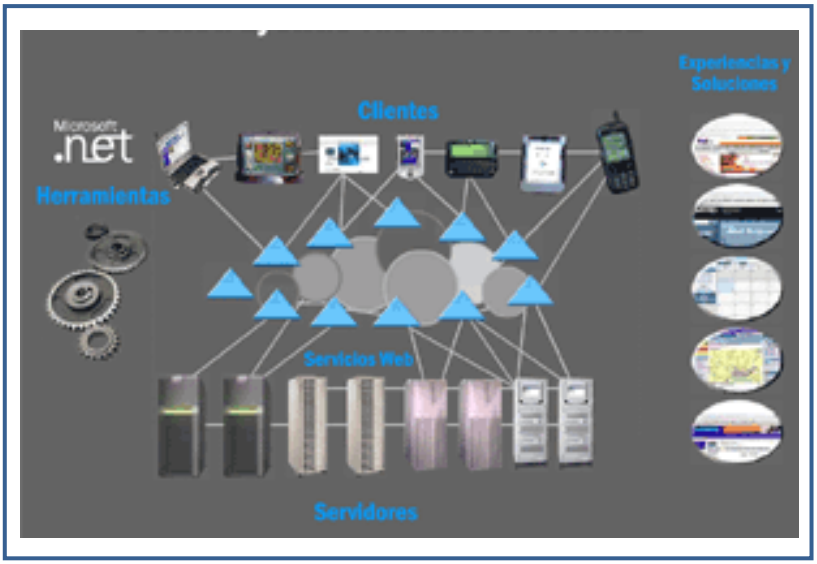
CAPITULO V

TECNOLOGÍA .NET



5.1 Introducción

5.2 Definición de .Net

5.3 Evolución de .Net

5.4 Componentes de la plataforma .Net

5.5 Plataforma de desarrollo .Net Framework

5.6 ASP.Net

5.7 Seguridad en ASP.Net

CAPITULO V

TECNOLOGÍA .NET

5.1 INTRODUCCION

Nos encontramos en un momento especial en la industria de computación, estamos en el inicio de una nueva manera de hacer y de integrar las aplicaciones.
Algunos ***gurús*** de la industria de computación vaticinan que este cambio ser equivalente al que produjo la introducción de la PC, la interfase visual o al surgimiento mismo de Internet. Dispositivos móviles como celulares, ***TabletPC*** hasta televisores u otros dispositivos hogareños estarán conectados entre sí, con servidores y distintas aplicaciones. El elemento integrador será Internet. Estamos ahora en el inicio de la tercera generación de Internet. Con Visual Studio .NET y ASP.NET

Microsoft.NET es el conjunto de nuevas tecnologías en las que Microsoft ha estado trabajando durante los últimos años con el objetivo de obtener una plataforma sencilla y potente para distribuir el software en forma de servicios que puedan ser suministrados remotamente y que puedan comunicarse y combinarse unos con otros de manera totalmente independiente de la plataforma, lenguaje de programación y modelo de componentes con los que hayan sido desarrollados. Ésta es la llamada ***plataforma .NET***, y a los servicios antes comentados se les denomina ***servicios Web***.

Toda la industria de software esta enfocada a usar los servicios web. La plataforma .NET es la infraestructura, más apropiada.
En resumen, con el uso de los servicios Web se integra la información que puede ser accedida desde distintos dispositivos, desde distintas plataformas de hardware o software y que puede estar guardada en distintos formatos. El lenguaje estándar para lograr esta integración es ***XML***.

Para crear aplicaciones para la plataforma .NET, tanto servicios Web como aplicaciones tradicionales (aplicaciones de consola, aplicaciones de ventanas, etc.), Microsoft ha publicado el denominado kit de desarrollo de software conocido como .***NET Framework***, que incluye las herramientas necesarias tanto para su desarrollo como para su distribución y ejecución y Visual Studio.NET, que permite hacer todo la anterior desde una interfaz visual basada en ventanas.

El concepto de Microsoft.NET también incluye al conjunto de nuevas aplicaciones que Microsoft y terceros han (o están) desarrollando para ser utilizadas en la plataforma .NET. Entre ellas podemos destacar aplicaciones desarrolladas por Microsoft tales como Windows.NET, Visual Studio.NET, MSN.NET, Office.NET, y los nuevos servidores para empresas de Microsoft (SQL Server.NET, ***Exchange.NET,*** etc.) [WEB. 5.1]

5.2 DEFINICIÓN DE .NET

.NET es una infraestructura propuesta por Microsoft, para desarrollar aplicaciones Windows y Web. Provee los cimientos para la nueva generación de software. Utiliza los Servicios Web como un medio para poder interoperar a distintas tecnologías. Permite conectar distintos sistemas operativos, dispositivos físicos, información y usuarios. Les da a los desarrolladores las herramientas y tecnologías para hacer rápidamente soluciones de negocios que involucran distintas aplicaciones, dispositivos físicos y organizaciones.

"Microsoft .NET es un conjunto de tecnologías de software de Microsoft para conectar su mundo de información, gente, sistemas y dispositivos".

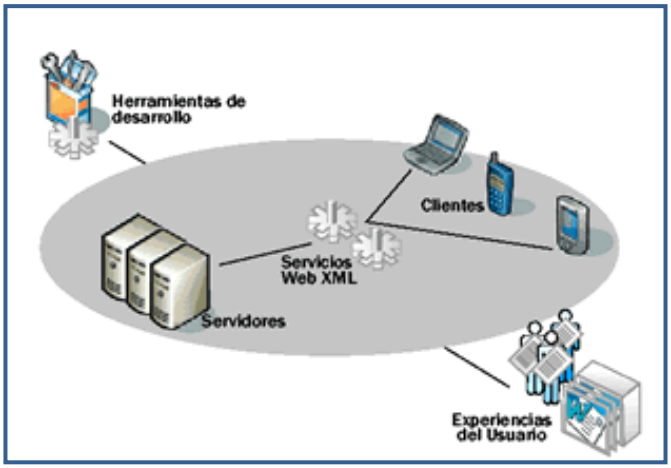


Fig.5.1 Plataforma.Net

Fuente: http://www.desarrolloweb.com/articulos/1681.php

La idea central detrás de la plataforma .NET es la de servicio. 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 los bloques de construcción o componentes sobre los cuales se basa el modelo de computación distribuida en Internet. 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 proveen.

