UNIVERSIDAD NACIONAL DE CHIMBORAZO

FACULTAD DE INGENIERÍA

ESCUELA DE INGENIERÍA EN SISTEMAS & COMPUTACIÓN

"Trabajo de grado previo a la obtención del Título de Ingeniero en Sistemas & Computación".

TRABAJO DE GRADUACION

Título del proyecto:

DESARROLLO DEL TOOLBOX MAGO DEVELOPER COMPONENT PARA LA PLATAFORMA DE DESARROLLO ASP.NET. CASO APLICATIVO: SISTEMA DE GESTIÓN DE RECURSOS HUMANOS UNACH.

Autor:

Fredy Geovanny Camacho León

Director:

Ing. Anita Elizabeth Congacha

Riobamba - Ecuador

2011

Los miembros del Tribunal de Graduación del proyecto de investigación de título: DESARROLLO DEL TOOLBOX MAGO DEVELOPER COMPONENT PARA LA PLATAFORMA DE DESARROLLO *ASP.NET*. CASO APLICATIVO: SISTEMA DE GESTIÓN DE RECURSOS HUMANOS *UNACH*, presentado por: Fredy Geovanny Camacho León y dirigido por: Ing. Anita Elizabeth Congacha.

Una vez escuchada la defensa oral y revisado el informe final del proyecto de investigación con fines de graduación escrito en la cual se ha constatado el cumplimiento de las observaciones realizadas, remite el presente para uso y custodia en la biblioteca de la Facultad de Ingeniería de la *UNACH*.

Para constancia de lo expuesto firman:

Presidente del Tribunal (nombre)	*Firma*
Director de Tesis (nombre)	*Firma*
Miembro del Tribunal (nombre)	*Firma*

AUTORÍA DE LA INVESTIGACIÓN

La responsabilidad del contenido de este Proyecto de Graduación, nos corresponde exclusivamente a: Fredy Geovanny Camacho León (autor) e Ing. Anita Elizabeth Congacha (director) y el patrimonio intelectual de la misma a la Universidad Nacional de Chimborazo.

DEDICATORIA

El presente trabajo de investigación lo dedico de manera muy especial a toda mi familia por el apoyo que siempre me han sabido brindar, guiándome por el sendero del bien para ser una buena persona y proporcionándome todo lo fundamental para ser un profesional.

AGRADECIMIENTO

A nuestro padre celestial Jehová Dios rey del universo por darme la vida, la salud y la fortaleza para conseguir mis metas anheladas.

Mi infinita gratitud a la magna Universidad Nacional de Chimborazo, Facultad de Ingeniería, Escuela de Sistemas & Computación, por acogerme en sus aulas y brindarme los conocimientos requeridos por un profesional.

i. Índice de contenidos

ii. Índice de recursos gráficos

iii. Listado de siglas y abreviaturas

Definición de término	Abreviatura (A - Z)
Objeto de acceso a datos.	ADO
Interface de programación de aplicaciones.	API
Páginas de servidor activo.	ASP
Capa de lógica de negocios.	BLL
Tiempo de ejecución en lenguaje común.	CLR
Especificación de lenguaje común.	CLS
Modelo de componentes de objeto.	COM
Hojas de estilo en cascada.	CSS
Sistema de tipo común.	CTS
Capa de acceso a datos.	DAL
Librería de vínculos dinámicos.	DLL
Asociación de fabricantes de computadoras internacional.	ECMA
Interface gráfica de usuario.	GUI
Lenguaje de marcado de hipertexto.	HTML
Entorno de desarrollo integrado.	IDE
Internet Information Services.	IIS
Depurador justo a tiempo (*Just-In-Time*).	JIT
Lenguaje de consulta integrado.	LINQ
Lenguaje intermedio de Microsoft.	MSIL
Kit de desarrollo de software Windows.	SDK
World Wide Web Consortium.	W3C
Instrumental de administración de Windows.	WMI
Lenguaje de marcado de hipertexto extensible.	XHTML
Lenguaje de marcado extensible.	XML

iv. Resumen

Enfocando el estudio al entorno de desarrollo de aplicaciones web con *ASP.NET* se encuentran intrínsecamente muchísimas limitaciones enmarcadas en relación a objetos necesarios sin referencia alguna hasta el momento, carencia de algunas propiedades necesarias en algunos objetos, limitantes de algunos eventos de acceso a servidor para la manipulación de dichos objetos, eventos y scripts enviados a través de aplicaciones cliente, basados en performance *callback* (rendimiento en devoluciones de llamadas asíncronas), aplicación de los nuevos estándares en la creación de hojas de estilo *CSS* basados en la versión *CSS3* y *HTML5*, entre otros aspectos importantes.

……

Por ende se desarrollará las bibliotecas de clases (*archivos .dll*), es decir crear componentes propios, ensamblarlos a partir de los sofisticados mecanismos de las clases de .NET Framework, basados en la versión 2.0 o superior.

……

Mediante la creación del Toolbox Mago Developer Component se obtendrá ordenación y agrupación semántica de código, código reutilizable en forma de objetos, manipulación de objetos de servidor mediante eventos basados en performance *callback* (rendimiento en devoluciones de llamadas asíncronas), colección jerárquica de scripts y eventos de aplicación cliente basados en *ECMAScript 262 5ta. Edición* y performance *callback*, creación de nuevos e innovadores objetos con compatibilidad para *CSS3*, múltiples servicios web, recursos *XML* y agentes de usuario para visualización de resultados.

……

Se ha seleccionado el *Método Deductivo* dentro de las metodologías de investigación existentes, porque para la investigación se parte de la documentación integrada en el entorno de desarrollo de Microsoft Visual Studio 2010 Ultimate (*MSDN Library*), que es la base de la cual se parte para el desarrollo del proyecto.

Luego del desarrollo del proyecto de investigación se ha obtenido un total de 26 nuevos e innovadores controles que cumplen con criterios de funcionalidad, practicidad, estandarización y usabilidad para su utilización en el Toolbox del entorno de desarrollo de *ASP.NET* basado en .NET Framework 2.0 o superior.

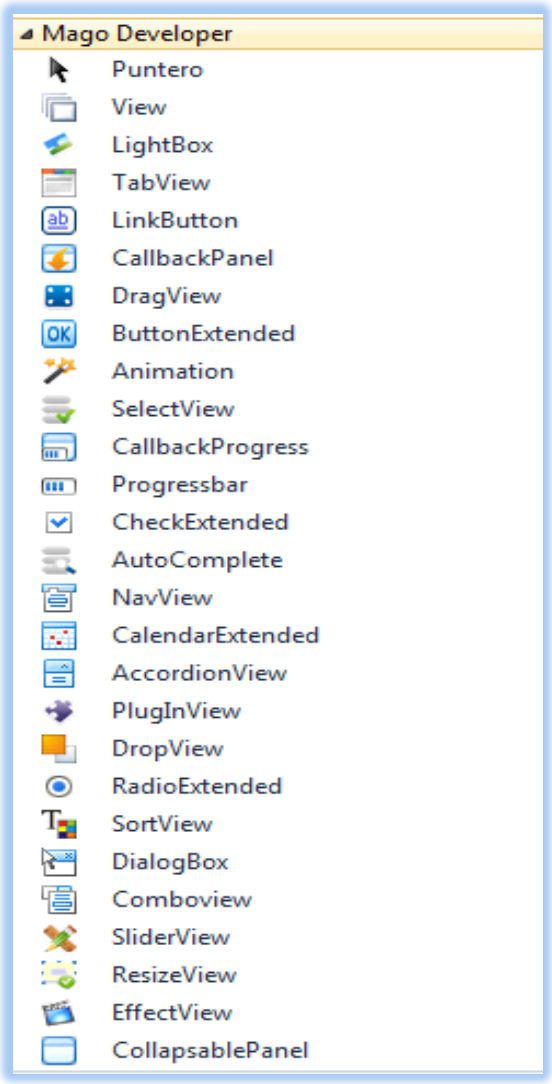

Figura 1. Toolbox Mago Developer Component.

