado las máquinas en el laboratorio es necesario regresar ala página de Acceso Laboratorio y oprimir el botón Salida Máquina.

**Página Tiempo Máquina (TiempoMaq.aspx)**: Esta página sirve para que los encargados de dar acceso al laboratorio puedan administrar el tiempo de uso de las máquinas. Mostrando en una cuadricula imágenes que representan a las máquinas del laboratorio de la siguiente forma:

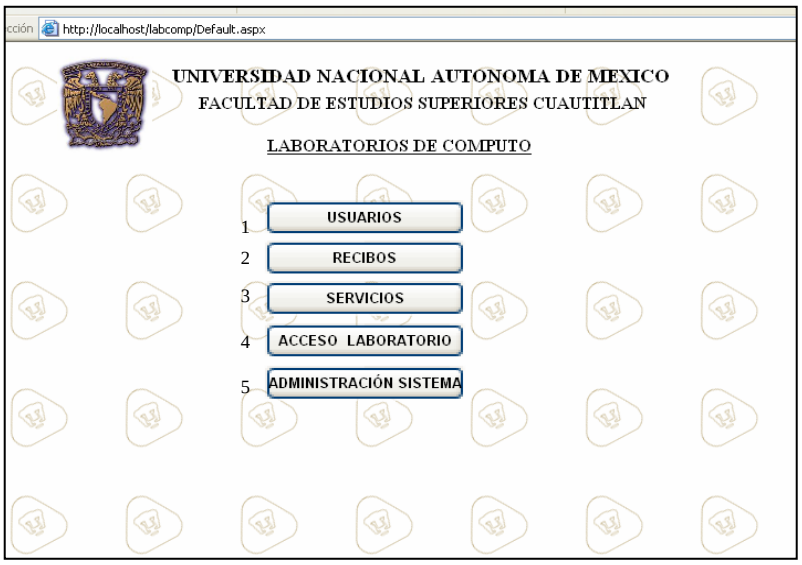
| Color | Tiempo |
|---|---|
| Negro | No disponibles |
| Azul | No asignadas o con menos de una hora |
| Verde | Una hora y menos de dos |
| Amarillo | Dos horas y menos de tres |
| Blanco | De tres a mas horas |




Al pasar el apuntador del mouse sobre alguna imagen, se muestra el detalle de quién es el usuario que la ocupa, así como su número de identificación, su hora de entrada y el tiempo que lleva dentro del laboratorio.

## 5. ADMINISTRACIÓN DEL SISTEMA

Para dar mantenimiento al sistema es necesario ingresar al Menú Usuario y elegir el botón de Administración Sistema.




**Pantalla Administración Sistema (Login.aspx):** Página restringida para la configuración del sistema. Sólo puede ser accesada por los coordinadores del laboratorio o aquellas personas que se les permita el acceso[19].


[19] Consultar el Capitulo 5 incisos 5.5.1 y 5.5.2 de este trabajo

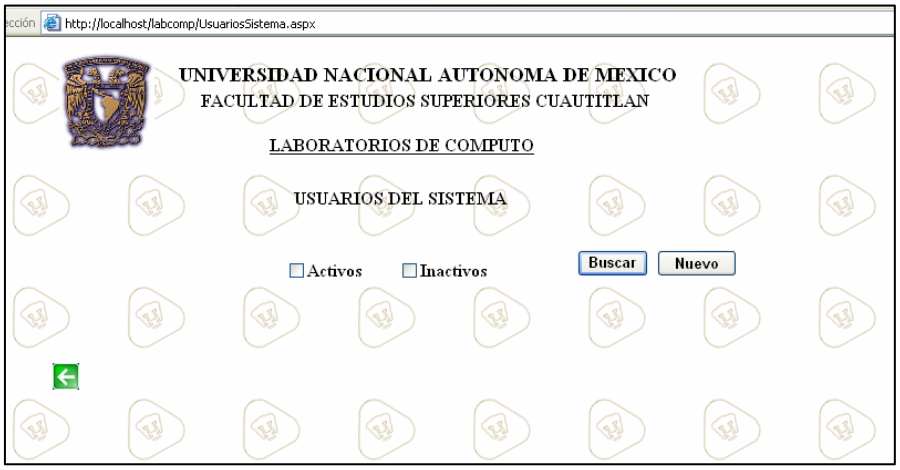
**Página Menú del Administrador (MenuAdministrador.aspx):** Esta página contiene todas las opciones que se pueden configurar del sistema. Lo cual afectara el comportamiento de las páginas explicadas anteriormente.


**Usuarios de Sistema**

Para acceder a esta opción es necesario oprimir el botón Usuarios de Sistema.

**Página Usuarios del Sistema (UsuariosSistema.aspx):** Sirve para agregar, actualizar ó inactivar a aquellas personas que puedan tener acceso a la administración del sistema LABCOMP.

Existen dos filtros Activos e Inactivos cuando no se seleccionada ninguno de los dos ó cuando se seleccionan los dos y se oprime el botón **Buscar.**




La cuadricula muestra todos los usuarios del sistema. Si sólo se selecciona Activos mostrara los activos y el mismo caso con Inactivos.

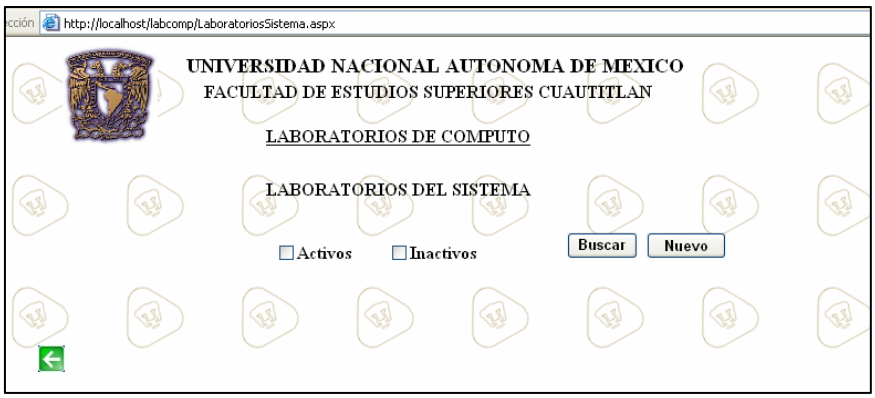
**Página Mantenimiento Usuarios (MantUsuarios.aspx):** Al dar clic a nuevo usuario ó a la flecha en **Ver** se desplegara la siguiente página para capturar o actualizar los datos del usuario. Donde se debe capturar el Nombre Completo, el Puesto dentro del laboratorio, seleccionar la Ubicación del Laboratorio en el que esta prestando sus servicios y el Nombre de usuario. La casilla Activo si esta seleccionada quiere decir que el usuario estará activo y podrá entrar a configurar el sistema y si no esta seleccionada quiere decir que el usuario no podrá entrar al sistema. Para todo cambio es necesario oprimir el botón **Guardar**. Nos regresa a la página Usuarios del Sistema. En caso de que haya ocurrido un error de datos se puede apretar el botón **Cancelar.**

**Laboratorios del Sistema**

Para acceder a esta opción es necesario ingresar al Menú del Administrador y oprimir el botón Laboratorios.


**Página Laboratorios del Sistema (LaboratoriosSistema.aspx):** Sirve para agregar, actualizar ó inactivar los diferentes laboratorios que existen dentro de la Facultad de Estudios Superiores Cuautitlán en el sistema LABCOMP.

Existen dos filtros Activos e Inactivos cuando no se seleccionada ninguno de los dos ó cuando se seleccionan los dos y se oprime el botón **Buscar.**



La cuadricula muestra todos los laboratorios. Si sólo se selecciona Activos mostrara los activos y el mismo caso con Inactivos.

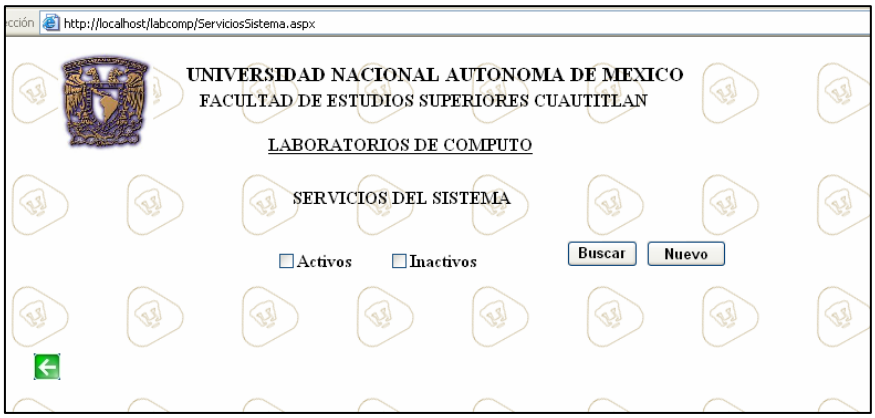

**Página de Mantenimiento de Laboratorios (MantLaboratorios.aspx):** Al dar clic a nuevo laboratorio ó a la flecha **Ver** se desplegara la siguiente página para capturar o actualizar los datos del laboratorio. Donde se debe capturar el Nombre, la Abreviatura (Nombre corto).La casilla Activo si esta seleccionada quiere decir que el laboratorio estará activo y si no esta seleccionada quiere decir que el laboratorio no estará accesible en el sistema. Para todo cambio es necesario oprimir el botón **Guardar**. Nos regresa a la página Laboratorios del Sistema. En caso de que haya ocurrido un error de datos se puede apretar el botón **Cancelar.**

**Servicios del Sistema**

Para acceder a esta opción es necesario ingresar al Menú del Administrador y oprimir el botón Servicios.


**Página Servicios del Sistema (ServiciosSistema.aspx):** Sirve para agregar, actualizar ó inactivar servicios en el sistema LABCOMP.

Existen dos filtros Activos e Inactivos cuando no se seleccionada ninguno de los dos ó cuando se seleccionan los dos y se oprime el botón **Buscar.**



La cuadricula muestra todos los servicios. Si sólo se selecciona Activos mostrará los activos y el mismo caso con Inactivos.


**Página Mantenimiento Servicios (MantServicios.aspx):** Al dar clic a nuevo servicio ó a la flecha en **Ver** se desplegara la siguiente página para capturar o actualizar los datos del servicio. Donde se debe capturar el Nombre, el Costo. La casilla Activo si esta seleccionada quiere decir que el servicio estará activo y si no esta seleccionada quiere decir que el servicio no estará accesible en el sistema. Para todo cambio es necesario oprimir el botón **Guardar**. Nos regresa a la página Servicios del Sistema. En caso de que haya ocurrido un error se puede apretar el botón **Cancelar.**

**Departamentos**

Para acceder a esta opción es necesario ingresar al Menú del Administrador y oprimir el botón Departamentos.


**Página Departamentos del Sistema (DepartamentosSistema.aspx):** Sirve para agregar, actualizar ó inactivar departamentos en el sistema LABCOMP.

Existen dos filtros Activos e Inactivos cuando no se seleccionada ninguno de los dos ó cuando se seleccionan los dos y se oprime el botón **Buscar.**

La cuadricula muestra todos los departamentos. Si sólo se selecciona Activos mostrara los activos y el mismo caso con Inactivos.

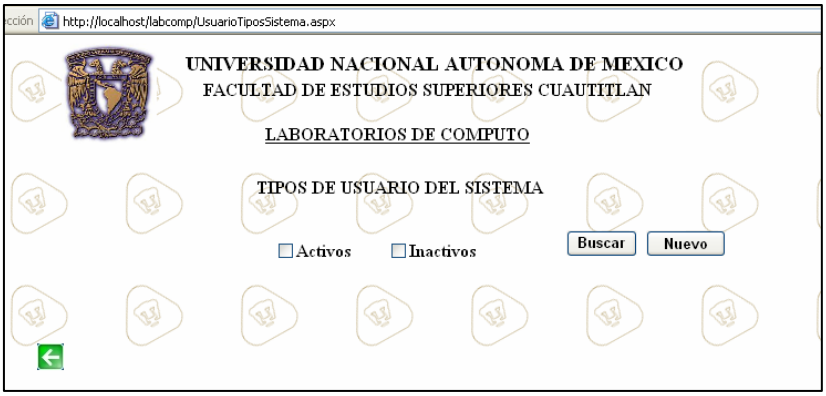

**Página Mantenimiento Departamentos (MantDepartamentos.aspx):** Al dar clic a nuevo departamento ó a la flecha en **Ver** se desplegara la siguiente página para capturar o actualizar los datos del departamento. Donde se debe capturar el Nombre. La casilla Activo si esta seleccionada quiere decir que el departamento estará activo y si no esta seleccionada quiere decir que el departamento no estará accesible en el sistema. Para todo cambio es necesario oprimir el botón **Guardar**. Nos regresa a la página Departamentos del Sistema. En caso de que haya ocurrido un error se puede apretar el botón **Cancelar.**

**Tipo de Usuarios**

Para acceder a esta opción es necesario ingresar al Menú del Administrador y oprimir el botón Tipo de Usuarios.