5.3 EVOLUCIÓN DE .NET

La informática se inició con programas monousuarios implantados en grandes ordenadores. Posteriormente estas primeras aplicaciones alcanzaron la capacidad de atender a diferentes usuarios. Pasaron los años y llegó la arquitectura cliente – servidor, que gracias a este modelo de desarrollo, la aplicación se dividía en una parte que interaccionaba con el usuario y otra parte destinada al procesamiento de información. Esto consiguió que cada una de las partes que constituían la aplicación pudiera residir en computadoras distintas.

A mediados de la década de los 90 y con la aparición del Internet y su posterior masificación a niveles jamás pensados, ha existido siempre la necesidad e inquietud por parte de las empresas desarrolladoras de software de buscar o contar con la manera de lograr la integración entre ***sistemas heterogéneos***, tanto de software como de hardware. Muchas empresas comenzaron una loza carrera para generar la mejor tecnología integradora de sistemas, pero a medida que la competencia se hacia más fuerte, la integración se hacia cada vez más difícil.

Conociendo de todo este hecho, Microsoft quiso aprovechar la oportunidad para desarrollar una tecnología llamada Microsoft .NET para generar un marco de trabajo en el que está inundado por la palabra "Servicios" y .NET es ofrecer servicios informáticos a través de redes TCP/IP y web, pero que fuera aprovechado por cualquier lenguaje de programación que se ciñera a sus estándares.

Microsoft entonces, diseño un FRAMEWORK, que es el corazón de .NET y es el resultado de la unión de dos proyectos uno relacionado con el desarrollo de aplicaciones web y de aplicaciones distribuidas, mientras que el segundo proyecto conocido como ***NGWS*** (Next Generation Windows Services – Siguiente Generación de Servicios Windows), es la creación de una plataforma para el desarrollo de software como servicio.

5.3.1 Finalidades De .Net

.NET representa la visión de Microsoft, del software como un servicio, habiendo sido diseñada con Internet en mente, cubre todas las capas del desarrollo de software, existiendo una lata integración entre las tecnologías de presentación, de componentes y de acceso a datos, posee varias finalidades y estas son:

- Mantener estándares abiertos de Internet con XML(transformar), ***HTTP***(conectar), ***SOAP***(solicitar), ***UDDI***(encontrar), ***WSDL***(describir). El reto de Microsoft es proporcionar la mejor implementación en el mercado para estos estándares con sus productos y herramientas.
- Servicios web XML mediante componentes de software, que puedan accederse de manera programática a través del web, logrando potencializar las aplicaciones.
- Proporcionar mecanismos de integración para que una empresa pueda ofrecer servicios a otras empresas o clientes de una forma sencilla y rápida ya sea de manera interna o expuesta a través de Internet.
- Modelo de programación simple y consistente permitiendo a desarrolladores centrarse en la lógica de la aplicación, ofreciendo herramientas y tecnologías mediante el soporte de estándares sobre los cuales se basan los servicios web.
- Liberar al programador de las cuestiones de infraestructura, es decir de los aspectos funcionales.
- Proporcionar soporte para arquitecturas fuertemente acopladas y débilmente acopladas, para conseguir un buen rendimiento, escalabilidad y confiabilidad con grandes sistemas distribuidos. [LIB 5.1]

5.3.2 Características de .Net

.Net es una plataforma que esta evolucionando y se caracteriza por:

- .NET es una nueva plataforma para el desarrollo, programación y explotación de aplicaciones “gestionadas” modernas y orientadas a objetos, en todas sus herramientas de Visual Studio.Net.
- Posee una plataforma de desarrollo llamada Framework.

- Las aplicaciones .NET se pueden desarrollar en cualquier lenguaje de programación que se ajusta a .NET
- .NET ofrece integración multi-lenguaje, reutilización de componentes, y herencia entre componentes desarrollados en diferentes lenguajes
- .NET ofrece una nueva manera de desarrollar aplicaciones basadas en navegador Web a través de ASP.NET
- Las clases ***ADO.NET*** proveen una arquitectura desconectada para acceso a datos a través de Internet
- .NET soporta la creación de Servicios Web XML independientes de la plataforma, a través de SOAP y WSDL.

5.3.3 Ventajas De .Net

Las ventajas más importantes que proporciona .Net son:

- Código administrado: El ***CLR*** realiza un control automático del código para que este sea seguro, es decir, controla los recursos del sistema para que la aplicación se ejecute correctamente.
- Interoperabilidad multi-lenguaje: El código puede ser escrito en cualquier lenguaje compatible con .Net ya que siempre se compila en código intermedio (***MSIL***).
- Compilación ***just-in-time***: El compilador JIT incluido en el Framework compila el código intermedio (MSIL) generando el código máquina propio de la plataforma. Se aumenta así el rendimiento de la aplicación al ser específico para cada plataforma.
- ***Garbage collector***: El CLR proporciona un sistema automático de administración de memoria denominado recolector de basura (garbage collector). El CLR 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 (mediante el método disponse() liberamos el objeto para que el recolector de basura lo elimine de memoria).

- Seguridad de acceso al código: Se puede especificar que una pieza de código tenga permisos de lectura de archivos pero no de escritura. Es posible aplicar distintos niveles de seguridad al código, de forma que se puede ejecutar código procedente del Web sin tener que preocuparse si esto va a estropear el sistema.
- Despliegue: Por medio de los ensamblados resulta mucho más fácil el desarrollo de aplicaciones distribuidas y el mantenimiento de las mismas. El Framework realiza esta tarea de forma automática mejorando el rendimiento y asegurando el funcionamiento correcto de todas las aplicaciones. [WEB 5.2]

5.4 COMPONENTES DE LA PLATAFORMA .NET

La plataforma .NET no es un solo 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.

Tanto la invocación de los servicios como su ejecución pueden ser hechas en cualquier dispositivo y sistema operativo, y accedido desde Internet. Los sitios se comunican entre sí y acceden a servicios y contenidos sin la intervención humana. Por eso se llama a la nueva generación de Internet "Internet inteligente".