v. Summary

Focusing the study to the development environment with *ASP.NET* web applications are intrinsically framed many limitations in relation to necessary objects without any reference so far, lack of some properties needed in some objects, limiting some access event server handling such events, and scripts sent through client applications based on *callback* performance (performance asynchronous *callbacks*), application of new standards in creating *CSS* based version *CSS3* and *HTML5* among other important topics.

……

Therefore be developed class libraries (*dll files*.), ie create your own components, and assemble them from the sophisticated mechanisms of classes .NET Framework based on version 2.0 or higher.

……

Through the creation of the Toolbox Mago Developer Component ordering and grouping is obtained semantic code, reusable code as objects, object manipulation server using *callback* events based on performance (performance asynchronous *callbacks*), hierarchical collection of scripts and events client application based on *ECMAScript 262 5th editing* and *callback* performance, creating new and innovative objects with support for *CSS3*, multiple web services, *XML* resources and user agents to display results.

……

You select the *Deductive Method* in existing research methodologies, because research is part of the integrated into the development environment Microsoft Visual Studio 2010 (*MSDN Library*), which is the basis of which party to project development.

After the development of the research project has been awarded a total of 26 new and innovative controls that meet the criteria of functionality, practicality, standardization and usability for use in the Toolbox of *ASP.NET* development environment based on .NET Framework 2.0 or higher.

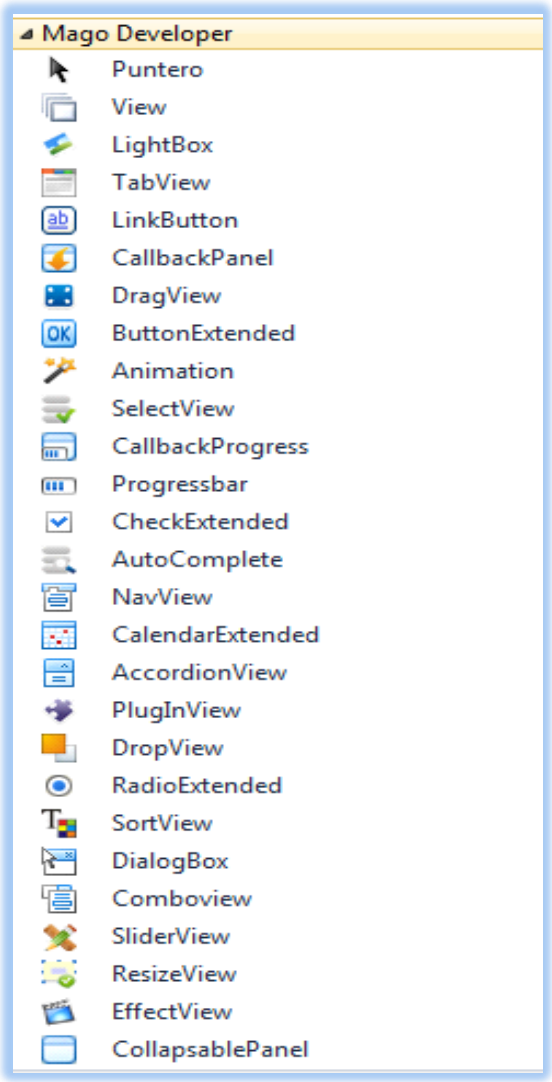

Figura 2. Toolbox Mago Developer Component.

1. Introducción

Dado el creciente y abrumador avance de las tecnologías de la información, las redes y la interconectividad a través de la red mundial de comunicaciones entre ordenadores (internet), los múltiples requerimientos que contemplados en las nuevas plataformas y entornos de desarrollo, tanto para las aplicaciones de escritorio, como para las aplicaciones web, *silverlight*, *WPF* entre otras de hoy en día; y tomando en cuenta que se cuenta con componentes o herramientas (Toolbox) de desarrollo con un amplio entorno de aplicación, pero aún limitados en algunos aspectos y sin ninguna referencia en otros.

Enfocando el estudio al entorno de desarrollo de aplicaciones web con *ASP.NET* se encuentran intrínsecamente muchísimas limitaciones enmarcadas en relación a objetos necesarios sin referencia alguna hasta el momento, carencia de algunas propiedades necesarias en algunos objetos, limitantes de algunos eventos de acceso a servidor para la manipulación de dichos objetos, eventos y scripts enviados a través de aplicaciones cliente basados en performance *callback* (rendimiento en devoluciones de llamadas asíncronas), aplicación de los nuevos estándares en la creación de hojas de estilo *CSS* basados en la versión *CSS3* y *HTML5*, entre otros aspectos importantes.

Por ende se desarrollará las bibliotecas de clases (*archivos .dll*), es decir crear componentes propios, ensamblarlos a partir de los sofisticados mecanismos de las clases de .NET Framework, basados en la versión 2.0 o superior.

Si se desea que los componentes se puedan utilizar en otros lenguajes de programación, se deben crearlos en un lenguaje conforme a *CLS* (*Common Language Specification*) y asegurarse de que todos los miembros públicos y protegidos son compatibles con *CLS*.

SDK (*Kit de desarrollo de software Windows*) proporciona compiladores para cuatro lenguajes conformes a *CLS*: *Visual Basic*, *C#*, *C++* y *J#*.

Mediante la creación del Toolbox Mago Developer Component se obtendrá ordenación y agrupación semántica de código, código reutilizable en forma de objetos, manipulación de objetos de servidor mediante eventos basados en performance *callback* (rendimiento en devoluciones de llamadas asíncronas), colección jerárquica de scripts y eventos de aplicación cliente basados en *ECMAScript 262 5ta. Edición* y performance *callback*, creación de nuevos e innovadores objetos con compatibilidad para *CSS3*, múltiples servicios web, recursos *XML* y agentes de usuario para visualización de resultados.

Se ha seleccionado el *Método Deductivo* dentro de las metodologías de investigación existentes, porque para la investigación se parte de la documentación integrada en el entorno de desarrollo de Microsoft Visual Studio 2010 Ultimate (*MSDN Library*), que es la base de la cual se parte para el desarrollo del proyecto.

Se toma como referencia esencial el estudio minucioso y detenido de la plataforma de desarrollo .NET Framework (la disponibilidad y compatibilidad entre versiones).

CAPÍTULO I

1 FUNDAMENTACIÓN TEÓRICA

1.1 Antecedentes de la investigación.

El autor ha desarrollado anteriormente los siguientes tipos de proyectos de biblioteca de clases o componentes *dll*:

- Componentes (*archivos .dll*) para la gestión de la información y métodos de manipulación de la misma detallados como métodos de búsquedas personalizadas, métodos de eliminación, actualización e inserción para entornos de aplicaciones *Windows Forms* n-capas.