**Página Tipos de Usuario del Sistema (UsuarioTiposSistema.aspx):** Sirve para agregar, actualizar ó inactivar los tipos de usuario en el sistema LABCOMP.

Existen dos filtros Activos e Inactivos cuando no se seleccionada ninguno de los dos ó cuando se seleccionan los dos y se oprime el botón **Buscar.**



La cuadricula muestra todos los tipos de usuario. Si sólo se selecciona Activos mostrara los activos y el mismo caso con Inactivos.

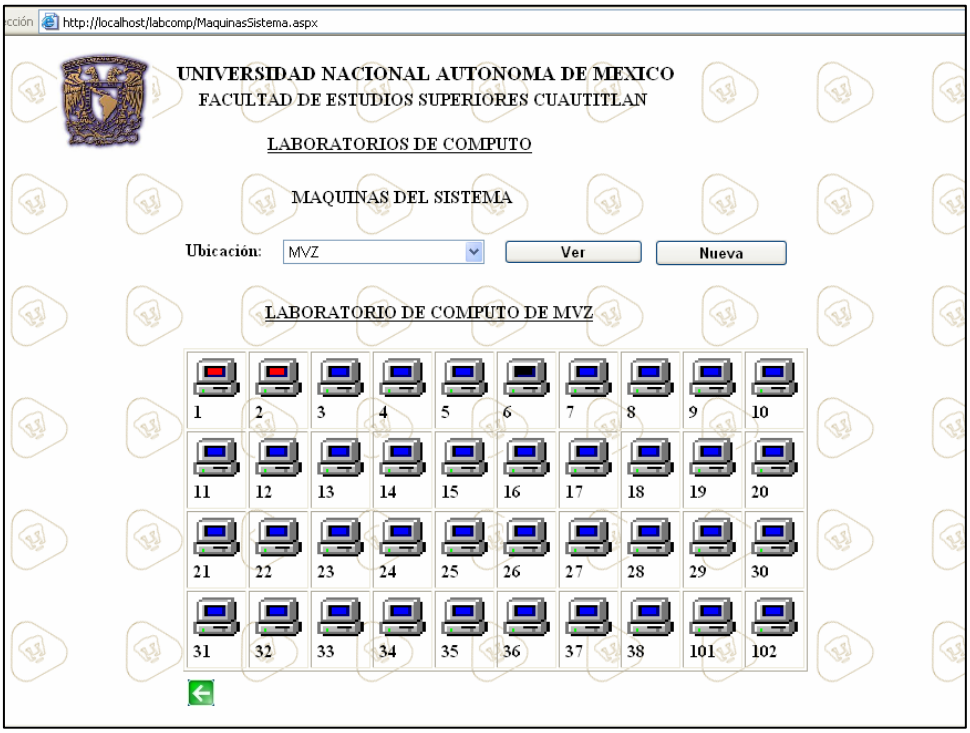

**Página de Mantenimiento de Tipos de Usuario (MantUsuarioTipos.aspx):** Al dar clic a nuevo tipo de usuario ó a la flecha en **Ver** se desplegara la siguiente página para capturar o actualizar los datos del tipo de usuario. Donde se debe capturar el Nombre. La casilla Activo si esta seleccionada quiere decir que el tipo de usuario estará activo y si no esta seleccionada quiere decir que el tipo de usuario no estará accesible en el sistema. Para todo cambio es necesario oprimir el botón **Guardar**. Nos regresa a la página Tipos de Usuario del Sistema. En caso de que haya ocurrido un error se puede apretar el botón **Cancelar.**

**Máquinas**

Para acceder a esta opción es necesario ingresar al Menú del Administrador y oprimir el botón Máquinas.


**Página Máquinas del Sistema (MáquinasSistema.aspx):** Sirve para agregar, actualizar ó inactivar máquinas en el sistema LABCOMP.

Debe seleccionar la Ubicación del laboratorio Ejemplo: MVZ  y oprimir el botón **Ver** mostrara el total de máquinas del laboratorio.




**Página Mantenimiento de Máquinas del Sistema (MantMáquinas.aspx):** Al dar clic a nueva máquina ó a una imagen de la cuadricula se desplegara la siguiente página para capturar o actualizar los datos de la máquina. En donde se tendrá que capturar el Número de la máquina, seleccionar su Ubicación (Laboratorio en la que se encuentra), Información (Observaciones acerca de la máquina), Dirección IP, Seleccionada la casilla Activa quiere decir que la máquina estará activa y si no esta seleccionada quiere decir que la máquina no estará accesible en el sistema para su uso.

El campo de Dirección IP sirve para que una máquina pueda usar la opción de Registrar Recibos, Acceso a laboratorio y ver el Control de Ingresos en el sistema LABCOMP. Si no tiene dirección IP estas acciones no podrán ser usadas en dicha máquina. Si se selecciona la casilla Mostrar es para que en la cuadricula de máquinas de Acceso a Laboratorios (Vea Inciso 4) no se muestre dicha máquina para su préstamo.

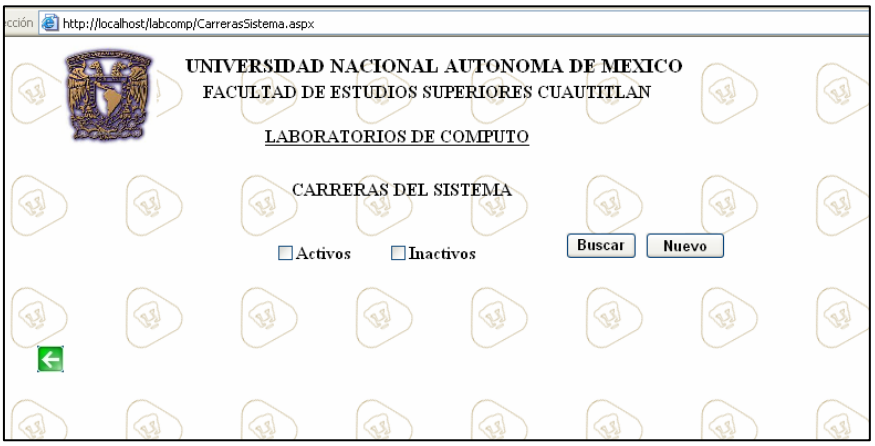
Para todo cambio es necesario oprimir el botón **Guardar**. Nos regresa a la página Máquinas del Sistema. En caso de que haya ocurrido un error se puede apretar el botón **Cancelar** regresa a la página anterior**.**

**Carreras**

Para acceder a esta opción es necesario ingresar al Menú del Administrador y oprimir el botón Carreras.

**Página Carreras del Sistema (CarrerasSistema.aspx):** Sirve para agregar, actualizar ó inactivar las carreras que se imparten dentro de la Facultad de Estudios Superiores Cuautitlán o de cualquier otra dependencia en caso de que sea necesario en el sistema LABCOMP.

Existen dos filtros Activos e Inactivos cuando no se seleccionada ninguno de los dos ó cuando se seleccionan los dos y se oprime el botón **Buscar.**




La cuadricula muestra todas las carreras. Si sólo se selecciona Activos mostrara los activos y el mismo caso con Inactivos.

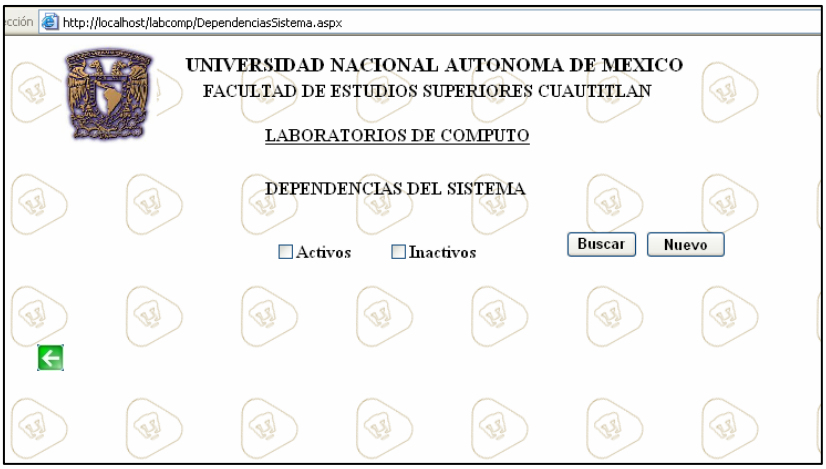
**Página Mantenimiento Carreras (MantCarreras.aspx):** Al dar clic a nueva carrera ó a la flecha en **Ver** se desplegara la siguiente página para capturar o actualizar los datos de la carrera. Donde se debe capturar el Nombre, la Abreviatura (Nombre corto). Seleccionar la casilla Activo quiere decir que la carrera estará activa y si no esta seleccionada quiere decir que la carrera no estará accesible en el sistema. Para todo cambio es necesario oprimir el botón **Guardar**. Nos regresa a la página Carreras del Sistema. En caso de que haya ocurrido un error se puede apretar el botón **Cancelar** regresa a la página anterior**.**


## Dependencias

Para acceder a esta opción es necesario ingresar al Menú del Administrador y oprimir el botón Dependencias.

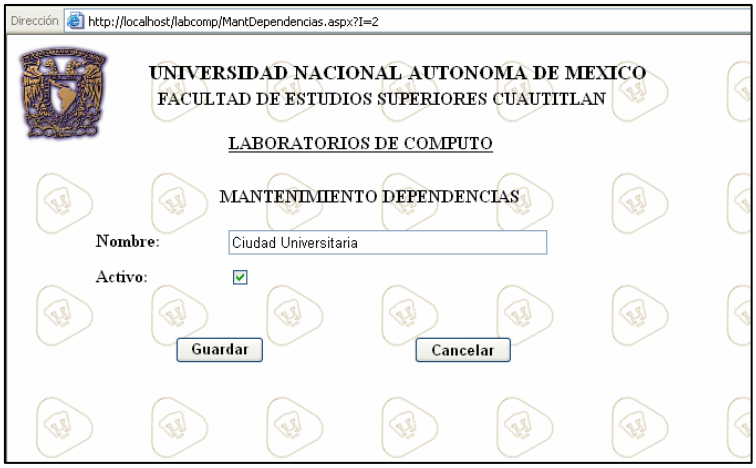
**Página Dependencias del Sistema (DependenciasSistema.aspx):** Sirve para agregar, actualizar ó inactivar las dependencias que conforman la Universidad Nacional Autónoma de México en el sistema LABCOMP.

Existen dos filtros Activos e Inactivos cuando no se seleccionada ninguno de los dos ó cuando se seleccionan los dos y se oprime el botón **Buscar.**




La cuadricula muestra todas las dependencias. Si sólo se selecciona Activos mostrara las activas y el mismo caso con Inactivos.