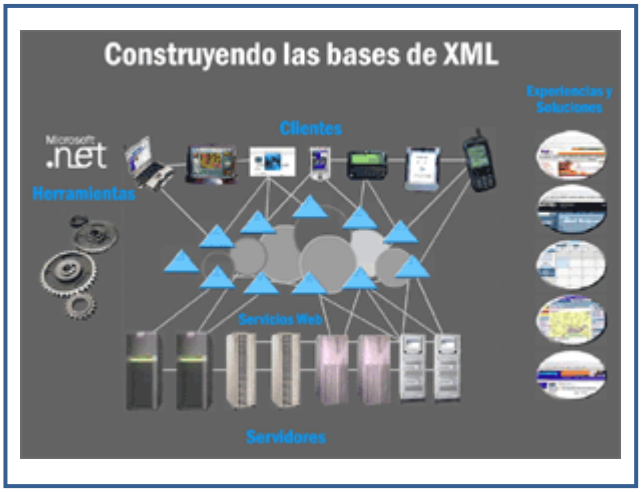


Fig. 5.2 Componentes de .NET

Fuente: http://www.desarrolloweb.com/articulos/1681.php

Los componentes de la plataforma .NET son:

- Clientes Inteligentes o Smart Clients
- Servidores
- Servicios Web basados en XML
- Herramientas de desarrollo

5.4.1 Clientes Inteligentes (SMART CLIENTS)

Son dispositivos muy variados. Lo que los hace 'Smart' o inteligentes es su capacidad para hacer uso de servicios Web.

Fig. 5.3 Clientes Inteligentes

Fuente: http://www.desarrolloweb.com/articulos/1681.php

Sus características son:

- Permiten acceder a la información en el formato apropiado, en cualquier momento y lugar.
- Hacen uso de 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 están. Por ejemplo la velocidad de transmisión.
- Tienen capacidad de procesamiento propio, y distribuyen el procesamiento en la red haciendo uso de los servicios Web.

Algunos ejemplos de Clientes Inteligentes son**:**

- ⇨ **SmartPhone** (Teléfono Inteligente)
- ⇨ **PocketPC** (PC de bolsillo)
- ⇨ **TabletPC**
- ⇨ **PCs:** Las computadoras personales.
- ⇨ **NoteBooks:** Las computadoras portátiles, y muchos otros dispositivos en desarrollo.

5.4.2 Servidores

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

Fig 5.4 Servidores

Fuente: Propia

Sistemas Operativos: Windows 2000: Server, Advance Server y Datacenter, Windows Server 2003: Standard, Enterprise, Datacenter y Web Server.

Algunos ejemplos de Servidores .NET son:

- **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 Commerce Server 2000:** Para construir rápidamente soluciones de e-commerce escalables.
- **Microsoft Content Management Server 2001:** Para administrar contenido para sitios Web de e-bussines 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 SharePoint 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.

5.4.3 Servicios Web Basados En Xml

Son cualquier componente o bloque programable de una aplicación, que es accesible a través de protocolos estándar 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.

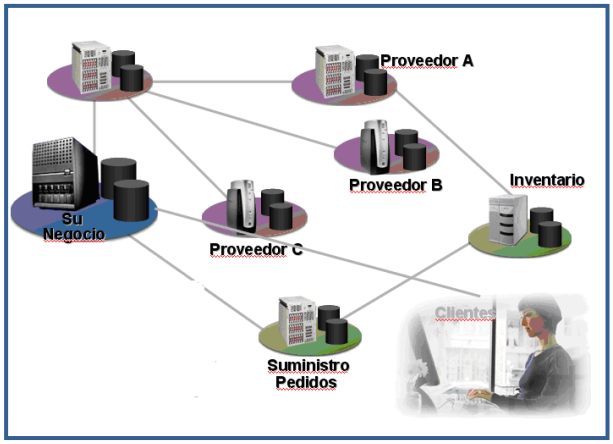


Fig. 5.5 Servicios Web basados en XML

Fuente: http://www.desarrolloweb.com/articulos/1681.php

Estos son algunos de los protocolos estándares que permiten hacer uso de los Servicios Web basados en XML:

- ***XML:*** (Lenguaje de Marcado eXtensible) Es un formato universal para representar los datos.
- ***SOAP:*** (Protocolo Simple de Acceso a Objetos) Es un protocolo que permite mover los datos entre aplicaciones y sistemas. Es el mecanismo por medio del cual los servicios Web son invocados e interactúan.
- ***UDDI:*** (Descubrimiento, Descripción e Integración Universal) Lenguaje que permite publicar, encontrar y usar los Servicios Web basados en XML. Es la 'Página Amarilla' de los servicios Web es decir un directorio para poder encontrarlos.
- ***WSDL:*** (Lenguaje de Descripción de Servicios Web) Lenguaje por medio del cual un servicio Web describe entre otras cosas qué hace o qué funcionalidad implementa.

Se busca crear una plataforma de servicios Web para conseguir aplicaciones centradas en el usuario. La idea es que el usuario 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, como por ejemplo la PC en el trabajo o una PocketPC. La información es la información del usuario y el software debe llevarla hacia él sin importar en qué dispositivo se encuentre trabajando.

5.4.4 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.

Fig. 5.6 Herramientas de desarrollo
Fuente Propia

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.

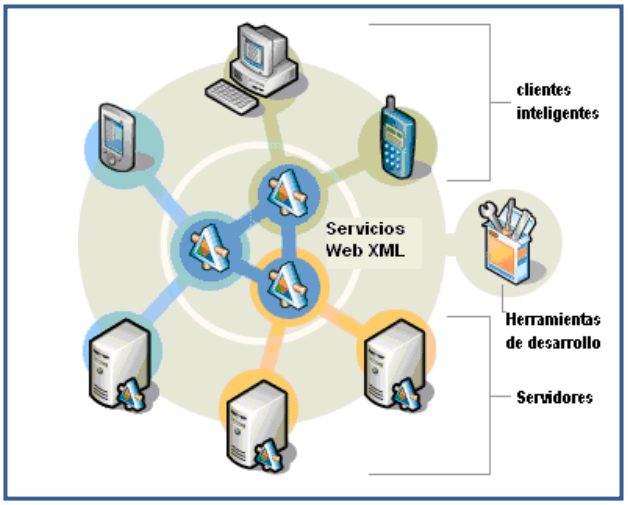


Fig. 5.7 Tipos de interacciones de los componentes de .NET
Fuente: "http://elementostecnologianet/montejava/docs/images.htm"

Tipos de interacciones posibles

Cliente con Cliente: 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 PC o un dispositivo móvil vía Internet.

Servidor con Servidor: Una aplicación en un servidor puede programáticamente 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.