- Componente (*archivo .dll*) para el entorno de desarrollo de *ASP.NET*, este componente se encarga de la gestión y visualización de archivos *.pdf*, basados en <*iframe*></*iframe*> tags.

- Componentes (*archivos .dll*) para la gestión de la información y métodos de manipulación de la misma detallados como métodos de búsquedas personalizadas, métodos de eliminación, actualización e inserción para entornos de aplicaciones *ASP.NET* n-capas.

1.2 Conceptos introductorios a las tecnologías de desarrollo.

✔ Microsoft Visual Studio 2010 Ultimate

Microsoft Visual Studio 2010 Ultimate es un conjunto de herramientas de desarrollo basadas en componentes y otras tecnologías para compilar aplicaciones eficaces de alto rendimiento, está optimizado para el diseño, el desarrollo y la implementación.

Visual Basic, *Visual C#* y *Visual C++* utilizan todos el mismo entorno de desarrollo integrado (*IDE*), que habilita el uso compartido de herramientas y facilita la creación de soluciones en varios lenguajes, dichos lenguajes utilizan las funciones de .NET Framework, las cuales ofrecen acceso a tecnologías clave para simplificar el desarrollo de aplicaciones.

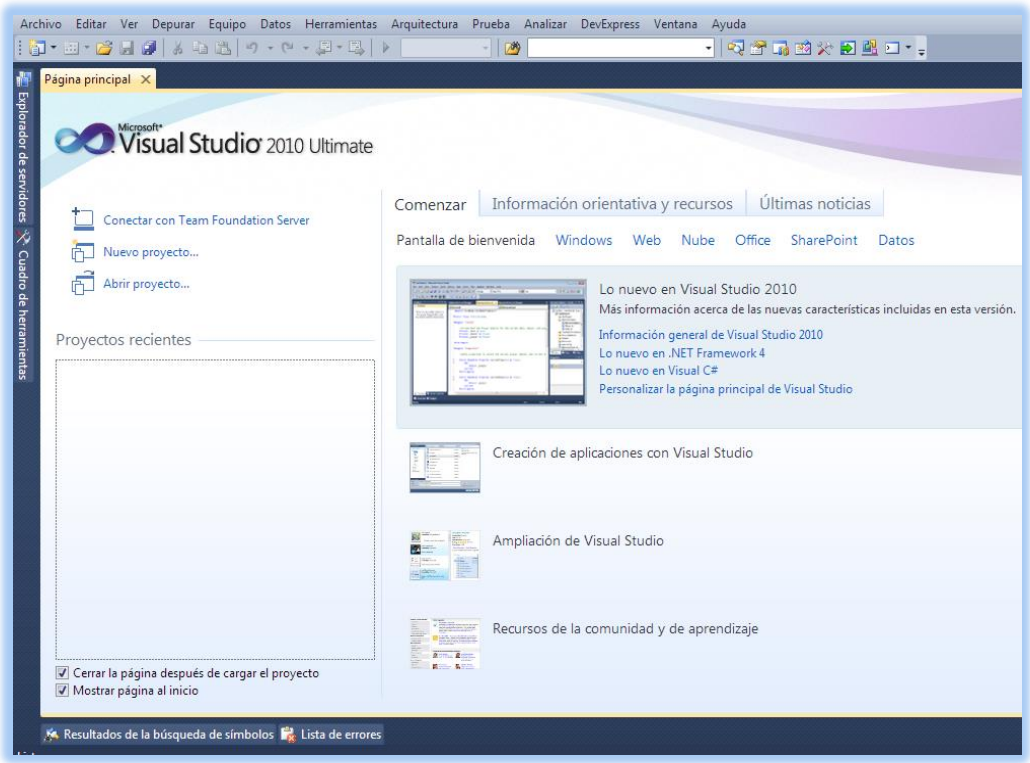

Figura 3. Entorno de desarrollo de Microsoft Visual Studio 2010 Ultimate.

- **.NET Framework**

.NET Framework, es un componente que forma la parte integral de Windows y que es compatible con la compilación y ejecución de las aplicaciones de próxima generación, así como con servicios web y *XML*.

- **Lenguaje de programación Visual C#**

Lenguaje de programación diseñado para crear aplicaciones empresariales que se compilan en .NET Framework. *C#*, que es una evolución de *C* y *C++*, garantiza la seguridad de tipos y está orientado a objetos.

- **ADO.NET**

Conjunto de tecnologías de acceso a datos incluidas en las bibliotecas de clases de .NET Framework que proporcionan acceso a datos relacionales y a *XML*.

- **ASP.NET**

Conjunto de tecnologías de Microsoft .NET Framework para la creación de aplicaciones y servicios web, las páginas *ASP.NET* se ejecutan en el servidor y generan lenguaje de marcado (como *HTML* o *XML*) que se envía a un explorador móvil o de escritorio.

- **Common Language Runtime**

Motor, que es el núcleo de la ejecución de código administrado, el motor en tiempo de ejecución proporciona al código administrado servicios como integración entre varios lenguajes, seguridad de acceso a código, administración de la duración de los objetos, y compatibilidad con la depuración y la generación de perfiles.

- **Metadatos**

Información que describe todos los elementos administrados por *Common Language Runtime*: un ensamblado, el archivo cargable, el tipo, el método, etc.

Esto puede incluir información necesaria para la depuración y la recolección de elementos no utilizados, así como atributos de seguridad, cálculo de referencias de datos, definiciones extendidas de clases y miembros, enlace de versión y otra información requerida por el motor en tiempo de ejecución.

- **Lenguaje Intermedio de Microsoft (MSIL)**

Lenguaje utilizado como salida de una serie de compiladores y como entrada para un compilador *Just-In-Time* (*JIT*). *Common Language Runtime* incluye un compilador *Just-In-Time* para convertir *MSIL* a *código nativo*.

- **Código nativo**

Compilación en código máquina específico del procesador.

- **Lenguaje de Marcado Extensible (*XML*)**

Subconjunto del lenguaje de marcado generalizado estándar (*SGML*) optimizado para su uso a través de la web. *XML* proporciona un método uniforme para describir e intercambiar datos estructurados que es independiente de las aplicaciones o los proveedores.

- **Clase**

Tipo de referencia que encapsula datos (constantes y campos) y el comportamiento de (métodos, propiedades, indizadores, eventos, operadores, constructores de instancia, constructores estáticos y destructores), y puede contener tipos anidados, *los tipos de clase admiten la herencia.*

- **Estructura**

Tipo de valor definido por el usuario, al igual que una clase, las estructuras pueden contener constructores, constantes, campos, métodos, propiedades, indizadores,

operadores y tipos anidados, sin embargo, a diferencia de las clases, *las estructuras no admiten la herencia*.

Pilares de la Programación Orientada a Objetos

1. **Herencia**

Es un mecanismo mediante el cual una clase derivada puede extender y especializar una clase base.

- ✔ Es un tipo de relación entre clases.
- ✔ Va de la generalización a la especialización.
- ✔ Clase base **:** clase derivada.