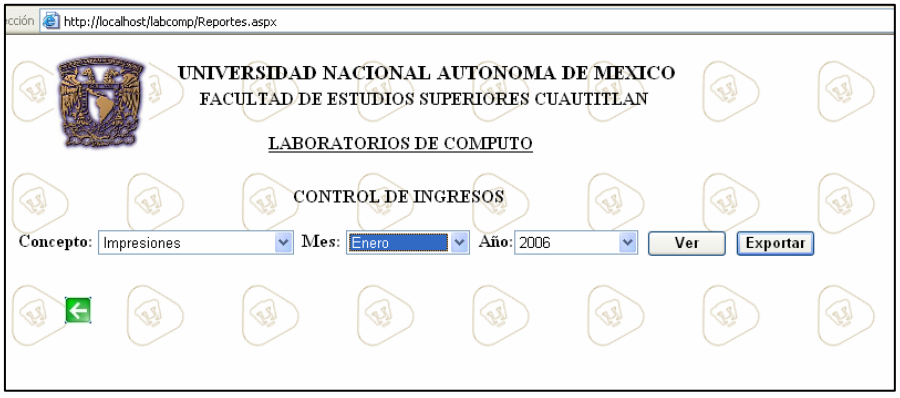
**Página Mantenimiento Dependencias (MantDependencias.aspx):** Al dar clic a nueva dependencia ó a la flecha en **Ver** se desplegara la siguiente página para capturar o actualizar los datos de la dependencia. Donde se debe capturar el Nombre de la dependencia. Seleccionar la casilla Activo quiere decir que la dependencia estará activa y si no esta seleccionada quiere decir que la dependencia no estará accesible en el sistema. Para todo cambio es necesario oprimir el botón **Guardar**. Nos regresa a la página Dependencias del Sistema. En caso de que haya ocurrido un error se puede apretar el botón **Cancelar** regresa a la página anterior**.**




## Control de Ingresos

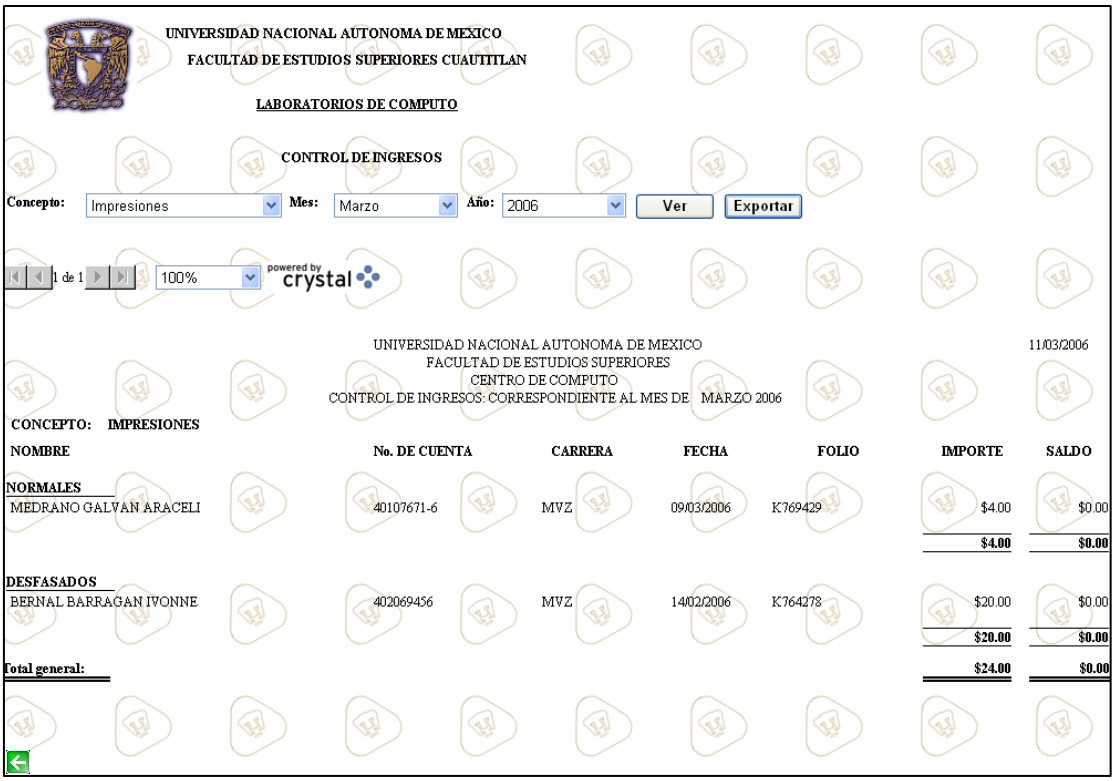
Para acceder a esta opción es necesario regresar al Menú del Administrador y pulsar el botón Control de Ingresos. El control de ingresos se lleva con el registro de todos recibos del sistema (ver punto 2 de este capitulo). Es importante llevar el control debido a que los coordinadores de cada laboratorio tienen que enviar un reporte mensual al Centro de Cómputo con los totales de los recibos ingresados.


**Página Control de Ingresos (Reportes.aspx):** Esta página sirve para emitir un reporte del control de ingresos de los recibos de cada laboratorio por mes. Haciendo una distinción entre recibos que se expidieron y se capturaron en el mes del reporte de aquellos que se expidieron en meses anteriores y se capturaron en el mes del reporte.



Para poder visualizar el reporte se llevan acabo los siguientes pasos:

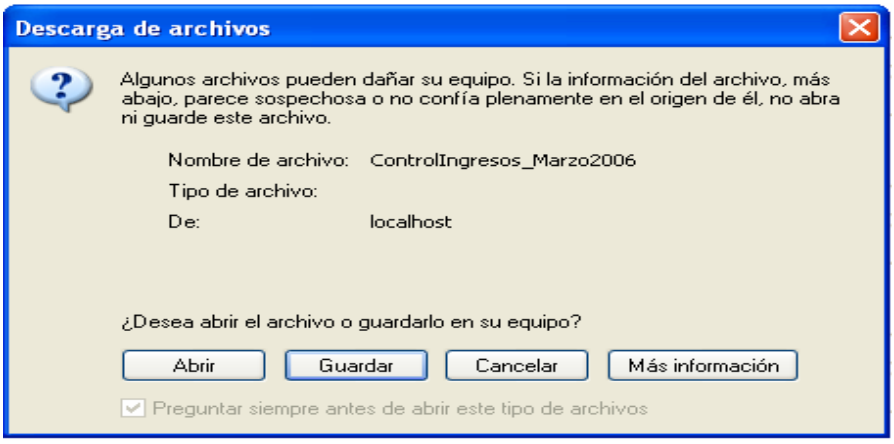
- Se selecciona el concepto (servicio) del cual se quiere obtener el reporte.
- Se selecciona el mes y año.
- Se oprime el botón **Ver**.
- Si desea guardar esta información en un archivo de Excel sólo es necesario oprimir el botón de Exportar.




Al seleccionar el botón de Exportar aparece un cuadro que le da cuatro opciones (Las tres pantallas siguientes son del sistema operativo):

- Debe elegir Guardar para crear el archivo de Excel[20].

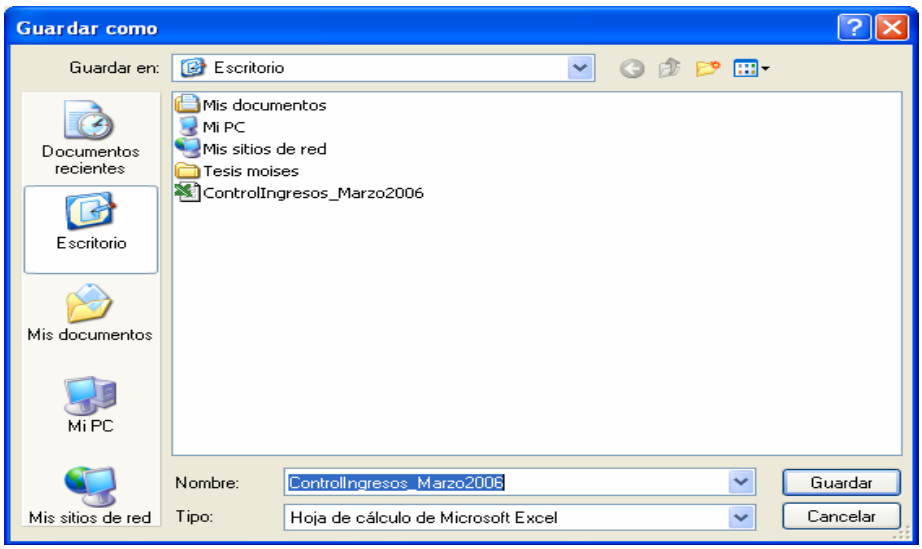
[20] El reporte se exporta a Excel debido a que así es solicitado por el Centro de Computo de la Facultad de Estudios Superiores Cuautitlan.



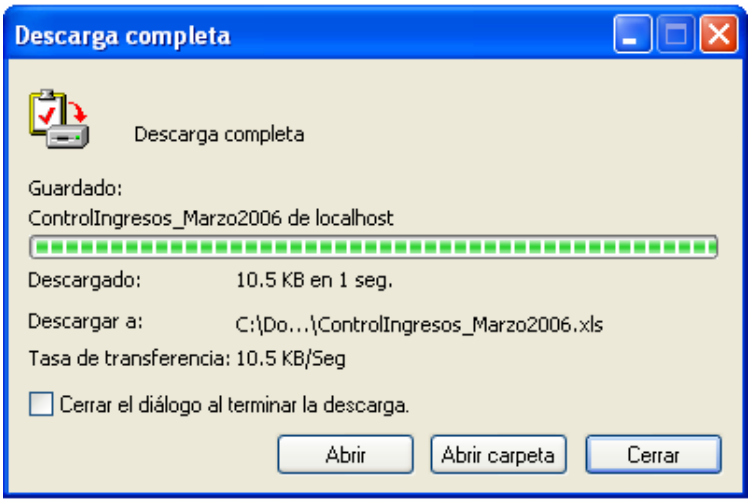

- Si elige Cancelar detendrá la creación del archivo en Excel.
- No debe elegir Abrir ya que aun no se ha creado en archivo de Excel.
- Si elige Mas información abre automáticamente la ayuda de Windows.

Cuando comienza a guardar el reporte en Excel aparece un recuadro que le solicita:

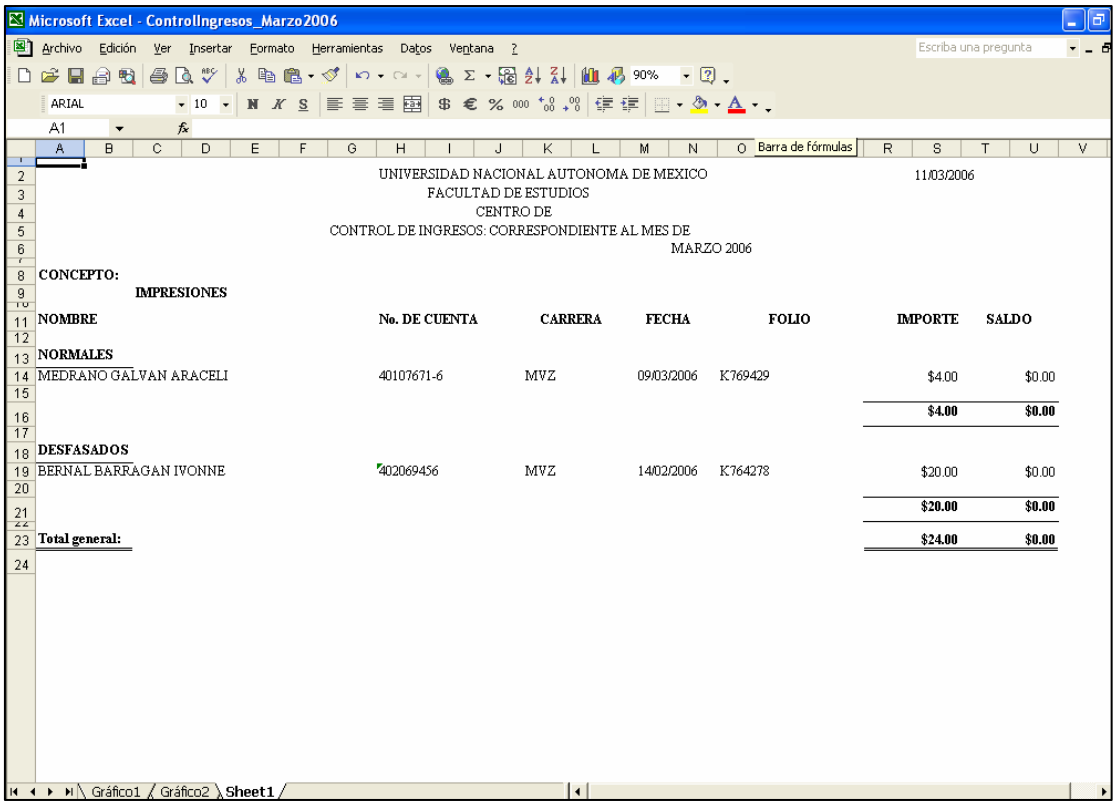
- La ubicación, esta es el lugar en el sistema donde desea guardar el archivo.
- El nombre del reporte lo crea automáticamente comienza con ControlIngresos_mes del reporte y el año del reporte. Si desea cambiar el nombre sólo es necesario renombrarlo de otra forma.



Le aparece un cuadro que indica que el archivo fue creado si desea Abrir el archivo sólo tiene que elegir esa opción.




Por ultimo puede ver el archivo de Excel recién creado. La siguiente imagen es del programa Microsoft Office Excel.



## 7.3 Guía de Instalación de Labcomp

Esta guía lo conduce por los pasos para instalar el sistema LABCOMP en su computadora.

Es importante seguir las instrucciones paso a paso.

### 7.3.1 Disco de Instalación

El disco de instalación de LABCOMP es entregado junto con el manual de usuario del sistema[21].

Contenido del disco:

1) Base de Datos: Esta carpeta contiene los archivos para instalar la base de datos en el programa Microsoft SQL Server Express versión 2005.

2) LabComp: Aquí se encuentran todos los archivos que conforman el sistema LABCOMP. Entre estos están las páginas diseñadas en ASP.NET, una carpeta de imágenes, una carpeta de reportes y una carpeta con dos ensamblados.

A partir del inciso 7.3.3 de este capitulo se describe como instalar el contenido del disco.