Existen varios ejemplos de la familia .NET y sus componentes, que a continuación se presentan en la siguiente tabla. [WEB 5.3]

Componentes	**Definición**	**Ejemplo**
Herramientas para los desarrolladores	Interfases de programación y herramientas para diseñar, crear, ejecutar e instalar soluciones basadas en .NET.	.NET Framework. Visual Studio .NET
Servidores	Infraestructura para construir, instalar y operar la plataforma .NET	Windows 2000 Server Servidores Corporativos .NET
Servicios Web XML	Servicios existentes que implementan tareas predefinidas. Servicios hechos por el desarrollador.	.NET Passport MapPoint .NET
Clientes	Dispositivos físicos que corren en sistemas operativos que integran e interactúan con otros elementos .NET.	Windows CE para dispositivos portátiles Windows XP para PCs

Tabla 5.1 Ejemplos de .NET

Fuente: http://www.desarrolloweb.com/articulos/1681.php

5.5 PLATAFORMA DE DESARROLLO .NETFRAMEWORK

Es claro entonces que el objetivo de la plataforma .NET es simplificar el desarrollo de aplicaciones Web. Provee las herramientas y tecnologías para transformar a Internet en una plataforma de computación distribuida en gran escala. Esta plataforma además soporta los estándares sobre los cuales se basan los servicios Web.

El **.NET Framework** se instala como un componente aparte en Windows 2000, mientras que Windows XP y las futuras versiones de Windows lo incorporan directamente al sistema operativo. Como por ejemplo Windows Server 2003

El **.NET Compact Framework** permite hacer uso de los servicios Web en dispositivos móviles. Debido a que es un subconjunto del .NET Framework comparte el mismo modelo de programación y herramientas de desarrollo de aplicaciones haciendo posible que los desarrolladores transfieran sus conocimientos existentes al desarrollo de aplicaciones móviles.

5.5.1 Definición

El Microsoft .NET Framework, es un componente de software que puede ser o es incluido en los sistemas operativos Microsoft Windows, es decir es un conjunto de servicios de programación diseñados para simplificar el desarrollo de aplicaciones en el entorno altamente distribuido de Internet. Provee soluciones pre-codificadas para requerimientos comunes de los programas y gestiona la ejecución de programas escritos específicamente para este framework.

Microsoft desea que todas las aplicaciones creadas para la plataforma Windows, sean basadas en el .NET Framework. Su objetivo es crear un marco de desarrollo de software sencillo, reduciendo las vulnerabilidades y aumentando la seguridad de los programas desarrollados.

5.5.2 Objetivos De .Net Framework

.Net Framework Es un componente integral de Windows que admite la creación y la ejecución de la siguiente generación de aplicaciones y servicios Web XML. El diseño de .Net Framework está enfocado a cumplir los siguientes objetivos:

- 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, pero distribuida en Internet o 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 fomente 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.
- Basar toda la comunicación en estándares del sector para asegurar que el código de .Net Framework se puede integrar con otros tipos de código. [WEB 5.4]

5.5.3 Componentes De .Net Framework

El Framework de .Net es una infraestructura sobre la que se reúne todo un conjunto de lenguajes y servicios que simplifican enormemente el desarrollo de aplicaciones. Mediante esta herramienta se ofrece un entorno de ejecución altamente distribuido, que permite crear aplicaciones robustas y escalables. Los principales componentes de este entorno son:

- Lenguajes de compilación
- Biblioteca de clases de .Net
- CLR (Common Language Runtime)

.Net Framework soporta múltiples lenguajes de programación y aunque cada lenguaje tiene sus características propias, es posible desarrollar cualquier tipo de aplicación con cualquiera de estos lenguajes. Existen más de 30 lenguajes adaptados a .Net, desde los más conocidos como C# (C Sharp), Visual Basic o C++ hasta otros lenguajes menos conocidos como ***Perl*** o Cobol. [WEB 5.2]

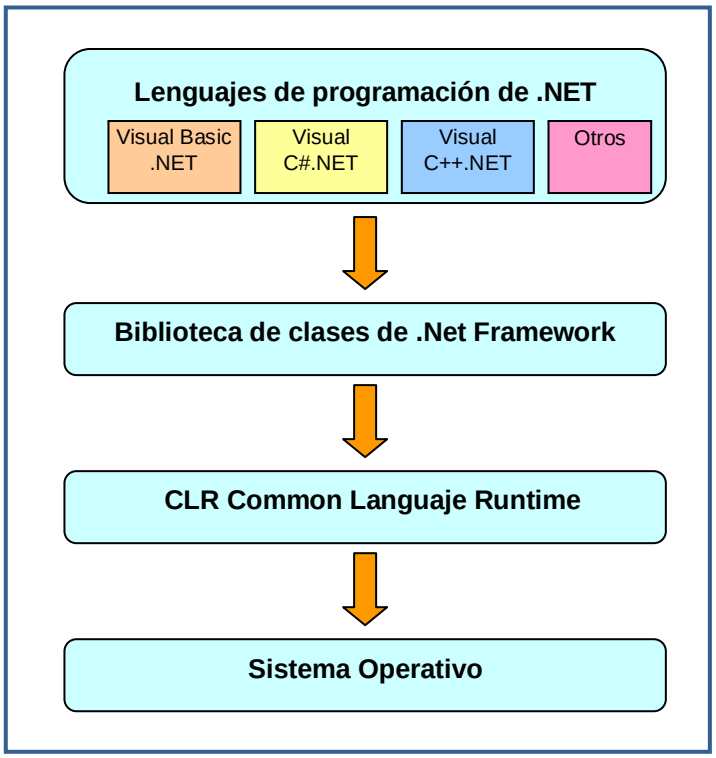


Fig. 5.8 Componentes de .NETFRAMEWORK

Fuente: http://www.desarrolloweb.com

5.5.3.1 CLR (COMMON LANGUAJE RUNTIME)

El CLR es el verdadero núcleo del Framework de .Net, ya que es el entorno de ejecución en el que se cargan las aplicaciones desarrolladas en los distintos lenguajes.

La herramienta de desarrollo compila el código fuente de cualquiera de los lenguajes soportados por .Net en un mismo código, denominado código intermedio (MSIL, Microsoft Intermediate Lenguaje). Para generar dicho código el compilador se basa en el Common Language Specification (***CLS***) que determina las reglas necesarias para crear código MSIL compatible con el CLR. De esta forma, indistintamente de la herramienta de desarrollo utilizada y del lenguaje elegido, el código generado es siempre el mismo, ya que el MSIL es el único lenguaje que entiende directamente el CLR. Este código es transparente al desarrollo de la aplicación ya que lo genera automáticamente el compilador. Sin embargo, el código generado en MSIL no es código máquina y por tanto no

puede ejecutarse directamente. Se necesita un segundo paso en el que una herramienta denominada compilador JIT (Just-In-Time) genera el código máquina real que se ejecuta en la plataforma que tenga la computadora.
De esta forma se consigue con .Net cierta independencia de la plataforma, ya que cada plataforma puede tener su compilador JIT y crear su propio código máquina a partir del código MSIL.
La compilación JIT la realiza el CLR a medida que se invocan los métodos en el programa y, el código ejecutable obtenido, se almacena en la memoria caché de la computadora, siendo recompilado sólo cuando se produce algún cambio en el código fuente.