2. **Encapsulación**

Posibilidad de que un objeto oculte sus datos y métodos internos, haciendo que solo sean accesibles mediante programación las partes deseadas del objeto.

3. **Polimorfismo (varias formas)**

Esto facilita la programación de funciones o procedimientos (métodos) que ejecutarán acciones que dependerán de los objetos de los que se apliquen, es decir las clases base pueden definir e implementar métodos virtuales y las clases derivadas pueden invalidarlos, lo que significa que proporcionan su propia definición e implementación.

- ✔ La definición del método reside en la clase base.
- ✔ La implementación del método reside en la clase derivada.
- ✔ La invocación es resuelta en tiempo de ejecución.

4. **Abstracción**

- ✔ Ignorancia selectiva.
- ✔ Decide que es importante y que no lo es.
- ✔ Se enfoca [*depende*] en lo que es importante.
- ✔ Ignora [*no depende*] de lo que no es importante.
- ✔ Se utiliza el concepto de encapsulación para reforzar la abstracción.

✔ **Evento**

En *WMI*, aparición de un cambio en los datos estáticos o dinámicos relacionado con un objeto administrado.

✔ **Delegado**

En .NET Framework, referencia a una función, un delegado es equivalente a un puntero a una función.

✔ **Método**

En *WMI*, función que describe el comportamiento de una clase, la inclusión de un método en una clase no garantiza una implementación del método, un método también es una función incluida en una interface *WMI*.

✔ **Métodos Estáticos**

- ✓ Pueden acceder a datos compartidos por todas las instancias de la clase.
- ✓ Son invocados en la clase, no en el objeto.
- ✓ No es necesaria la creación de una instancia para invocarlos.

- **Métodos Abstractos**

  - ✓ Solo en clases abstractas.
  - ✓ No pueden contener implementación.
  - ✓ Deben ser implementados por las clases derivadas.
  - ✓ Los métodos abstractos son virtuales.
  - ✓ Los métodos abstractos pueden sobrescribir métodos de la clase base declarados como virtuales "*virtual -> C#"*.
  - ✓ Los métodos abstractos pueden sobrescribir métodos de la clase base declarados como "*override -> C#*".

- **Métodos Virtuales**

  - ✓ Es un método que la clase base permite que sea sobrescrito en una clase derivada.
  - ✓ Un método no-virtual es la *única* implementación posible para ese método.

- **Enum (enumeración)**

Lista de constantes con nombre.

- **Campo**

Miembro que representa una variable asociada con un objeto o una clase.

- **Indizador**

Miembro de una clase que permite tener acceso a las instancias de una clase o una estructura de la misma manera que las matrices, los indizadores son similares a las propiedades excepto en que los métodos descriptores de acceso *get* y *set* de los indizadores toman parámetros, mientras que los métodos de descriptor de acceso de las propiedades no los toman.

- **Interface**

Tipo de referencia que define un contrato, otros tipos implementan una interface para garantizar que admiten ciertas operaciones, la interface especifica los miembros de clases u otras interfaces que los implementan y que deben suministrar, al igual que las clases, las interfaces pueden contener como miembros métodos, propiedades, indizadores y eventos.

- ✓ Contienen solo métodos sin implementación.
- ✓ Describen un "*contrato*".
- ✓ No heredan atributos.
- ✓ No se pueden crear instancias de una interface.
- ✓ Las clases derivadas deben implementar todas las operaciones heredadas.

- **Propiedad**

En .NET Framework, miembro de clase que es como un campo público, pero que incluye características como control de versiones, encapsulación y la posibilidad de ejecutar otra lógica adicional mediante los métodos de descriptor de acceso *get* y *set*.

- **Atributo**

Declaración descriptiva que se puede aplicar a elementos de programación como tipos, campos, métodos y propiedades, los atributos se guardan con los metadatos de un archivo de .NET Framework y pueden utilizarse para describir el código para *Common Language Runtime* o para influir en el comportamiento de la aplicación en tiempo de ejecución.

- **Método genérico**

Método cuya definición tiene marcadores de posición, llamados parámetros de tipo genérico, para uno o varios tipos utilizados en el cuerpo del método o como tipos de los parámetros del método, el usuario especifica los tipos reales (argumentos de tipo

genérico) para los parámetros de tipo al llamar al método genérico, un método sólo es genérico si tiene sus propios parámetros de tipo.

- **Tipo genérico**

Clase, interface o estructura cuya definición tiene marcadores de posición, llamados parámetros de tipo genérico, para uno o varios tipos utilizados en sus definiciones de miembros (por ejemplo, como tipos de parámetro de método), el usuario especifica los tipos reales (argumentos de tipo genérico) para los parámetros de tipo al crear una instancia de tipo genérico.

- **Globalización**

Proceso de diseñar y programar un producto de software de manera que funcione en varias configuraciones regionales.

- **Tipo de referencia**

Tipo de datos representado por una referencia (*similar a un puntero*) al valor real del tipo, si se asigna un tipo de referencia a una variable, esa variable hace referencia (o "*señala*") *al valor original*, no se realiza ninguna copia, los tipos de referencia se componen de clases, interfaces, delegados y tipos de valor con conversión boxing.

- **Tipo de valor**

Tipo de datos representado por el valor real del tipo, si se asigna un tipo de valor a una variable, esa variable obtiene una copia reciente del valor, los tipos de valor normalmente se crean en un marco de pila de método, en lugar de crearse en la pila de recolección de elementos no utilizados.

- **Boxing**

Conversión de una instancia de tipo de valor a un objeto, para ello, hace una copia del tipo de valor y la incrusta en un objeto recién asignado.

- **Unboxing**

Conversión de la instancia de un objeto a un tipo de valor.

- **Contravarianza**

En los lenguajes de programación, es la capacidad para usar un tipo menos derivado que el especificado originalmente.

- **Covarianza**

En los lenguajes de programación, es la capacidad para usar un tipo más derivado que el especificado originalmente.

La covarianza y la contravarianza en las interfaces genéricas y los delegados permiten la conversión implícita de los parámetros de tipo genérico, la covarianza y la contravarianza también se admiten en los delegados no genéricos para hacer coincidir las firmas de método con los tipos de delegado.

- **Sobrecarga**
  - ✓ Definir más de un método por cada mensaje, los tipos de argumentos ayudan a decidir a qué mensaje se invoca.
  - ✓ Tareas similares son realizadas por métodos con un mismo nombre.
  - ✓ Simplifican la tarea del desarrollador, al no tener que recordar distintos nombres para comportamientos iguales.
- **Ocultamiento**
  - ✓ Esconde un método idéntico.
  - ✓ Introduce un nuevo método a la jerarquía de la clase.
  - ✓ Oculta los métodos virtuales y no-virtuales.
  - ✓ Esconde métodos con firmas idénticas.
  - ✓ Esconde campos con el mismo identificador.

1.3 Introducción al desarrollo de componentes

✔ Componente

En .NET Framework, un componente es una clase que implementa la interface *System.ComponentModel.IComponent* o se deriva directa o indirectamente de una clase que implementa *IComponent*, en programación, el término componente se utiliza normalmente para objetos que se pueden volver a utilizar y que pueden interactuar con otros objetos, un componente de .NET Framework cumple esos requisitos generales y además dispone de características como control sobre recursos externos y compatibilidad en tiempo de diseño.