### 7.3.2 Prerrequisitos

Antes de la instalación, asegúrese de que su computadora presenta los siguientes requisitos:

**HARDWARE:**

- Intel ® o compatible Pentium 166 MHz o superior
- 64 MB mínimo en RAM

[21] El manual de usuario viene en formato de Microsoft Office Word (*.doc)

- 50 MB de espacio disponible en disco
- Unidad de CD-ROM

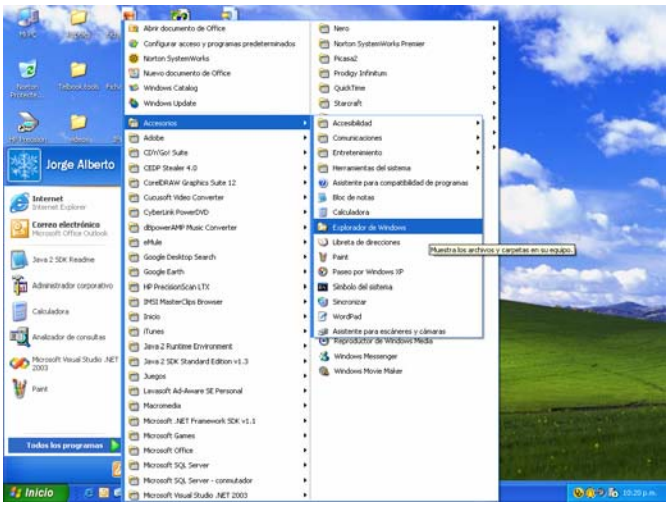
**SOFTWARE (Tener instalados)[22]:**

- Microsoft SQL Server Express versión 2005
- Microsoft .NET Framework SDK v2.0
- Internet Explorer 6 Service Pack 1 ó superior
- Visual Web Developer version 2005 Express

### 7.3.3 Instalación de la Base de Datos

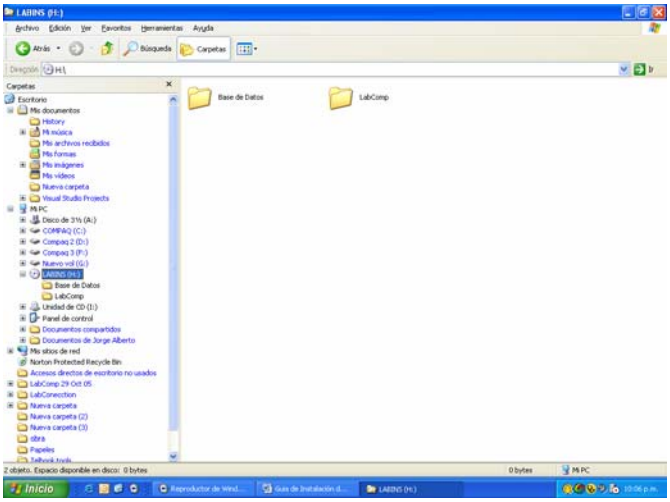
1) Insertar el disco de instalación de LABCOMP en la unidad de CD-ROM.

2) Abrir el Explorador de Windows. Dar clic a Inicio en la barra de tareas -> Todos los programas -> Accesorios -> Explorador de Windows.

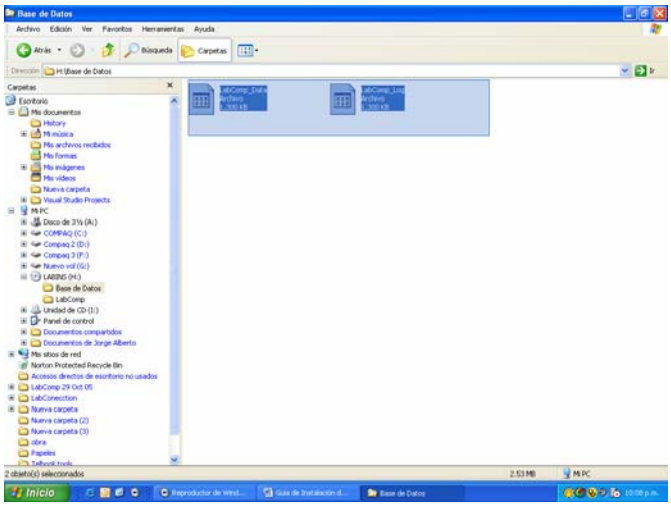



[22] En caso de no tenerlos instalados en el servidor se pueden descargar de la siguiente dirección: **http://www.microsoft.com/spanish/msdn/vstudio/express/VWD/default.mspx**

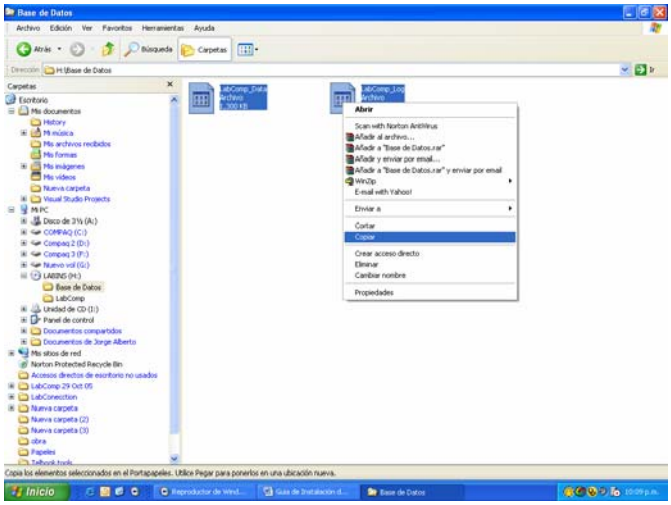
3) En el Explorador de Windows siga la siguiente ruta Mi PC y seleccionar la unidad de CD-ROM con el nombre LABINS. Se muestran dos carpetas BASE DE DATOS y LABCOMP.



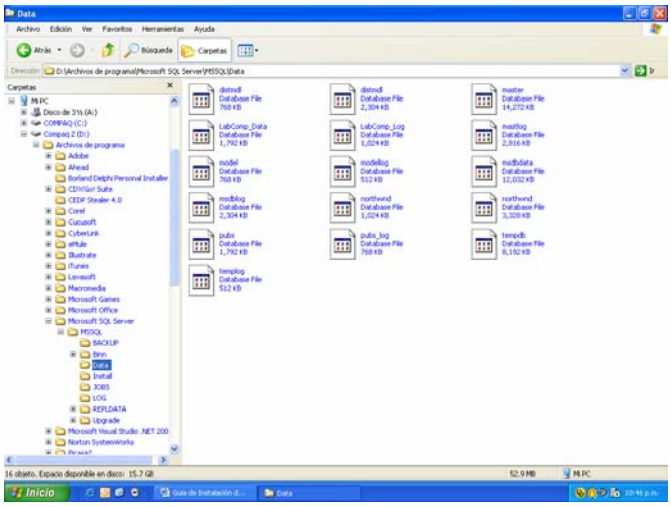
4) Dar doble clic a la carpeta BASE DE DATOS. Seleccionar los archivos LabComp_Data y LabComp_Log con el apuntador del mouse.



5) Dar clic con el botón derecho sobre alguno de los archivos y seleccionar la opción copiar del menú que aparece.

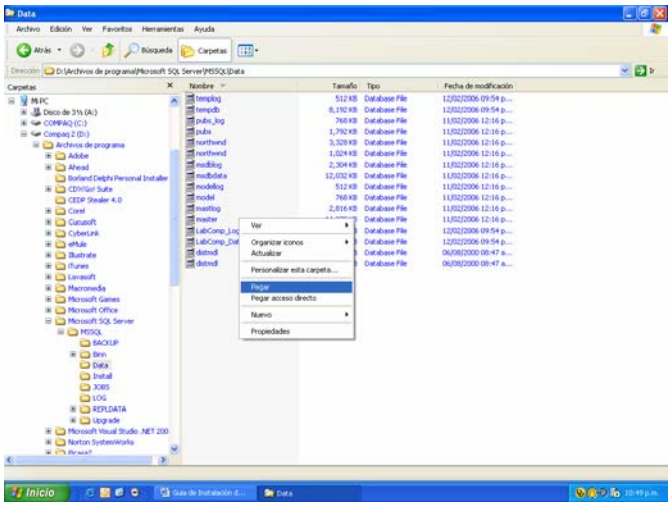


6) Entrar a la siguiente ruta: unidad de disco [23] -> Archivos de Programa -> Microsoft SQL Server -> MSSQL -> Data.

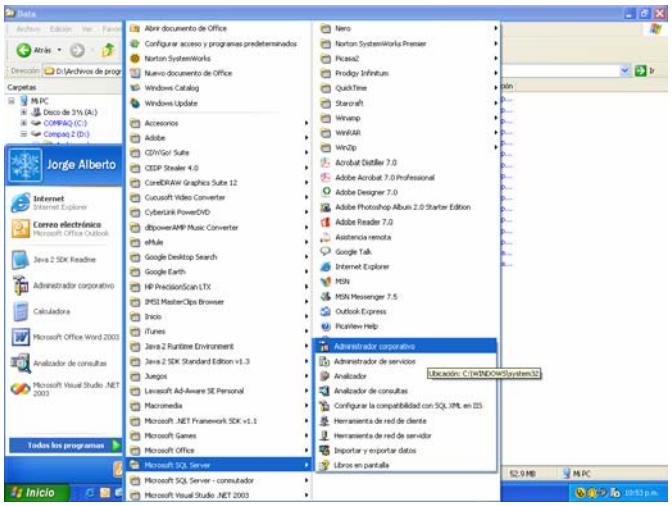


[23] La unidad puede variar dependiendo la computadora. Por lo general esta representada por la letra C.

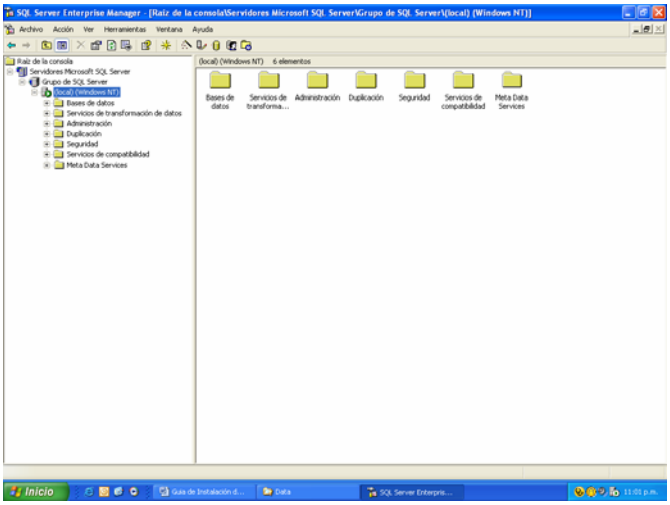
7) Dar clic al botón derecho del mouse y seleccionar la opción pegar del menú que aparece. Después de esta acción los archivos LabComp_MDF y LabComp_Log deben de aparecer en la ventana (De lo contrario volver al punto 3 de este inciso).



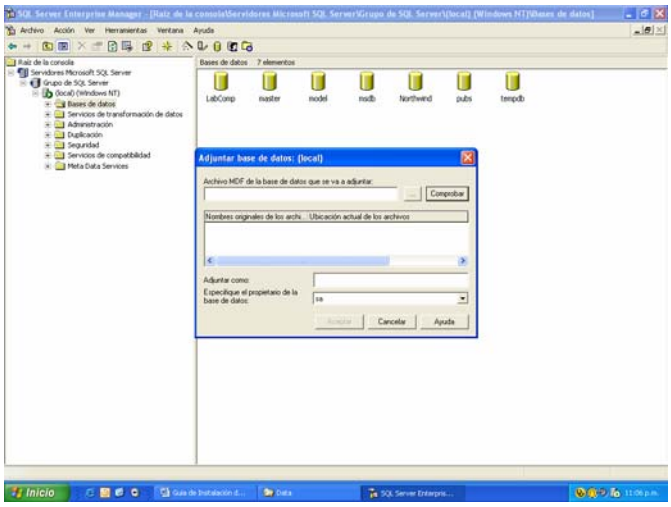

8) Abrir el Administrador Corporativo de SQL. Dar clic a Inicio en la barra de tareas -> Todos los programas -> Microsoft SQL Server -> Administrador Corporativo.