El Common Language Runtime (CLR) es el núcleo de la plataforma .NET. Es el motor encargado de gestionar la ejecución de las aplicaciones para ella desarrolladas y a las que ofrece numerosos servicios que simplifican su desarrollo y favorecen su fiabilidad y seguridad. Las principales características y servicios que ofrece el CLR son:

Modelo de programación consistente: A todos los servicios y facilidades ofrecidos por el CLR se accede de la misma forma: a través de un modelo de programación orientado a objetos. Esto es una diferencia importante respecto al modo de acceso a los servicios ofrecidos por los algunos sistemas operativos actuales (por ejemplo, los de la familia Windows), en los que a algunos servicios se les accede a través de llamadas a funciones globales definidas en ***DLL's*** y a otros a través de objetos (***objetos COM*** en el caso de la familia Windows)

Modelo de programación sencillo: Con el CLR desaparecen muchos elementos complejos incluidos en los sistemas operativos actuales (registro de Windows, ***GUIDs***, etc.) El CLR no es que abstraiga al programador de estos conceptos, sino que son conceptos que no existen en la plataforma .NET

Eliminación del "infierno de las DLLs": En la plataforma .NET desaparece el problema conocido como "infierno de las DLLs" que se da en los sistemas

operativos actuales de la familia Windows, problema que consiste en que al sustituirse versiones viejas de DLLs compartidas por versiones nuevas puede que aplicaciones que fueron diseñadas para ser ejecutadas usando las viejas dejen de funcionar si las nuevas no son 100% compatibles con las anteriores. En la plataforma .NET las versiones nuevas de las DLLs pueden coexistir con las viejas, de modo que las aplicaciones diseñadas para ejecutarse usando las viejas podrán seguir usándolas tras instalación de las nuevas. Esto, obviamente, simplifica mucho la instalación y desinstalación de software.

Ejecución multiplataforma: El CLR actúa como una máquina virtual, encargándose de ejecutar las aplicaciones diseñadas para la plataforma .NET. Es decir, cualquier plataforma para la que exista una versión del CLR podrá ejecutar cualquier aplicación .NET.

Integración de lenguajes: Desde cualquier lenguaje para el que exista un compilador que genere código para la plataforma .NET es posible utilizar código generado para la misma usando cualquier otro lenguaje tal y como si de código escrito usando el primero se tratase. La integración de lenguajes esta que es posible escribir una clase en C# que herede de otra escrita en Visual Basic.NET que, a su vez, herede de otra escrita en C++ con extensiones gestionadas.

Gestión de memoria: El CLR incluye un recolector de basura que evita que el programador tenga que tener en cuenta cuándo ha de destruir los objetos que dejen de serle útiles. Este recolector es una aplicación que se activa cuando se quiere crear algún objeto nuevo y se detecta que no queda memoria libre para hacerlo, caso en que el recolector recorre la memoria dinámica asociada a la aplicación, detecta qué objetos hay en ella que no puedan ser accedidos por el código de la aplicación, y los elimina para limpiar la memoria de "objetos basura" y permitir la creación de otros nuevos. Gracias a este recolector se evitan errores de programación muy comunes como intentos de borrado de objetos ya borrados, agotamiento de memoria por olvido de eliminación de objetos inútiles o solicitud de acceso a miembros de objetos ya destruidos.

Seguridad de tipos: El CLR facilita la detección de errores de programación difíciles de localizar comprobando que toda conversión de tipos que se realice durante la ejecución de una aplicación .NET se haga de modo que los tipos origen y destino sean compatibles.

Aislamiento de procesos: El CLR asegura que desde código perteneciente a un determinado proceso no se pueda acceder a código o datos pertenecientes a otro, lo que evita errores de programación muy frecuentes e impide que unos procesos puedan atacar a otros.

Distribución transparente: El CLR ofrece la infraestructura necesaria para crear objetos remotos y acceder a ellos de manera completamente transparente a su localización real, tal y como si se encontrasen en la máquina que los utiliza.

Seguridad avanzada: El CLR proporciona mecanismos para restringir la ejecución de ciertos códigos o los permisos asignados a los mismos según su procedencia o el usuario que los ejecute. Es decir, puede no darse el mismo nivel de confianza a código procedente de Internet que a código instalado localmente o procedente de una red local; puede no darse los mismos permisos a código procedente de un determinado fabricante que a código de otro; y puede no darse los mismos permisos a un mismo códigos según el usuario que lo esté ejecutando o según el rol que éste desempeñe. Esto permite asegurar al administrador de un sistema que el código que se esté ejecutando no pueda poner en peligro la integridad de sus archivos, la del registro de Windows, etc.

Interoperabilidad con código antiguo: El CLR incorpora los mecanismos necesarios para poder acceder desde código escrito para la plataforma .NET a código escrito previamente a la aparición de la misma y, por tanto, no preparado para ser ejecutando dentro de ella. Estos mecanismos permiten tanto el acceso a objetos COM como el acceso a funciones sueltas de DLLs preexistentes (como la API Win32) [WEB 5.5]

5.5.3.2 Biblioteca De Clases

La Librería de Clase Base (***BCL***) es una librería incluida en el .NET Framework formada por cientos de tipos de datos que permiten acceder a los servicios ofrecidos por el CLR y a las funcionalidades más frecuentemente usadas a la hora de escribir programas. Además, a partir de estas clases prefabricadas el programador puede crear nuevas clases que mediante herencia extiendan su funcionalidad y se integren a la perfección con el resto de clases de la BCL.

Esta librería está escrita en MSIL, por lo que puede usarse desde cualquier lenguaje cuyo compilador genere MSIL. A través de las clases suministradas en ella es posible desarrollar cualquier tipo de aplicación, desde las tradicionales aplicaciones de ventanas, consola o servicio de Windows NT hasta los novedosos servicios Web y páginas ASP.NET. Es tal la riqueza de servicios que ofrece que puede crearse lenguajes que carezcan de librería de clases propia y sólo usen la BCL como C#.