✔ Control sobre recursos externos

La interface *IComponent* extiende la interface *IDisposable*, que tiene un método denominado *Dispose* en su contrato.

En su implementación el método *Dispose*, un componente debe liberar los recursos externos de forma explícita, esta es una manera determinista de liberar recursos, en contraste con la limpieza no determinista predeterminada que sucede durante la recolección de elementos no utilizados, además, un componente derivado debe invocar al método *Dispose* de su clase base.

Nota: Aunque proporcione control explícito sobre los recursos mediante *Dispose*, siempre debe realizar una limpieza implícita mediante el finalizador (destructor) para evitar que los recursos se pierdan de forma permanente si un usuario no llama a *Dispose* en el componente.

✔ Compatibilidad en tiempo de diseño

Un componente se puede agregar al cuadro de herramientas (Toolbox) de Microsoft Visual Studio 2010 Ultimate, se puede arrastrar y colocar en un formulario, y se puede manipular en una superficie de diseño, obsérvese que .NET Framework incorpora compatibilidad en tiempo de diseño con los tipos *IComponent*.

✔ Hospedar un componente

Un componente se puede ubicar (*hospedar*) en un contenedor, cuando se ubica un componente, interactúa con el contenedor a través de su sitio y puede consultar y obtener servicios de su contenedor a través del mismo.

Para asegurarse de que los recursos se liberan cuando el contenedor deje de usarse, el contenedor debe implementar la interface *IDisposable*, en su implementación del método *Dispose*, un contenedor debe liberar todos los recursos que mantiene e invocar al método *Dispose* de cada uno de los componentes que contiene.

La contención es lógica y no necesita tener representación visual, un ejemplo de contención no visual es un contenedor de nivel medio que hospeda componentes de base de datos, la contención visual se ve en los diseñadores de *Windows Forms* y *Web Forms*.

✔ Calcular las referencias de un componente

Los componentes pueden ser de uso remoto o de uso no remoto.

Las referencias de los componentes de uso remoto se calculan por referencia o por valor, el cálculo de referencias implica el envío de objetos más allá de los límites, como dominios de aplicación (*procesos ligeros*), procesos e incluso equipos.

Cuando las referencias de un objeto se calculan por referencia, se crea un proxy que realiza llamadas remotas al objeto, cuando las referencias de un objeto se calculan por valor, se envía una copia serializada del objeto a través de los límites relevantes.

Las referencias de los componentes de uso remoto que encapsulan recursos del sistema, que son grandes o que existen como instancias únicas, se deben calcular por referencia, la clase base de los componentes cuyas referencias se calculan por referencia es *System.ComponentModel.Component*, esta clase base implementa la interface *IComponent* y deriva de *MarshalByRefObject*, muchos componentes de la biblioteca de clases de .NET Framework se derivan de *Component*, incluidos *System.Windows.Forms.Control* (la clase base de los formularios de Windows), *System.Web.Services.WebService* (la clase base de los servicios web *XML* creados mediante *ASP.NET*) y *System.Timers.Timer* (una clase que genera eventos recurrentes).

Las referencias de los componentes de uso remoto que sólo guardan un estado se deben calcular por valor, la clase base de los componentes cuyas referencias se calculan por valor es *System.ComponentModel.MarshalByValueComponent*, esta clase base implementa la interface *IComponent* y deriva de *Object*, sólo unos pocos componentes de la biblioteca de clases de .NET Framework derivan de *MarshalByValueComponent* todos esos componentes están en el espacio de nombres *System.Data* (*DataColumn*, *DataSet*, *DataTable*, *DataView* y *DataViewManager*).

Nota: Las clases base de los objetos cuyas referencias se calculan por valor y por referencia son *Object* y *MarshalByRefObject*, respectivamente, pero las clases derivadas correspondientes se denominan *MarshalByValueComponent* y *Component*, la razón de la utilización de este esquema de nombres es que el tipo utilizado más habitualmente tiene el nombre más sencillo.

Si un componente no va a ser remoto, no debe derivarse de las implementaciones base de *Component*; en su lugar, implemente *IComponent* directamente.

✔ Control

Un control es un componente que proporciona o habilita funciones de interface de usuario, .NET Framework dispone de dos clases base para controles: una para controles de formularios *Windows Forms* de cliente y otra para controles de servidor de *ASP.NET*, estas son *System.Windows.Forms.Control* y *System.Web.UI.Control*.

Todos los controles de la biblioteca de clases de .NET Framework se derivan directa o indirectamente de estas dos clases. *System.Windows.Forms.Control* se deriva de *Component* y proporciona funciones de interface de usuario. *System.Web.UI.Control* implementa *IComponent* y proporciona la infraestructura en la se puede agregar fácilmente funcionalidad de interface de usuario.

Nota Importante: *Todos los controles son componentes, pero no al revés.*

✔ Contenedor y sitio

Si se va a programar componentes y controles para formularios *Windows Forms* o para páginas de formularios *Web Forms* (páginas de *ASP.NET*), no es necesario implementar contenedores o sitios, los diseñadores de *Windows Forms* y de formularios *Web Forms* son contenedores de formularios *Windows Forms* y de los controles de servidor de *ASP.NET*, los contenedores proporcionan servicios a los componentes y controles hospedados en ellos, en tiempo de diseño, los controladores se hospedan en el diseñador y obtienen servicios del mismo.

✓ **Container**

Un contenedor es una clase que implementa la interface *System.ComponentModel.IContainer* o que deriva de una clase que implementa esta interface, de forma lógica, un contenedor contiene uno o varios componentes, que se denominan componentes secundarios del contenedor.

✓ **Site**

Un sitio es una clase que implementa la interface *System.ComponentModel.ISite* o que deriva de una clase que implementa esta interface, un contenedor proporciona los sitios para administrar y comunicarse con sus componentes secundarios.

Normalmente, un contenedor y un sitio se implementan como una unidad.

1.4 Referencias imprescindibles en el desarrollo de componentes

- Si la clase usa recursos externos pero no se va a emplear en una superficie de diseño, se debe implementar *System.IDisposable* o derivar de una clase que implemente *IDisposable* de forma directa o indirecta.

- Si la clase se va a usar en una superficie de diseño (como el diseñador de formularios *Windows Forms* o *Web Forms*), se debe implementar *System.ComponentModel.IComponent* o derivar de una clase que implemente *IComponent* de forma directa o indirecta, se debe notar que *IComponent* extiende *IDisposable*, por lo que un tipo *IComponent* es siempre un tipo *IDisposable*, un tipo *IComponent* tiene una pequeña sobrecarga de rendimiento con respecto a un tipo *IDisposable* que no sea *IComponent*, pero esto se suele compensar por la posibilidad de ubicar un tipo *IComponent* en tiempo de diseño y en tiempo de ejecución.

- Si se desea disponer de una clase que se pueda diseñar (usar en una superficie de diseño) y cuyas referencias se calculen por referencia, se puede derivar de *System.ComponentModel.Component*, *Component* es la implementación base de un tipo *IComponent* cuyas referencias se calculan por referencia.

- Si se desea disponer de una clase diseñable cuyas referencias se calculen por valor, se puede derivar de *System.ComponentModel .MarshalByValueComponent. MarshalByValueComponent* es la implementación base de un tipo *IComponent* cuyas referencias se calculan por valor.