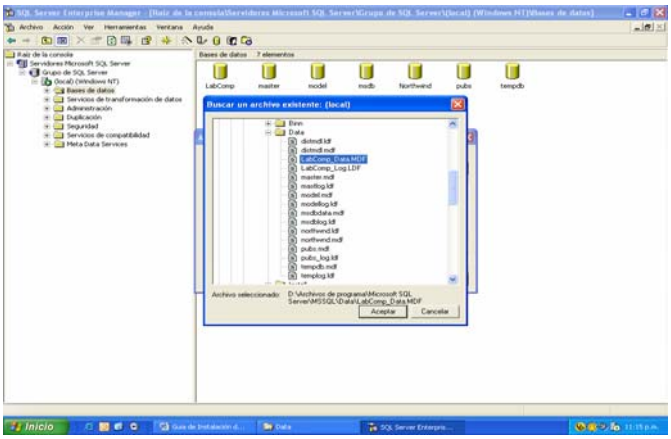
9) En el Administrador Corporativo Servidores Microsoft SQL Server -> Grupo de SQL Server -> (local)(Windows NT)



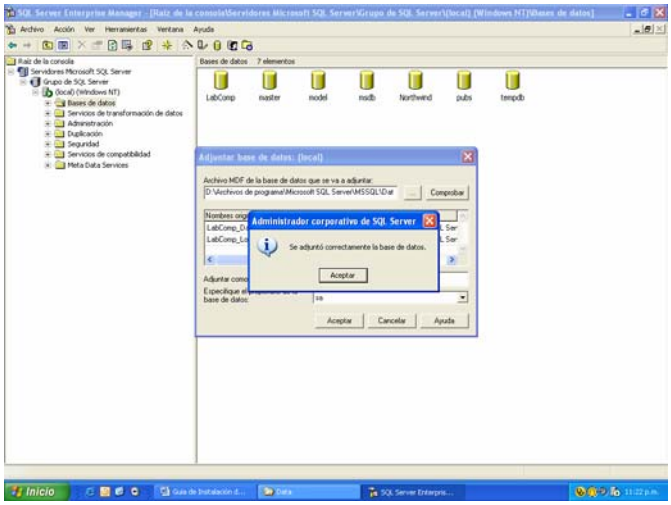
10) Dar clic con el botón derecho del mouse sobre Base de Datos y seleccionar la opción Todas las tareas y luego adjuntar base de datos. En la ventana que aparece de clic en el botón con los tres puntos.



11) Aparecerá una ventana con las unidades de disco que se encuentran en la computadora. Debe explorar las unidades y buscar la ruta donde se pegaron los archivos de la base de datos (Véase puntos 6 y 7 de este inciso). Seleccionar el archivo LabComp_Data.MDF y de clic en el botón Aceptar.



12) En la ventana del punto 11 de este inciso aparecerán los datos de la base que se va a adjuntar. De clic en el botón Aceptar. Si la base de datos se adjunta correctamente aparecerá una ventana de confirmación con el mensaje “Se adjuntó correctamente la base de datos” de clic en Aceptar. En caso de error volver al punto 10 de este inciso.



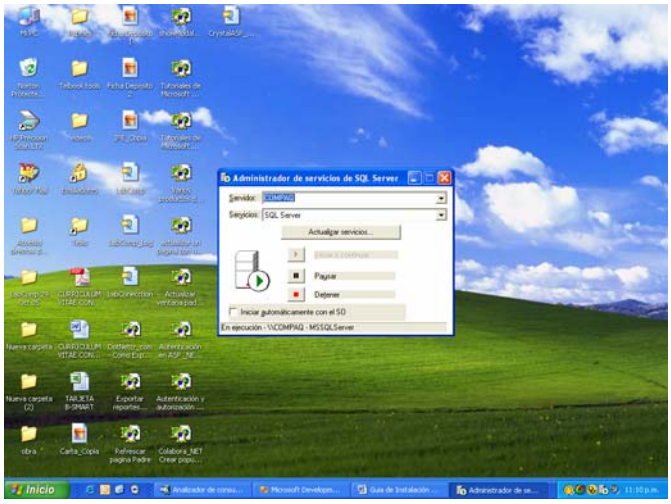
14) Después de haber completado los puntos anteriores la base de datos estará instalada correctamente en su computadora.

### 7.3.4 Configuración de la Cadena de Conexión

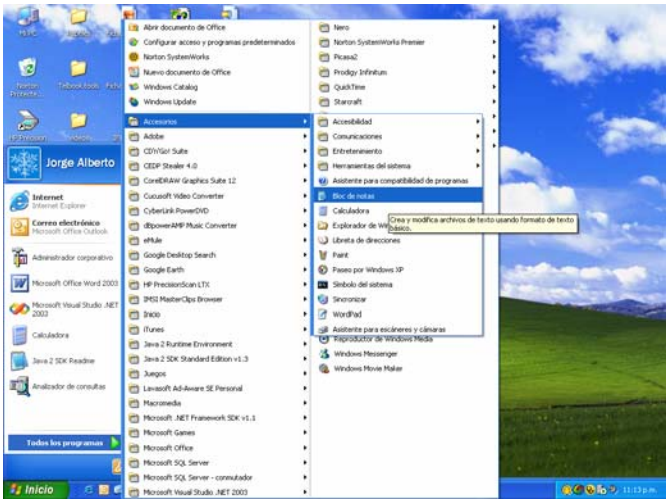
La configuración de la cadena de conexión es un punto importante. De esto depende que el sistema LABCOMP pueda accesar a la base de datos que se instalo anteriormente (en el inciso 7.3.3 de este capitulo). Esta configuración sólo se realiza una vez mientras el nombre del servidor (depende el servidor), el nombre de usuario (sa) y contraseña de SQL no cambien (labcomp).

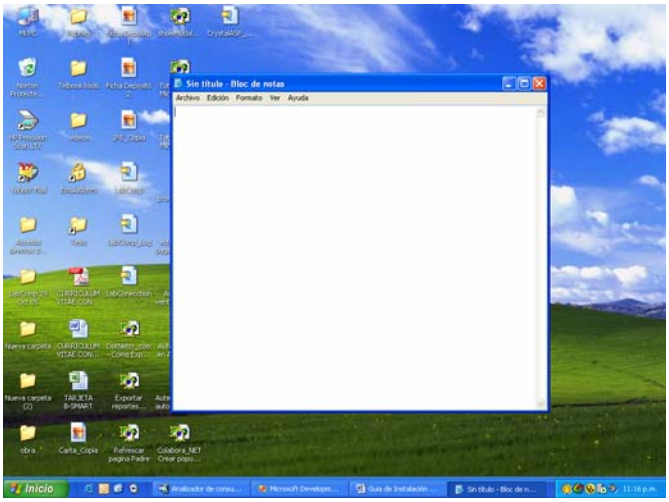
1) Abrir el Administrador de servicios de SQL. Dar doble clic a la figura que se encuentra en la esquina inferior derecha de la barra de tareas de Windows.

2) Copiar el nombre que aparece dentro del cuadro de texto de la ventana del Administrador de servicios (En este caso **COMPAQ**). Después de esto cierre la ventana.



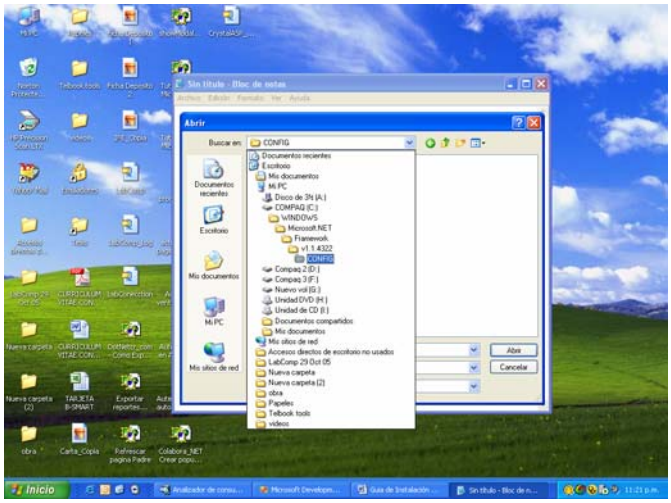

3) Abra el Bloc de Notas o cualquier editor de texto. Dar clic a Inicio en la barra de tareas -> Todos los programas -> Accesorios -> Bloc de notas.





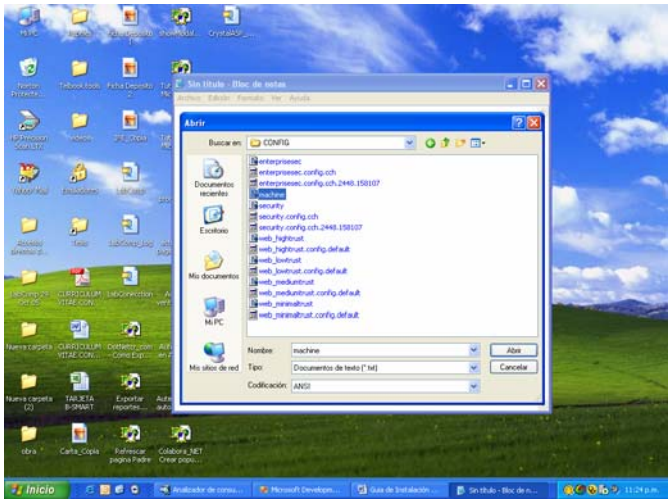
4) Dar clic en Archivo -> Abrir. En la ventana que aparece busque la siguiente ruta:

Unidad de disco [24] -> Windows -> Microsoft.Net -> Framework -> v1.1.4322 -> Config



[24] La unidad puede variar dependiendo la computadora. Por lo general esta representada por la letra C.

5) En el cuadro de texto Nombre escribir ***.config** y dar clic en el botón Abrir para que aparezca una lista de archivos en el recuadro superior. Seleccionar el archivo con el nombre **machine** vuelva a pulsar el botón Abrir.




6) Debajo del texto **</configSections>** (línea 45) introduzca el siguiente texto reemplazando COMPAQ por el nombre de el punto 2 de este inciso:

```
<appSettings>
    <add key="cnn" value="Data Source=COMPAQ;Initial Catalog=LabComp;User
Id=sa;Password=labcomp"/>
    <add key="Servidor" value="COMPAQ"/>
    <add key="BaseDatos" value="LabComp"/>
    <add key="Usuario" value="sa"/>
    <add key="Password" value="labcomp"/>
</appSettings>
```

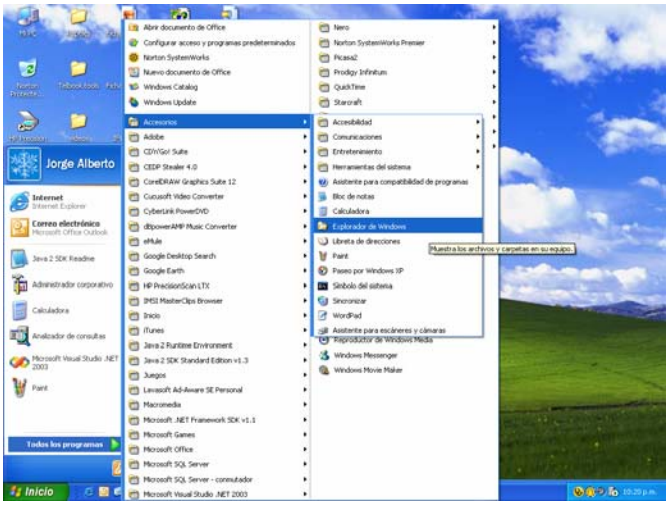
**NOTA: El usuario y contraseña representados por sa y labcomp son establecidos en el momento que se instala SQL en su computadora.**

7) Guardar los cambios y cerrar el Bloc de Notas o el editor de texto utilizado.

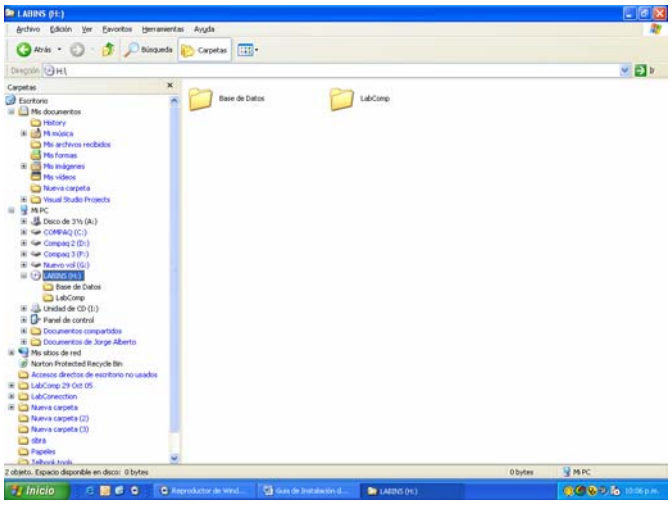
### 7.3.5 Instalación de Labcomp

Insertar el disco de instalación de LABCOMP en la unidad de CD-ROM.