La biblioteca de clases de .Net Framework incluye, entre otros, tres componentes clave:

- ASP.NET para construir aplicaciones y servicios Web.
- ***Windows Forms*** para desarrollar interfaces de usuario.
- ADO.NET para conectar las aplicaciones a bases de datos.

La forma de organizar la biblioteca de clases de .Net dentro del código es a través de los espacios de nombres (***namespaces***), donde cada clase está organizada en espacios de nombres según su funcionalidad.
La principal ventaja de los espacios de nombres de .Net es que de esta forma se tiene toda la biblioteca de clases de .Net centralizada bajo el mismo espacio de nombres (System). Además, desde cualquier lenguaje se usa la misma sintaxis de invocación, ya que a todos los lenguajes se aplica la misma biblioteca de clases.
Por ejemplo, los espacios de nombres más usados son: [WEB 5.5]

Espacio de nombres	Utilidad de los tipos de datos que contiene
System	Tipos muy frecuentemente usados, como los tipos básicos, tablas, excepciones, fechas, números aleatorios, recolector de basura, entrada/salida en consola, etc.
System.Collections	Colecciones de datos de uso común como pilas, colas, listas, diccionarios, etc.
System.Data	Manipulación de bases de datos. Forman la denominada arquitectura **ADO.NET**.
System.IO	Manipulación de ficheros y otros flujos de datos.
System.Net	Realización de comunicaciones en red.
System.Reflection	Acceso a los ***metadatos*** que acompañan a los módulos de código.
System.Runtime.Remoting	Acceso a objetos remotos.
System.Security	Acceso a la política de seguridad en que se basa el CLR.
System.Threading	Manipulación de hilos.
System.Web.UI.WebControls	Creación de interfaces de usuario basadas en ventanas para aplicaciones Web.
System.Winforms	Creación de interfaces de usuario basadas en ventanas para aplicaciones estándar.
System.XML	Acceso a datos en formato XML.

Tabla 5.2 Ejemplos de espacios de nombres

Fuente: Visual Studio 2005

5.6 ASP.NET

ASP.NET es un conjunto de tecnologías de desarrollo de aplicaciones web comercializado por Microsoft. Es usado por programadores para construir sitios web domésticos, portales, aplicaciones web y servicio XML.

Una aplicación ASP.NET se define como todos los archivos, páginas, controladores, módulos y código ejecutable a los que se puede llamar desde un directorio virtual y sus subdirectorios en un único servidor de aplicaciones web.

5.6.1 Antecedentes de ASP.NET

ASP.NET es más que una nueva versión de paginas Active Server (ASP) proporciona un modelo de desarrollo web unificado que incluye los servicios

necesarios para que los programadores creen aplicaciones web. ASP.NET es mucho mejor que el ASP tradicional, con ello trae diversas mejoras entre las cuales se destacan:

Rendimiento: La aplicación se compila en una sola vez al lenguaje nativo, y luego, en cada petición tienen una compilación JIT, es decir se compila desde el código nativo, lo que permite mucho mejor rendimiento.

Rapidez en programación: Mediante diversos controles, se puede con unas pocas líneas y en pocos minutos mostrar toda una base de datos y hacer rutinas complejas.

Servicios Web: Trae herramientas para compartir datos e información entre distintos sitios.

Seguridad: Tiene diversas herramientas que garantizan la seguridad de nuestras aplicaciones.

ASP.NET es un entorno compilado basado en .NET, puede crear aplicaciones en cualquier lenguaje compatible con .NET, como visual Basic.NET, C#, etc. Además .NET Framework está disponible en su totalidad para cualquier aplicación ASP.NET. Se pueden aprovechar fácilmente las ventajas de estas tecnologías, que incluyen el entorno CLR administrando seguridad de tipos, herencia, etc. [LIB. 5.2]

5.6.2 Características de ASP.NET

ASP.NET se ha construido bajo los siguientes principios:

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

Facilidad de desarrollo.- ASP.NET introduce un nuevo concepto, los "***server controls***", que permiten a modo de etiquetas HTML tener controles manejados por

el servidor que identifican el navegador. Tareas tediosas como la validación de datos se convierten en fáciles y sencillas.
Posibilidad de elección del lenguaje de programación, puedes elegir el lenguaje de programación que más te guste, por defecto lleva integrado C#, VB.NET y J#, pero podrías usar otro lenguaje.
Independencia de la herramienta de desarrollo. Puedes utilizar desde el ***Notepad***, hasta la sofistica y potente Visual Studio .NET, pasando por la gratuita Web Matriz. Y lo mejor de todo es la rica biblioteca de clases que lleva incorporada, ya no necesitarás obtener componentes de otras empresas para por ejemplo enviar un email, hacer "upload" de un fichero o generar gráficos en tiempo de ejecución.

Alto rendimiento y escalabilidad.- El código es compilado para ser ejecutado en el CLR. Puedes optar por tenerlo en el servidor precompilado o dejar que el servidor lo compile la primera vez que lo ejecute. El resultado es de 3 a 5 veces superior en velocidad que las antiguas páginas ASP.
Rico sistema de ***caché***. El uso adecuado del potente caché incorporado aumenta considerablemente el rendimiento y la escalabilidad de la aplicación. La caché te permitirá cachear desde páginas completas a partes completas, pasando por conjuntos de datos extraídos de la base de datos.
ASP.NET está preparado para poder tener granjas de servidores web para sitios con alto volumen de tráfico y repartir la carga entre distintos servidores.

Mejora de la fiabilidad.- ASP.NET es capaz de detectar perdidas de memoria, problemas con bloqueos y protección ante caídas. Entre otras cosas, es capaz de detectar aplicaciones web que pierden memoria, arrancando otro proceso limpio con una nueva instancia de la aplicación para cerrar la que pierde memoria liberando así la memoria perdida.

Fácil distribución e instalación.- Una aplicación ASP.NET se instala tan fácilmente como copiando los ficheros que la componen. No es necesario registrar ningún componente, tan solo copiar los ficheros al web. Puedes recompilar la

aplicación o enviar nuevos ficheros sin necesidad de reiniciar la aplicación ni el servidor web. [LIB. 5.3]

5.6.3 Funcionalidad de ASP.NET

ASP.NET introduce el concepto del ***code-behind***, por el que una misma página se compone de dos ficheros: el de la interfaz de usuario y el de código. Con ello se facilita la programación de aplicaciones en múltiples capas, lo que en definitiva se traduce en la total separación entre lo que el usuario ve y lo que la base de datos tiene almacenado.
El funcionamiento de su proceso comienza con el ***browser*** el cual ejecuta una petición http al servidor, este la recibe, ejecuta los ***scripts*** correspondientes a la página y solicitud adecuados en el servidor. Lo cual genera una página html la misma que es reenviada al cliente que realizo la petición.