- Si se desea introducir un tipo *IComponent* en la jerarquía de modelos de objeto y no se puede derivar de una implementación base como *Component* o *MarshalByValueComponent* debido a una herencia simple, se debe implementar *IComponent*.

- Si se desea obtener una clase que se pueda diseñar y que proporcione una interface de usuario, la clase es un control, un control debe derivarse directa o indirectamente de una de las clases de control base: *System.Windows.Forms .Control* o *System.Web.UI.Control.*

Nota: Si la clase no se puede diseñar ni contiene recursos externos, no son precisos los tipos *IComponent* e *IDisposable*.

1.5 Pasos básicos para la creación de un proyecto de biblioteca de clases.

Los componentes proporcionan código reutilizable en forma de objetos.

Una aplicación que utiliza el código de un componente para crear objetos y llamar a sus métodos y propiedades se conoce como cliente, un cliente puede estar o no en el mismo ensamblado del componente que utiliza.

1. En el menú Archivo del entorno de desarrollo se debe seleccionar Nuevo, a continuación Proyecto.
2. En la nueva ventana que aparece en pantalla se seleccionará la opción plantillas instaladas, en este caso se escogerá el lenguaje de programación *Visual C#*, después se selecciona la opción Windows, luego la plantilla biblioteca de clases, también existe una opción adicional sumamente importante que es seleccionar la plataforma de .NET Framework con la que se va a desarrollar la aplicación, en el entorno de desarrollo la opción predeterminada es la plataforma de .NET Framework 4.0, y finalmente se le dará un nombre al proyecto, en este caso se le ha denominado Client.

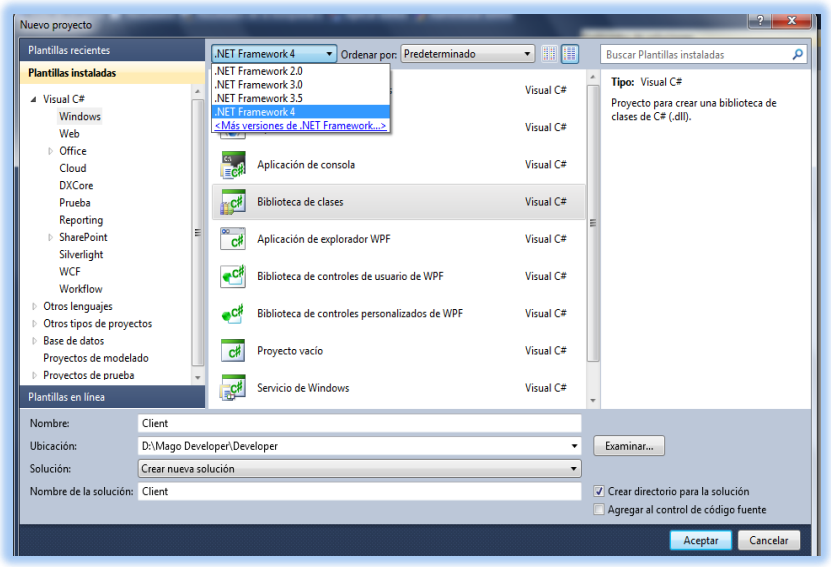

Figura 4. Creando un nuevo proyecto de biblioteca de clases (*componente*).

3. En el explorador de soluciones del entorno de desarrollo se mostrarán los siguientes archivos por defecto.

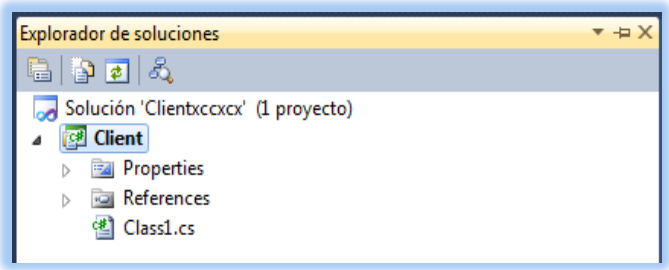

Figura 5. Explorador de soluciones por defecto en la creación de un nuevo componente.

4. Se deberá agregar la cantidad archivos necesarios y del tipo deseado al proyecto como por ejemplo archivos de clases, de información de ensamblado, de configuración, de acceso a datos, de recursos, de hojas de estilo *CSS*, de interface, de manifiestos, de recursos *XML*, de *JScript*, etc.

Todos los elementos agregados se pueden organizar jerárquicamente por carpetas, subcarpetas o como recursos libres en el explorador de soluciones.

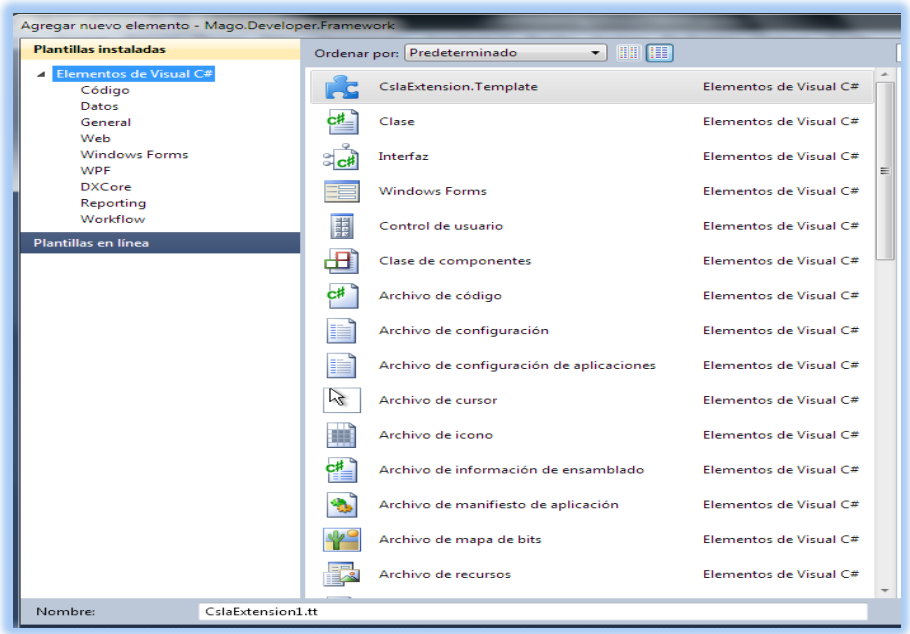

Figura 6. Agregando nuevos elementos al explorador de soluciones del proyecto.