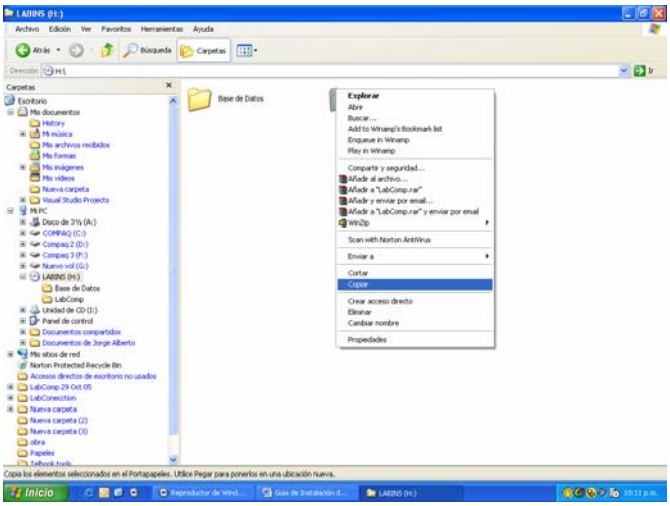
1) Abrir el Explorador de Windows. De clic a Inicio en la barra de tareas -> Todos los programas -> Accesorios -> Explorador de Windows.



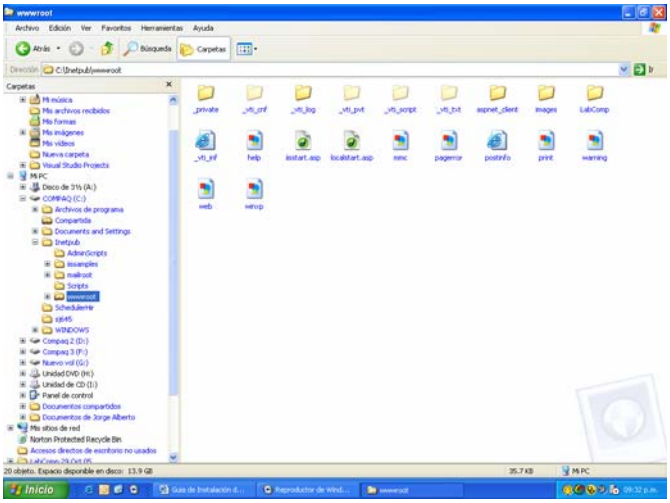

2) En el Explorador de Windows explorar Mi PC y seleccionar la unidad de CD-ROM con el nombre LABINS. Se mostraran dos carpetas BASE DE DATOS y LABCOMP.



3) Dar clic con el botón derecho sobre la carpeta LABCOMP y seleccionar la opción copiar del menú que aparece.

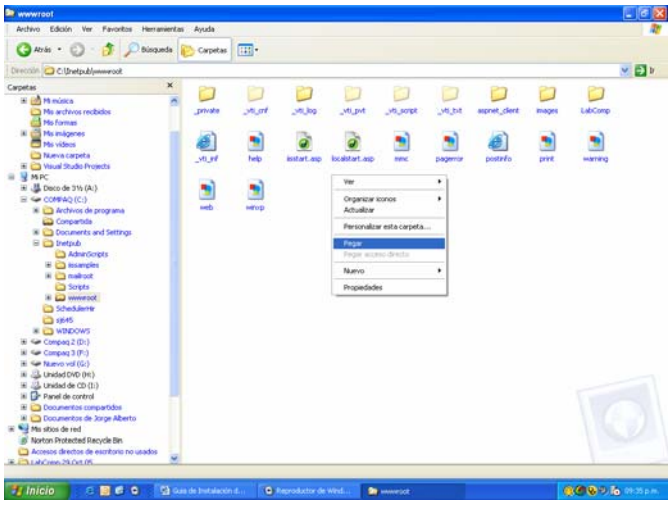


4) Explorar la unidad de disco [25] -> Inetpub -> Documents And Settings -> wwwroot

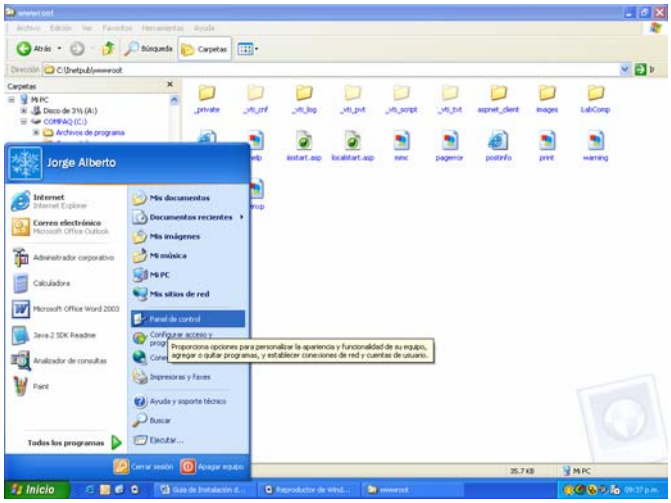


[25] La unidad puede variar dependiendo la computadora. Por lo general esta representada por la letra C.

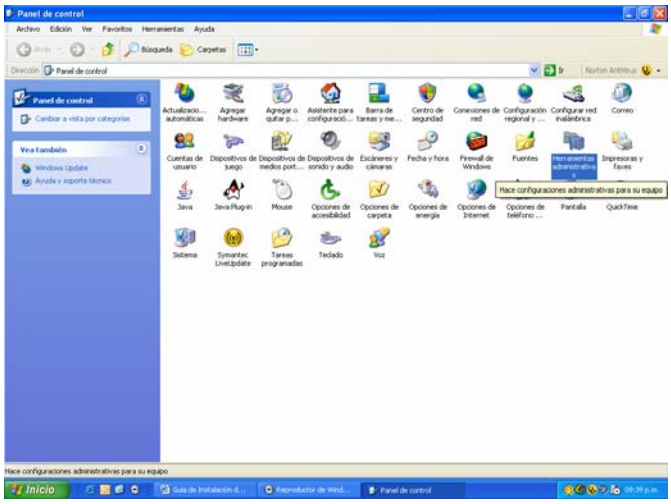
5) Dar clic al botón derecho del mouse y seleccionar la opción pegar del menú que aparece. Después de esta acción la carpeta LabComp deben de aparecer en la ventana (De lo contrario volver al punto 3 de este inciso).



6) Dar clic al botón de Inicio en la barra de tareas y seleccionar Panel de Control.

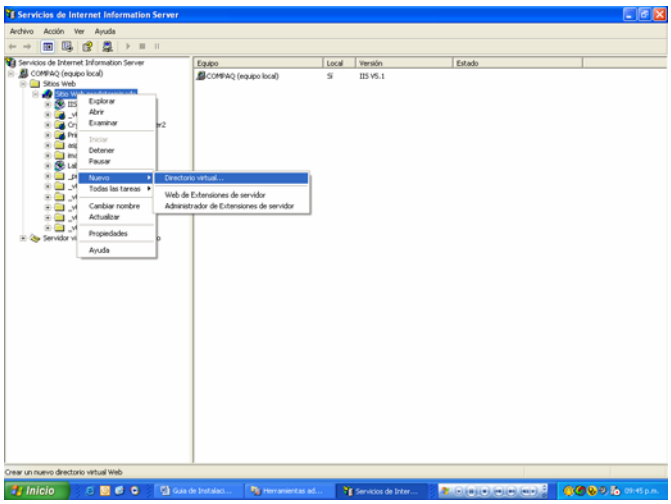


7) En el panel de control dar doble clic a Herramientas Administrativas.

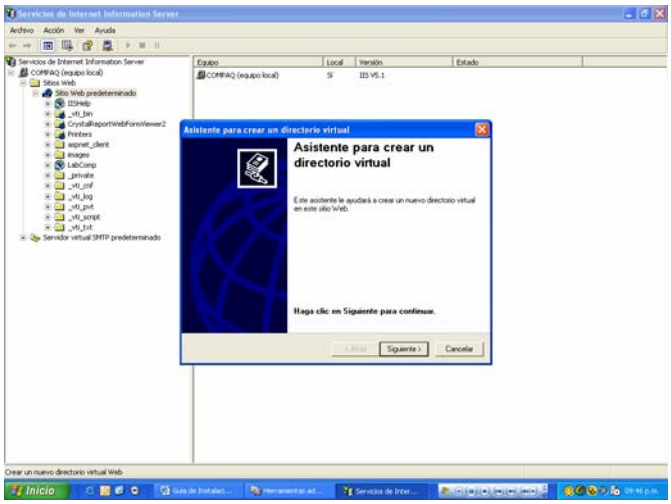



8) En Herramientas Administrativas dar doble clic a Servicios de Internet Information Server.

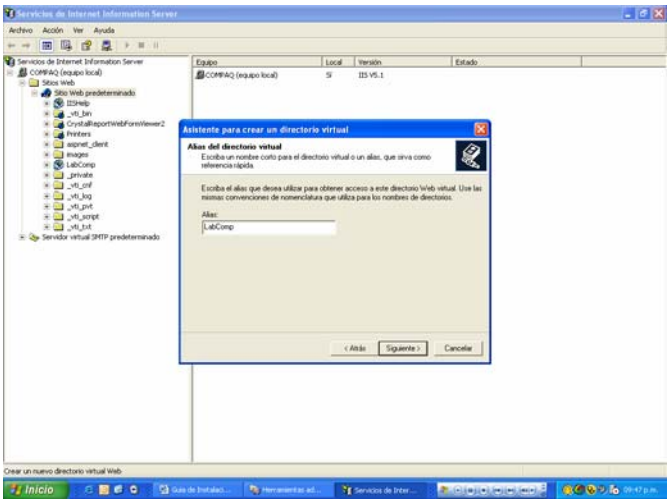
9) En la ventana que aparece de Servicios de Internet Information Server explorar (equipo local) -> Sitios Web -> Sitio Web Predeterminado (**Si ya existe un directorio con el nombre LabComp vaya al punto 23 de este inciso**) .Dar clic derecho a Sitio Web Predeterminado y seleccione la opción de Nuevo -> Directorio Virtual.



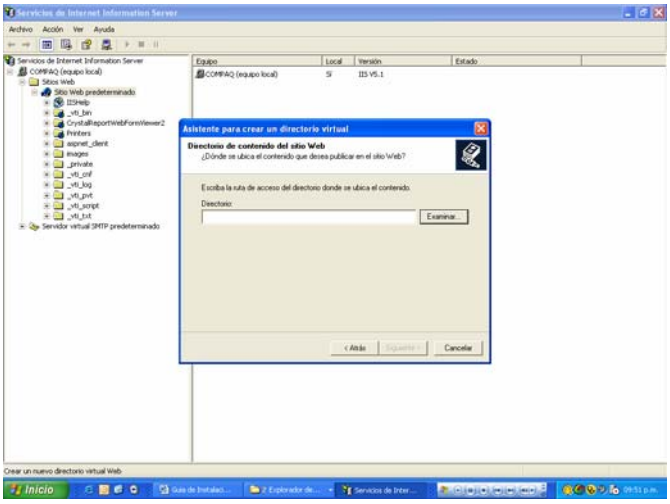

10) Dar clic al botón Siguiente.



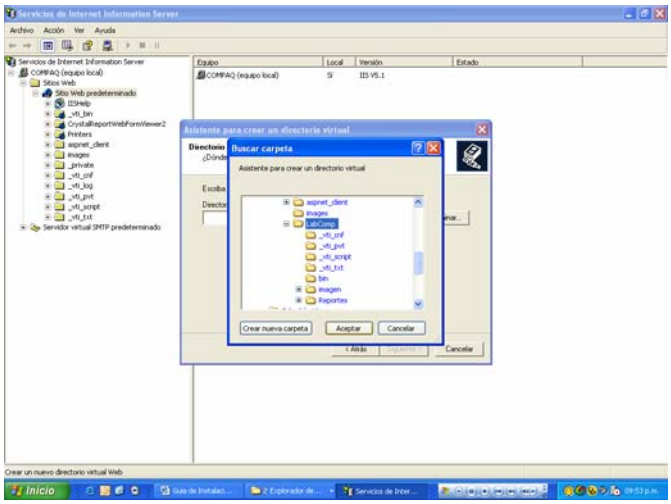
11) En el cuadro de Texto escribir el nombre **LabComp**.