Desarrollo de aplicaciones cliente.- Las aplicaciones cliente constituyen lo más parecido a una aplicación de estilo tradicional en la programación basada en Windows. En este tipo de aplicaciones se muestran ventanas o formularios en el escritorio, lo que permite al usuario realizar una tarea. Entre las aplicaciones cliente se incluyen los procesadores de texto y las hojas de cálculo, además de aplicaciones empresariales, como herramientas de entrada de datos, de informes, etc. En las aplicaciones cliente se suelen emplear ventanas, menús, botones y otros elementos de la interfaz gráfica de usuario, y suelen tener acceso a recursos locales como el sistema de archivos y a dispositivos periféricos como las impresoras.
Otro tipo de aplicación cliente es el tradicional ***control ActiveX*** (reemplazado ahora por el control de ***Windows Forms***) implementado en Internet como una página web. Esta aplicación es muy parecida a otras aplicaciones cliente: se ejecuta de forma nativa, tiene acceso a los recursos locales e incluye elementos gráficos.

Desarrollo de aplicaciones de servidor.- Las aplicaciones de servidor en entornos administrados se implementan mediante ***hosts*** de motor de tiempo de ejecución. Las aplicaciones no administradas alojan CLR, que permite al código administrado personalizado controlar el comportamiento del servidor. Este modelo proporciona todas las características de CLR y la biblioteca de clases, además de obtener el rendimiento y la escalabilidad del servidor host.

5.6.4 Ventajas de ASP.NET frente a ASP

- Para que todo ocurra en una página web, es habitual escribir una gran cantidad de código para resolver necesidades sencillas. ASP.NET incorpora un modelo declarativo a la programación web: los controles de servidor funcionan en una página Web simplemente declarándolos. Cuando se carga la página ASP.NET, se ***instancian*** los controles listados en la página ASP y es responsabilidad del control emitir código HTML que el navegador pueda entender.

- ASP clásico es un tanto desorganizado. En una página ASP podemos incluir casi todo: HTML plano, código script, objetos COM y texto. No hay una distinción formal entre el contenido de una página y su comportamiento: simplemente, insertamos código en la página, y a ver qué pasa. ASP.NET impone un cierto orden sobre el modelo de programación estándar ASP. En cierto modo, esta "desorganización" puede evitarse fácilmente usando el sentido común y algunas de las nuevas tecnologías.

- La tercera limitación en el desarrollo con ASP es que con el tradicional utilizamos lenguajes de ***scripting*** no tipados como VBScript o JScript. Podemos instalar otros motores de scripting que impongan verificación de tipos; sin embargo, no son universalmente conocidos o utilizados como los anteriores. ASP.NET claramente separa la porción basada en script de una página web de su contenido.

- ASP.Net, puede decirse que en nuevo nivel de abstracción en la construcción de sitios web, por que se pueden crear rápidamente aplicaciones web, basándose en los controles incluidos en el frameWork o muchos gratuitos que hay en la red, ocultando el código de mucho. Excelente para desarrollo de aplicaciones multicapas.

- ASP solo utiliza lenguajes Script (VBscript y Jscript) los cuales son interpretados y no son fuertemente ***tipados***. Mientras que ASP.NET soporta múltiples lenguajes compilados y fuertemente tipados incluyendo (Scripts) que ya son compilados antes de su ejecución, lo que permite una ganancia en su rendimiento. [WEB 5.6]

5.7 SEGURIDAD EN ASP.NET

La seguridad de los sitios Web es una cuestión de importancia fundamental, además de compleja, para los desarrolladores de sitios Web. La protección de un sitio requiere la elaboración cuidadosa de un plan; por consiguiente, los programadores y administradores de sitios Web deben comprender perfectamente las opciones para proteger los sitios.
ASP.NET funciona junto con Microsoft .NET Framework y Servicios de Microsoft Internet Information Server (***IIS***) para ayudar a proporcionar aplicaciones Web seguras. Para ayudar a proteger la seguridad de una aplicación ASP.NET, se deben llevar a cabo las dos funciones principales que son:

- autorización
- autenticación

5.7.1 Arquitectura de Seguridad de ASP.NET

Una importante de muchas aplicaciones web radica en la capacidad de identificar usuarios y controlar el acceso a los recursos. Cuando se esta ejecutando una aplicación ASP.NET, puede utilizar las características de seguridad de ASP.NET

integradas. Además puede utilizar las características de seguridad de .NET Framework.

Como se muestra en la siguiente figura, todos los clientes web se comunican con las aplicaciones ASP.NET a través de Servicios de Microsoft Internet Information Server (IIS). IIS autentica la solicitud si fuera necesario y a continuación busca el recurso solicitado. Si el cliente esta autorizado, el recurso estará disponible. [WEB 5.8].

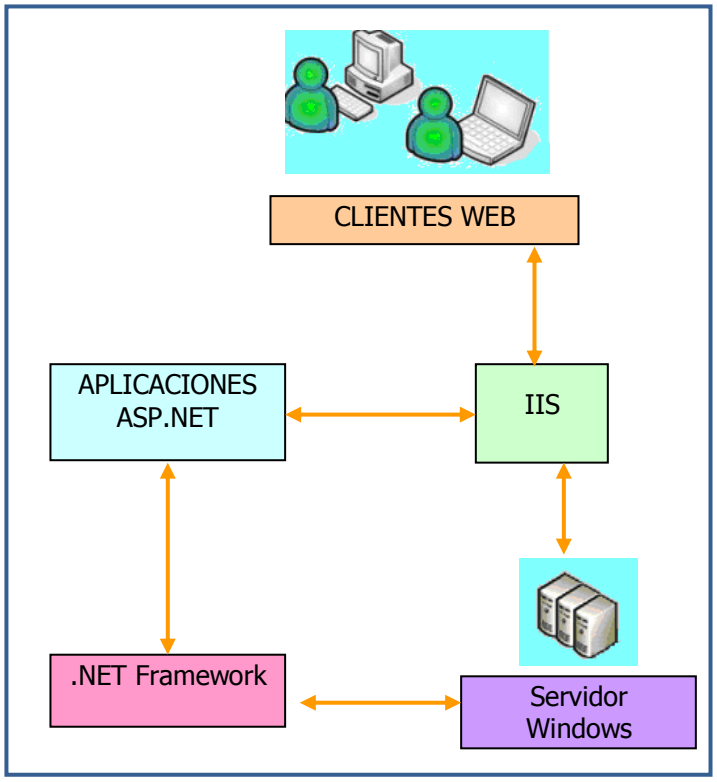


Fig. 5.9 Arquitectura de seguridad de ASP.NET

Fuente: http://i3.msdn.microsoft.com/Platform/MasterPages/Library/Library-bn1907.2.css