5. Finalmente un proyecto de biblioteca de clases bastante avanzado tendrá al menos el siguiente aspecto.

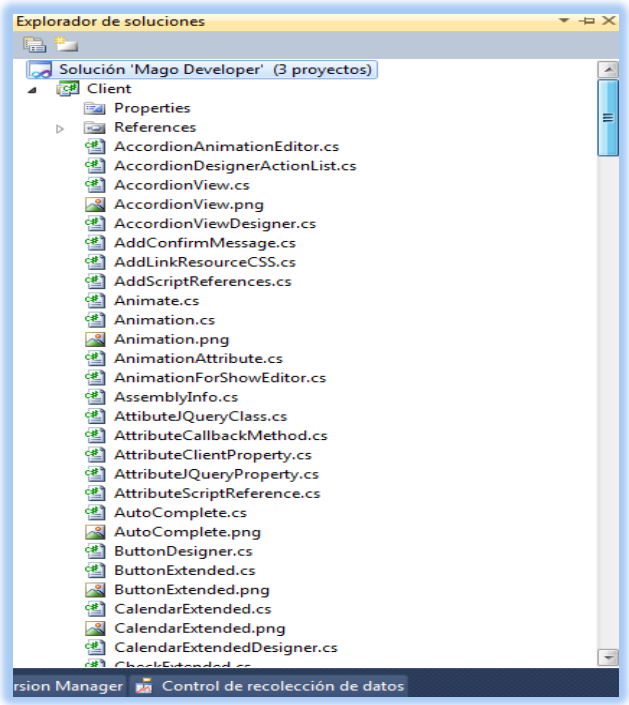

Figura 7. Explorador de soluciones de un proyecto de biblioteca de clases avanzado.

Nota Importante: No se puede iniciar directamente un proyecto con un tipo de resultado de biblioteca de clases (*componente .dll*), para depurar este tipo de proyecto, se deberá agregar a esta solución un proyecto ejecutable (*Windows Forms*), sitio web (*ASP.NET*) que haga referencia al proyecto de biblioteca, el proyecto ejecutable o sitio web se deberá establecer como proyecto de inicio.

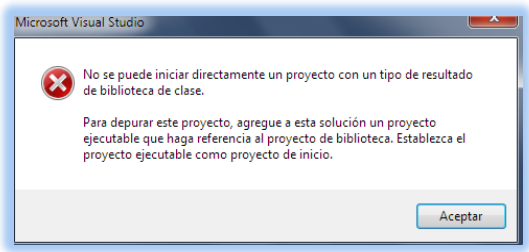

Figura 8. Compilación del proyecto de biblioteca de clases (*genera **Mago.Developer.Client.dll)***

1.6 Introducción a lenguaje *C#* 

Lenguaje de programación diseñado para crear aplicaciones empresariales que se compilan en .NET Framework. *C#*, que es una evolución de *C* y *C++*, garantiza la seguridad de tipos y está orientado a objetos.

La sintaxis de *C#* simplifica muchas de las complejidades de *C++* y proporciona características eficaces tales como tipos de valor que admiten valores *null*, enumeraciones, delegados, expresiones lambda y acceso directo a memoria, que no se encuentran en *Java*.

C# admite métodos y tipos genéricos, que proporcionan mayor rendimiento y seguridad de tipos, e iteradores, que permiten a los implementadores de clases definir comportamientos de iteración personalizados que el código usuario puede utilizar fácilmente.

Como lenguaje orientado a objetos, *C#* admite los conceptos de encapsulación, herencia, polimorfismo y abstracción, todas las variables y métodos, incluido el método *Main* que es el punto de entrada de la aplicación, se encapsulan dentro de definiciones de clase.

Una clase puede heredar directamente de una clase primaria, pero puede implementar cualquier número de interfaces, los métodos que reemplazan a los métodos virtuales en una clase primaria requieren la palabra clave *override* como medio para evitar redefiniciones accidentales.

En *C#*, un *struct* es como una clase sencilla, es un tipo asignado en la pila que puede implementar interfaces pero que no admite la herencia.

C# facilita el desarrollo de componentes de software a través de varias e innovadoras construcciones de lenguaje, entre las que se incluyen las siguientes:

- Firmas de métodos encapsulados denominadas delegados, que habilitan notificaciones de eventos con seguridad de tipos.
- Propiedades, que actúan como descriptores de acceso para variables miembro privadas.
- Atributos, que proporcionan metadatos declarativos sobre tipos en tiempo de ejecución.
- Comentarios en línea de documentación *XML*.
- *Language-Integrated Query* (*LINQ*) que proporciona funciones de consulta integradas en una gran variedad de orígenes de datos.

Si necesita interactuar con otro software de Windows, como objetos *COM* o archivos *dll* nativos de Win32, podrá hacerlo en *C#* mediante un proceso denominado "*interoperabilidad*", la *interoperabilidad* habilita los recursos de *C#* para que puedan realizar prácticamente las mismas tareas que en una aplicación *C++* nativa.

C# admite incluso el uso de punteros y el concepto de código no seguro "*unsafe*" en los casos en que el acceso directo a la memoria es totalmente crítico.

El proceso de compilación de *C#* es simple en comparación con el de *C* y *C++*, y es más flexible que en *Java*, no hay archivos de encabezado independientes, ni se requiere que los métodos y los tipos se declaren en un orden determinado.

Un archivo de código fuente de *C#* puede definir cualquier número de clases, structs, interfaces y eventos.

1.7 Introducción al lenguaje *ASP.NET*

ASP.NET rompe totalmente con el pensamiento de script que se tenía anteriormente, el cambio en la arquitectura es radical, de hecho, lo único que mantiene de *ASP* es el nombre, el propietario y la evolución de *Visual Basic* a *Visual Basic .NET*, lo demás es totalmente nuevo y renovado.

Dado que la web no se lee secuencialmente sino que se compila, lo primero que llama la atención es el enorme incremento en la velocidad de respuesta del servidor, además, al compilarse, el incremento en seguridad y robustez de aplicación es muy grande.

ASP.NET introduce el concepto de *code-behind*, por el que una misma página se puede componer de dos ficheros: el de la interface de usuario y el de código asociado a un lenguaje de programación más completo y robusto en cuanto a su sintaxis de programación que puede ser *Visual Basic .NET*, o *Visual C# .NET*.

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 un recurso remoto como puede ser una base de datos tiene almacenado, por tanto, cualquier cambio drástico de especificaciones minimiza los cambios en la aplicación y maximiza la facilidad de mantenimiento.

Así también, *ASP.NET* es tan útil tanto para desarrollar webs sencillas como para grandes aplicaciones, no se debe olvidar de la orientación a objetos y la naturaleza compilada, permitiendo que se haga uso de herramientas externas en la creación de webs, las más importantes de la familia de Microsoft Visual Studio, que faciliten mucho la tarea de programación, estas herramientas permiten hacer webs sencillas y

de bajas prestaciones en un tiempo record, así como llevar el mantenimiento de grandes aplicaciones de forma muy sencilla.

Resumiendo, se tiene mayor velocidad, mayor robustez, mayor seguridad, mayor facilidad de mantenimiento y herramientas de trabajo, pero las ventajas no paran aquí.

A continuación se enumeran algunas otras que no tienen *ASP*, *PHP* o *JSP*:

- Caché: se puede almacenar en la caché del servidor tanto páginas enteras, como controles personalizados o simples variables, en páginas críticas con mucha carga de datos remotos no es muy útil almacenar información en caché, reduciendo enormemente el consumo de recursos.

- Carpetas especializadas, como por ejemplo *App_code* que compila automáticamente las clases que se alojan en ella, *App_themes* en la que se alojan ficheros que marcan los temas de estilos de la web, entre otras carpetas especializadas más.

- Los archivos de configuración *Web.config* y *Machine.config* permiten realizar la operación de configuración en ficheros que hasta ahora había que realizarse en el servidor.

- La adaptación automática del código devuelto a los dispositivos que le acceden, una misma página puede ser utilizada para *Internet Explorer*, para el *Pocket Internet Explorer* desde una *PDA* o para un navegador de un móvil cualquiera.