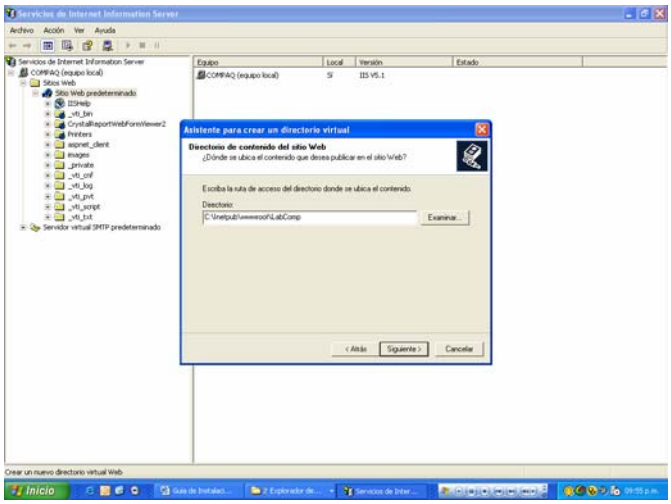

12) Dar clic al botón Examinar**.**



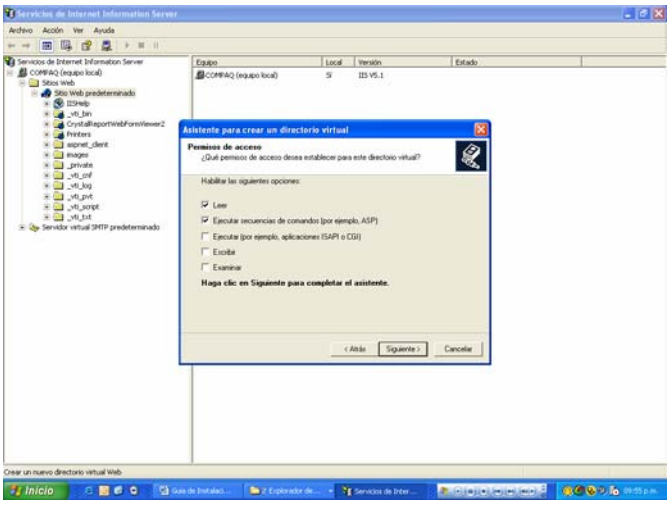

13) En la ventana que aparece, seleccionar la ruta donde se copio la carpeta LabComp (Véase punto 5 de este inciso). Y dar clic en el botón Aceptar.



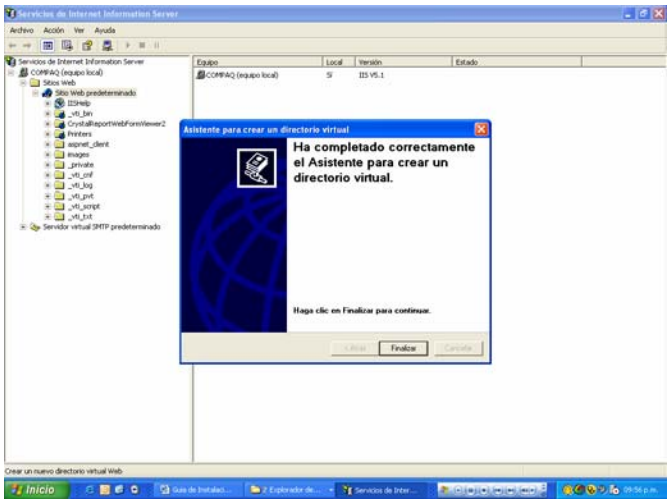
14) En el campo de texto directorio aparecerá la dirección donde copiamos la carpeta LabComp. Dar clic en el botón Siguiente.



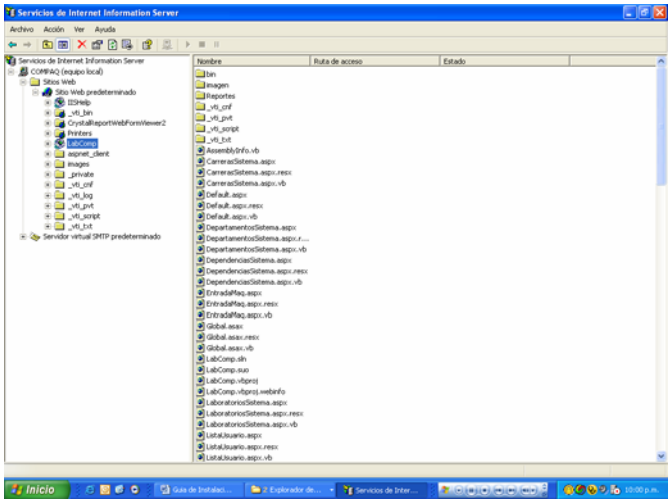
15) En la siguiente ventana dar clic en Siguiente.



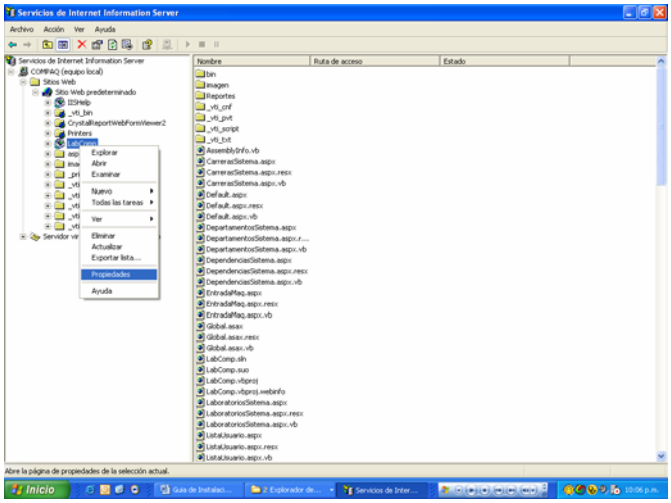

16) Dar clic en Finalizar.



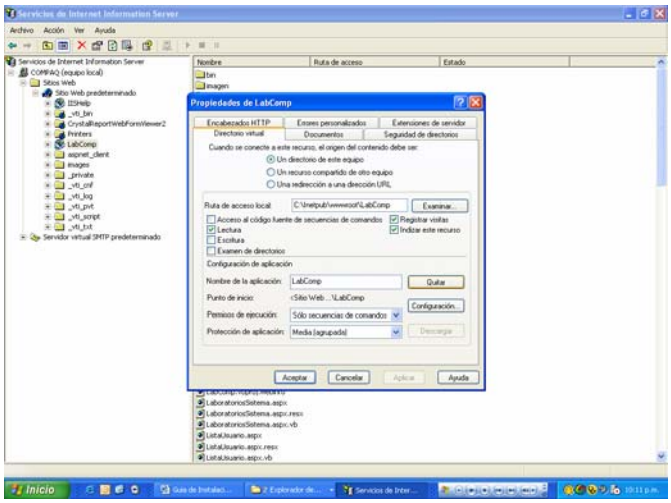

17) Debajo de Sitios de Web Predeterminado aparecerá una carpeta con el nombre de LabComp (De lo contrario volver al punto 10 de este inciso).



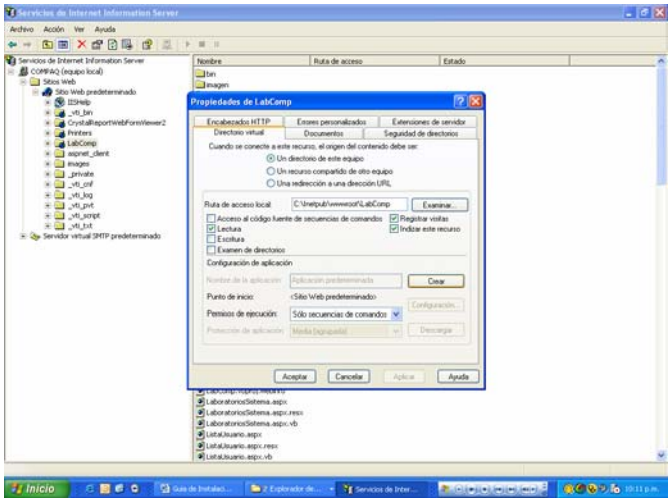
18) Dar un clic con el botón derecho sobre el directorio recién creado y seleccionar la opción Propiedades.



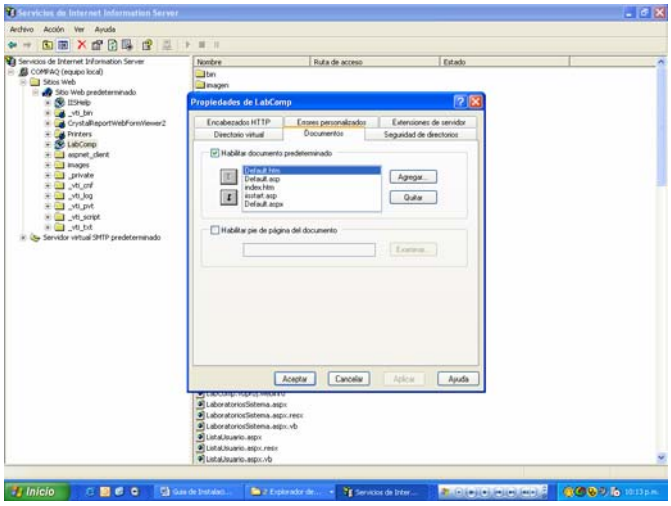
19) La ventana que aparece contiene 6 pestañas. En la primera “Directorio Virtual” Dar clic en el botón Quitar y posteriormente dar clic en el botón Crear (Este texto aparece en el mismo botón después de haberlo presionado).




En caso de que cuando aparezca la ventana el botón no tenga el texto Quitar, presionar el botón con el texto Crear.

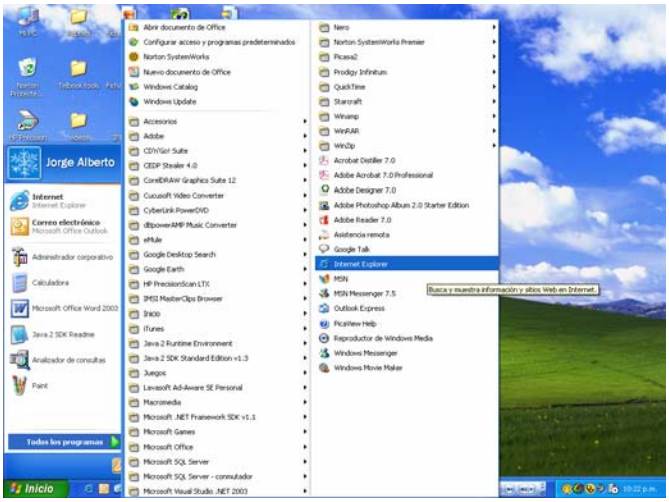


20) Pasar a la pestaña “Documentos” y borrar todos los documentos excepto Default.aspx. Para borrar un documento seleccione de la lista uno de ellos y a continuación dar clic el botón Quitar.

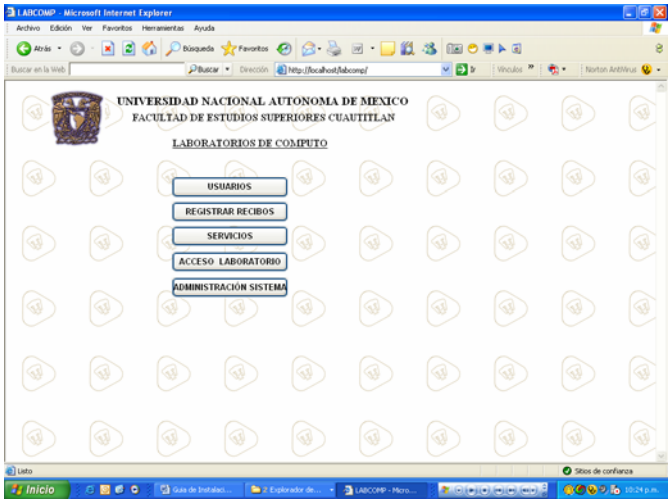



21) Para terminar dar clic en Aceptar y cerrar la ventana de Servicios de Internet Information Server.

22) Para ingresar a LabComp. Dar clic al botón de Inicio de la barra de tareas -> Todos los programas -> Internet Explorer.



Y en la barra de direcciones del explorador teclear la siguiente dirección **http://localhost/labcomp/.** Pulsar el botón Ir. Si logra ver la pantalla de inicio, el sistema esta correctamente instalado y listo para usarse. De lo contrario verificar los puntos anteriores de esta guía.



### 7.4 Descripción del proceso de instalación y su conexión con los capítulos

El proceso de instalación descrito anteriormente en el inciso 7.3, sigue todos los pasos para dejar funcionando LABCOMP en el servidor.

En el desarrollo del sistema se utilizaron los temas tratados en los capítulos de este trabajo, como a continuación se describe:

- Capitulo 1- Plataforma .NET: El .NET Framework versión 2.0 tiene que estar instalado en el servidor para que se pueda ejecutar el sistema. En el inciso 7.3.2 se menciona que entre los prerrequisitos se encuentra el .NET Framework.

  Es posible instalar todo el sistema en el servidor sin que este el .NET Framework. Pero si se intenta accesar al sistema este manda error debido a que el .NET Framework tiene las clases que el sistema requiere para poder ejecutarse.

  El .NET Framework contiene las clases de ASP .NET y ADO .NET.

- Capitulo 2- ASP .NET: ASP .NET se instala al mismo tiempo que el .NET Framework. En el inciso 7.3.2 se hace referencia al Visual Web Developer versión 2005 Express el cual es un entorno de desarrollo para aplicaciones Web ASP .NET. Sin embargo, se puede usar un editor de texto para el desarrollo y edición de las páginas.