5.7.2 Autenticación y Autorización

ASP.NET proporciona una serie de mecanismo de autorización mediante declaraciones y programación que pueden utilizarse junto con una gran variedad de esquemas de autenticación. De este modo se puede desarrollar una estrategia de autorización exhaustiva y otra que pueda configurarse para ofrecer diferentes grados de funcionalidad: Por ejemplo, por usuario o por grupo, basado en funciones.

AUTENTICACIÓN.- Ayuda a comprobar que el usuario es precisamente quien dice ser. La aplicación obtiene las credenciales (diversas formas de identificación, como nombre y contraseña) de usuario, y las valida consultando a una autoridad determinada. Si las credenciales son válidas, se considera a la entidad que ha enviado las credenciales como una entidad autenticada.

AUTORIZACIÓN.- Limita los derechos de acceso mediante la concesión o negación de permisos específicos a una identidad autenticada.
ASP.NET dispone de funciones predefinidas de gran valor relacionadse con la autenticación y la autorización basadas en roles. Se puede configurar una aplicación ASP.NET sin más que asignar usuarios y roles en cada directorio (aplicación o sub-aplicación). Además, se puede utilizar las funciones de ASP.NET para negar el acceso a una aplicación en función del usuario o del rol.

5.7.3 Seguridad de las Aplicaciones WEB EN IIS Y ASP.NET

La seguridad de las páginas web empieza por el servidor web, en este caso IIS. Puesto que se trata de un servicio basado en Windows, IIS está totalmente integrado en la seguridad de Windows. Como en el caso de cualquier otro proceso, para obtener acceso a un archivo IIS necesita autenticación apropiada. Cuando los usuarios envían una solicitud desde su explorador a IIS, este debe leer el archivo de una carpeta Windows y sea cual sea la autenticación definida por Windows que se aplica a dicho archivo, se deberá también aplicar a IIS. Es decir, para obtener acceso a un recurso, IIS debe proporcionar las credenciales adecuadas como cualquier otro proceso.

Cuando se ejecuten las aplicaciones web, lo harán en ASP.NET, que tiene sus propios medios de seguridad. Estos entran en juego cuando la aplicación requiere acceso a los recursos. Por ejemplo, si desea leer o escribir un archivo en la aplicación web, es el contexto de seguridad de ASP.NET, lo que determina si la solicitud procederá con éxito o no. Sin embargo, no todos los usuarios dispondrán de la autenticación apropiada para leer archivos en un servidor Windows,

específicamente, en las aplicaciones disponibles de forma pública en Internet. Por tanto, IIS y ASP.NET proporcionan varios mecanismos para establecer la autenticación.

Acceso anónimo. Para las aplicaciones que recibirán solicitudes de usuarios anónimos (generalmente, aplicaciones web públicas), IIS admite un usuario "anónimo", es decir, un usuario sin credenciales de autenticación. En esta situación, el servidor en el que se ejecuta IIS dispone de un usuario de Windows adicional, con el nombre de equipo. Esta cuenta suele definirse con derechos de acceso muy restringidos.

Cuando IIS recibe una solicitud de un usuario desconocido, IIS pasa la solicitud a Windows utilizando el nombre de usuario ***anonymous*** como credenciales. Es decir IIS representa a los usuarios anónimos con la finalidad de obtener acceso al recurso.

Autenticación básica e implícita.- IIS admite asimismo el modelo de autenticación básico estándar de web. En esta situación, se solicita a los usuarios sin credenciales que proporcionen un nombre de usuario y una contraseña. Dichos nombres de usuarios y contraseña se devuelven a IIS, y pasan a estar disponibles para la aplicación. La autenticación básica proporciona un método útil para ofrecer acceso restringido a una aplicación web pública. Sin embargo, no es totalmente segura, ya que el usuario pasa un nombre de usuario y una contraseña a IIS en forma de texto normal.

La autenticación implícita es similar a la básica, pero utiliza cifrado para enviar la información del usuario al servidor. Sin embargo, este estilo de autenticación requiere un controlador de dominio de Windows.

Seguridad integrada de Windows.- Si el usuario que efectúa la solicitud ya se ha autenticado en una red basada en Windows, IIS puede pasar las credenciales del usuario cuando solicita el acceso a un recurso. Las credenciales no incluyen el nombre de usuario y la contraseña, únicamente un símbolo cifrado que indica el estado de seguridad del usuario. Se trata de una situación típica en una intranet corporativa, en la que los usuarios inician la sesión en la red antes de utilizar la

aplicación. No obstante, la seguridad integrada no es práctica en el caso de aplicaciones que deben pasar a través de servidores de seguridad corporativos. Incluso dentro de la misma red, la seguridad integrada tiene limitaciones para obtener acceso a recursos que estén en equipos remotos.

Autenticación por certificado.- Finalmente, IIS admite el uso de certificados digitales y la capa de ***sockets*** seguros (***SSL***). En esta situación, el servidor o el cliente tienen un certificado obtenido de un tercero. El certificado se pasa a la aplicación junto con las solicitudes. Se puede asignar a una cuenta de usuario de Windows y concederle los permisos apropiados. Esta autenticación tiene la ventaja de que puesto que es la autoridad del certificado quien valida cada solución o respuesta, se utiliza en aplicaciones que precisan de una relación de confianza. [WEB 5.9]

ASP.NET, conjuntamente con Microsoft Internet Information Services (IIS), puede autenticar las credenciales del usuario como nombres y contraseñas mediante los métodos de autenticación siguientes:

- Windows: básica, implícita, y Autenticación de Windows integrada
- Autenticación mediante formularios, con la que crea una página de inicio de sesión y se administra la autenticación en la aplicación.
- Autenticación mediante certificados de cliente