  En el inciso 7.3.5 se configura la carpeta LABCOMP la cual tiene las páginas desarrolladas en ASP.NET junto con las imágenes que utilizan. También tiene el código que se asocia con cada página para ejecutar las acciones en el servidor.

- Capitulo 3- Bases de Datos: La base de datos que se configura en el inciso 7.3.3 esta desarrollada en SQL Server Express versión 2005 y utiliza el modelo relacional.

La base cuenta con su diagrama, tablas, vistas y procedimientos almacenados.

Es indispensable que este instalado el programa Microsoft SQL Server Express versión 2005 para poder acceder a la base de datos como se indica en el inciso 7.3.2.

- Capitulo 4- ADO .NET: Al igual que ASP .NET ADO .NET se instala al mismo tiempo que el .NET Framework. Dentro de la carpeta LABCOMP del inciso 7.3.5 se encuentra una carpeta llamada BIN. La carpeta BIN contiene un ensamblado llamado LabConnection la cual usa las clases de ADO .NET para la conexión con la base de datos.

  Para poder hacer la conexión se necesita una cadena de conexión[26] la cual se configuró en el inciso 7.3.4.

  Cuando una página requiere realizar una acción sobre la base de datos. La petición se va al ensamblado LabConnection y para que este funcione necesita ver en donde se encuentra la base de datos, por lo que se dirige al archivo machine.config y lee la cadena para poder ejecutar la acción a la base de datos.

- Capitulo 5- Seguridad en aplicaciones ASP .NET: La seguridad utilizada en el sistema fue mixta, es decir una mezcla entre la seguridad de Windows y la de Formularios. Dentro de la carpeta LABCOMP del inciso 7.3.5 hay un archivo que se llama Web.Config en el cual se indica que tipo de seguridad va a tener la aplicación Web.

  Dentro del archivo Web.Config se escribe el siguiente código para tener la seguridad basada en Formularios:

```
<authentication mode="Forms">
    <forms name="CookieAutoriz" loginUrl="LogIn.aspx">
```

[26] La cadena de conexión contiene la ruta donde se encuentran los archivos de la base de datos.

```
            <credentials passwordFormat="Clear">
               <user name="admin" password="admin"/>
               <user name="usuario1" password="mvz"/>
            </credentials>
         </forms>
      </authentication>
```

En el código se especifica cual es la página a la cual se tiene que dirigir la aplicación en caso de que la persona que esta entrando al sistema no este autorizada para hacerlo. Además se especifican los usuarios que tengan permitido el acceso al sistema[27].

La seguridad de Windows se establece automáticamente cuando el sistema es dado de alta en los Servicios de Información de Internet (Internet Information Server o IIS) como se vio en el inciso 7.3.5 del punto 8 al 16.

- Capitulo 6- Modelo de Aplicación de tres capas: El sistema aplica el modelo de tres capas.

  La capa de presentación se ubica en el inciso 7.3.5 con la carpeta LABCOMP. Dentro de esta carpeta están todas las páginas de presentación con la extensión *.aspx. Estas páginas pueden modificarse sin interferir con el funcionamiento del sistema.

  La capa de negocios se describe también en el inciso 7.3.5. Dentro de la carpeta LABCOMP se encuentra la carpeta BIN. Dentro de la carpeta BIN se encuentran los dos componentes desarrollados para el sistema (LabConnection y LabClass).

  La capa de datos se ubica en el inciso 7.3.3 con los archivos de la base de datos LabComp_Data.mdf y LabComp_Log_ldf.

[27] Debe de existir una línea de código por cada usuario: <user name=”usuario” password=”pass”/>

# CONCLUSIONES

Microsoft se ha caracterizado por comercializar programas siempre con un costo de licencia que por lo general suele ser alto. Pero a partir del año 2000 presento su propuesta llamada Microsoft .NET la cual fue lanzada de una forma totalmente gratuita. En un principio sólo lo fue el .NET Framework y a partir del 2006 el entorno de desarrollo Visual Web Developer versión 2005 Express junto con SQL Server Express versión 2005.

En el desarrollo de LABCOMP pudimos observar que el elemento principal en el que se basa la Plataforma .NET es el .NET Framework. El cual contiene todos los elementos necesarios para desarrollar aplicaciones de Windows, ASP .NET, Web Services, Aplicaciones de consola, Librerías, entre otras. En este caso nos enfocamos a las aplicaciones Web con el uso de ASP.NET.

Las ventajas que observamos al usar la tecnología Microsoft .NET en el sistema LABCOMP fueron las siguientes:

- ASP.NET: Dispone de varios controles, los cuales facilitan el desarrollo de las aplicaciones Web, un ejemplo son los controles de validación que evitan al programador agregar líneas de código para validar los datos de los controles de los formularios.

  Separa el código de la interfaz de usuario facilitando el mantenimiento del sistema.

  Contiene métodos para el manejo de excepciones, lo cual permite identificar los errores que ocurren en la ejecución del código.

  Permite actualizar el sistema copiando el ensamblado modificado sin necesidad de reiniciar el servidor.

  La seguridad mixta basada en Windows y Formularios evita que personas no autorizadas entren al sistema.

- ADO.NET: Dispone de clases que permiten la manipulación de la información sin conexión. Además de que la información se maneja a través de un lenguaje universal como es el XML.

  El uso de objetos sin conexión permite al sistema que la base de datos no se sature con peticiones o conexiones activas.

- .NET Framework: Dispone de clases listas para ser utilizadas o para poder crear más a partir de ellas.

  Dispone de compiladores que nos permitió escoger un lenguaje de programación entre varios que soporta, en este caso Visual Basic .NET.

El titulo de nuestra tesis fue “Desarrollo de un Sistema de Administración para los Laboratorios de Cómputo de la Facultad de Estudios Superiores Cuautitlan con Tecnologías Microsoft .Net” el cual a nuestro punto de vista satisface con los objetivos planteados.

El sistema LABCOMP cubre las necesidades principales de administración de los laboratorios de cómputo como son:

- El llevar el control de las entradas y salidas a los laboratorios por parte de los alumnos y profesores.
- Registrar los recibos por concepto de servicios y llevar un control de ellos mediante reportes.

Nuestra propuesta es que este sistema despierte el interés de seguir desarrollando aplicaciones, ya sea, para los laboratorios de cómputo u otras áreas administrativas. Con esta u otra tecnología (o con la combinación de ambas).

Consideramos que el sistema puede ser modificado, ampliado o mejorado a las necesidades futuras que se vayan presentando en los laboratorios de cómputo. Tomando en cuenta que el código esta disponible para tales efectos.

## Bibliografía


**1.** Sharon Bjeletich, Grez Mabble, et al, Microsoft SQL Server, Madrid 1999, pp. 912.

**2.** Chris Payne, Aprendiendo ASP.NET, Edit. Prentice may, 1a. Edición, México 2002, pp. 1016.

**3.** Rick Dobson**,** Programación de Microsoft SQL Server 2000 con Microsfot Visual Basic .NET, Edit. Mc. Graw Hill, 1a. Edición, España 2002, pp. 643.

**4.** Matthew MacDonald, ASP.NET Manual de Referencia, Edit. Mc. Graw Hill, 1a. Edición, España 2002, pp. 876.

**5.** Francisco Balena, Programación avanzada con Microsoft Visual Basic .NET, Edit. Mc. Graw Hill, 1a. Edición, España 2003, pp. 1238.

**6.** Duncan Mankenzie, Kent Sharkey, Aprendiendo Visual Basic .NET, Edit. Prentice Hall, 1a. Edición, México 2003, pp. 660.

**7.** Michael Amudsen, Paul Litwin, Creación de Sitios Web con ASP.NET, Edit. Prentice Hall 1a. Edición, Madrid 2002, pp. 512.

**8.** Francisco Javier Ceballos, El Lenguaje de Programación Visual Basic .NET, Edit. Alfa Omega Ra-ma, México 2002, pp. 442.

**9.** Erich R. Bühler, Visual Basic .NET Guía de Migración y Actualización, Edit. Mc. Graw Hill, Madrid 2002, pp.949.

**10.** Maria Jesús Ramos, Alicia Ramos y Fernando Montero, Desarrollo de Aplicaciones en entornos de 4ta. Generación y con Herramientas Case, Edit. Mc. Graw Hill, España 2000, pp. 565.

**11.** Jeff Webb, Desarrollo de Aplicaciones Web con Microsoft Visual Basic .NET y Microsoft Visual C# .NET, Publicado por Microsoft, 1a. Edición, USA 2002, pp. 756.

**12.** Chris Payne, Aprendiendo ASP .NET en 21 dias, Edit. Prentice Hall, pp. 1016.

## Referencias de Internet

| | |
|---|---|
| MSDN en español | http://www.microsoft.com/spanish/msdn/spain/default.asp |
| Microsoft México | http://www.microsoft.com |
| ASP.NET | http://www.programacion.com/asp/noticias/2/2 |
| Desarrollador 5 estrellas de Microsoft | http://www.microsoft.com/spanish/msdn/comunidad/dce/1/default.asp#topic1 |
| Información sobre Proyecto Mono | http://es.tldp.org/Presentaciones/200103hispalinux/garcia/html/primera-sec.html |
| Desarrollo Web | http://www.desarrolloweb.com/articulos/1329.php?manual=48 |
| Wikipedia | http://es.wikipedia.org/wiki/DLL_Hell |
| C-SharpCorner | http://www.c-sharpcorner.com/Code/2004/Jan/GloblizationInDotNEt.asp |

## Glosario de Términos

**ASP .NET:** Active Server Pages (Páginas de Servidor Activas o ASP) es una tecnología del lado servidor de Microsoft para páginas web generadas dinámicamente, que ha sido comercializada como un anexo a Internet Information Server (IIS).

**ADO .NET:** ActiveX Data Object (Objetos de datos ActiveX o ADO) es una Tecnología desarrollada por Microsoft para aplicaciones basadas en .NET para el acceso a bases de datos.

**COM:** Component Object Model **(**Modelo de Objetos Componentes) Es un estándar de comunicación entre objetos, la idea principal de Microsoft es tener un estándar en el cual se especifica que los objetos recibirán y enviaran mensajes de una forma definida, esto garantiza que los objetos, no importa en que lenguaje son escritos, se comuniquen entre si.

**COOKIE:** Una cookie (en inglés, galleta) es un fragmento de información que se almacena en el disco duro del visitante de una página web a través de su navegador, a petición del servidor de la página. Esta información puede ser luego recuperada por el servidor en posteriores visitas. Las inventó Lou Montulli, un antiguo empleado de Netscape Communications.

**IIS:** Internet Information Services (Servicios de Información de Internet), es una serie de servicios para los ordenadores que funcionan con Windows. Originalmente era parte del Option Pack para Windows NT. Luego fue integrado en otros sistemas operativos de Microsoft destinados a ofrecer servicios, como Windows 2000 o Windows Server 2003. Windows XP Profesional incluye una versión limitada de IIS.

**INTEROPERAR**: Facultad que tienen ordenadores de distintos fabricantes para cooperar usando un conjunto de protocolos común.

**SCHEMA:** También llamado "Esquema", se trata de un documento de definición estructural al estilo de los DTD, que además cumple con el estándar XML. Los documentos

Schema (usualmente con extensión XSD) se concibieron como un sustituto de los DTD teniendo en cuenta los puntos débiles de estos y la búsqueda de mayores y mejores capacidades a la hora de definir estructuras para los documentos XML, como la declaración de los tipos de datos.

**XLS:** Extensión de los ficheros de creados con Excel.

**XML:** Es el acrónimo del inglés eXtensible Markup Language (lenguaje de marcado ampliable o extensible) desarrollado por el World Wide Web Consortium (W3C).

Lenguaje universal de marcado para documentos estructurados y datos en la web, más amplio y dinámico que HTML. No sólo es un lenguaje de marcado, sino también un metalenguaje que permite describir otros lenguajes de marcado. Permite el uso ilimitado de los tipos de datos que pueden utilizarse en Internet, lo cual resuelve los problemas que surgen entre las instituciones que deben intercambiar datos procedentes de estándares distintos.

**XPATH:** XPath es un lenguaje (basado en XML) que permite seleccionar subconjuntos de un documento XML. La idea es parecida a las expresiones regulares para seleccionar partes de un texto sin atributos (plain text). XPath permite buscar y seleccionar teniendo en cuenta la estructura jerárquica del XML.

**XSL:** XSL está formada por dos lenguajes: XSLT (lenguaje de hojas extensibles de transformación), que permiten generar salidas desde bases de datos XML, incluir etiquetas XHTML y estilos CSS y las XSL-FO, que son las hojas extensibles de formato. En el año 2005 ya son soportadas por algunos navegadores navegador por ejemplo mozilla o Internet Explorer, aunque se pueden usar las CSS que 100% compatibles aunque con una codificación diferente por supuesto en su reemplazo.