- La eliminación total de los frames *HTML* con la introducción de las *Master Pages*.

- La extraordinaria compatibilidad con *XML* y los servicios web.

- La multitud de controles web que se encuentran en el cuadro de herramientas del entorno de desarrollo, permitiendo mucha funcionalidad con poco código.

- Se puede utilizar hasta cuarenta lenguajes de programación distintos para el desarrollo en *ASP.NET*, aunque en el 95% de las aplicaciones se usa *C# .NET* y *VB.NET*.

1.7.1 Componentes de una aplicación *ASP.NET*

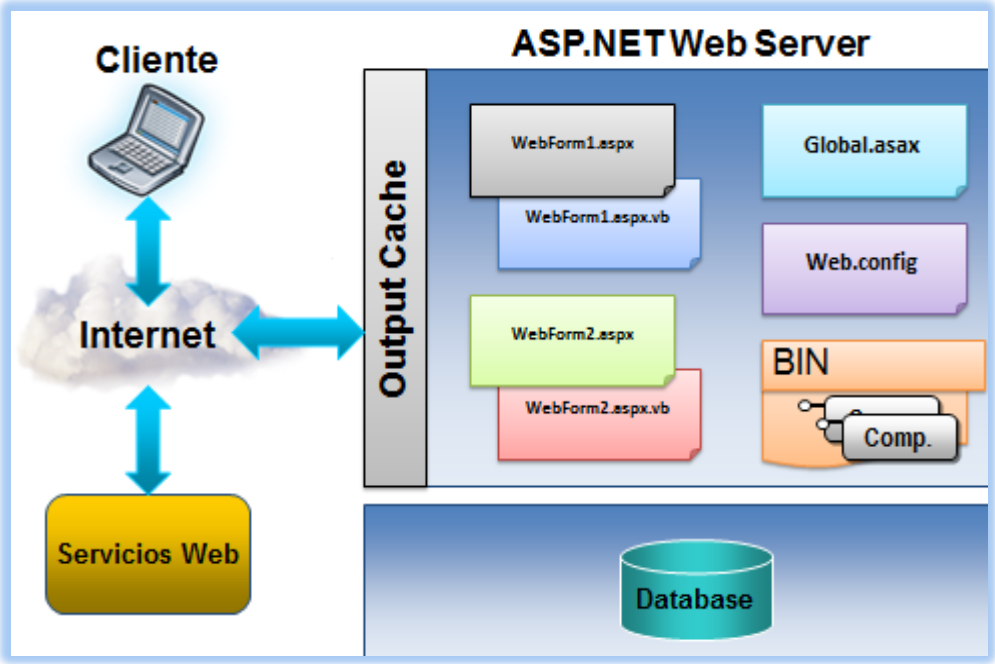

Figura 9. Componentes de una aplicación *ASP.NET*.

1.7.2 Modelo de eventos de un *WebForm*

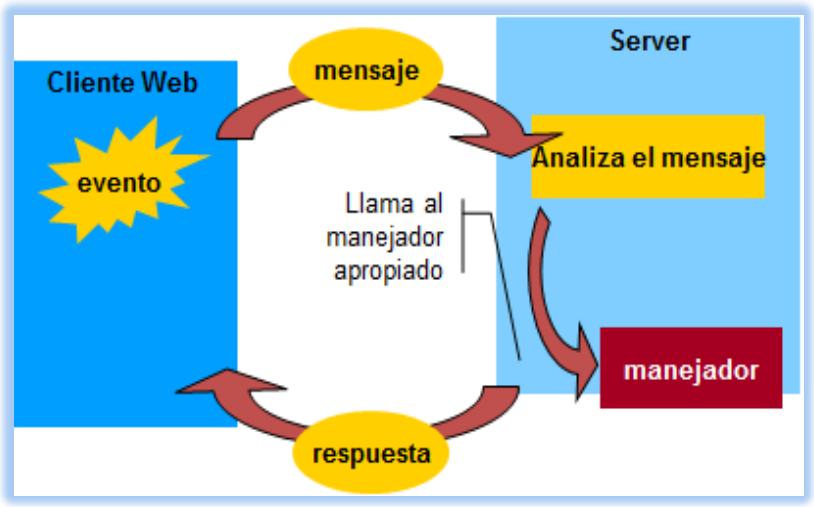

Figura 10. Modelo de eventos de un *WebForm*.

1.7.3 Compilación dinámica en *ASP.NET*

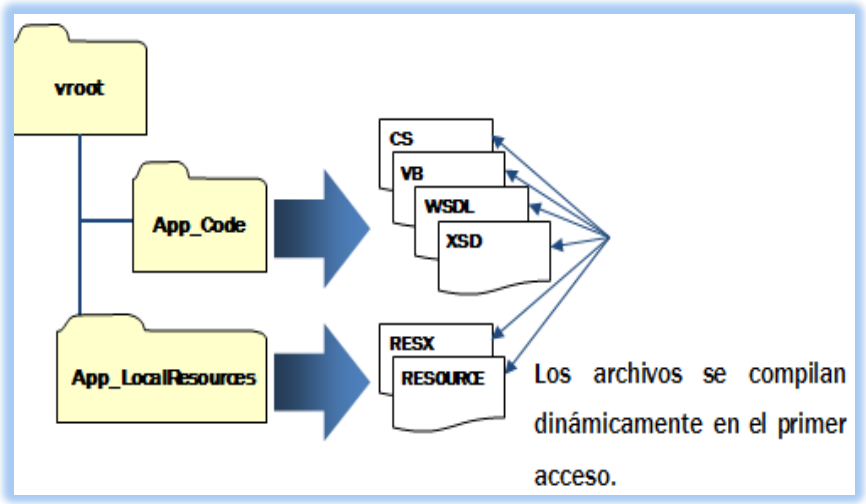

Figura 11. Compilación dinámica en *ASP.NET*.

1.7.4 Administración de estados en *ASP.NET*

Del lado del servidor	Del lado del cliente
Application state Información disponible para todos los usuarios de la aplicación Web.	**Cookies** Archivos de texto que guardan información de estado en la PC cliente.
Session state Información disponible únicamente para un usuario de una sesión específica.	**ViewState** Mantiene valores entre múltiples solicitudes a la misma página.
Database En algunos casos se utiliza una Base de Datos para guardar la información de estado.	**Query strings** Información anexada al final de la URL.

Figura 12. Administración de estados en *ASP.NET*.

1.7.5 ViewState

- Mantiene el estado de los controles, entre *postback* de una página.
- El *ViewState* se implementa mediante un campo oculto en el *html* generado y viaja en cada *post*.

1.8 Introducción al lenguaje *CSS*

CSS (**C**ascading **S**tyle **S**heets)

Hojas de estilo en cascada

1.8.1 ¿Qué es *CSS*?

CSS es un lenguaje creado para controlar el aspecto o presentación de los documentos electrónicos definidos con *HTML* y *XHTML*. *CSS* es la mejor forma de separar los contenidos y su presentación y es imprescindible para crear páginas web complejas.

Es un mecanismo simple que describe cómo se va a mostrar un documento en la pantalla, la ubicación, tamaño, colores, fondos, sangrías, espaciados, formatos de texto, en otras muchísimas funcionalidades que se puede aplicar a cada una de las herramientas o controles web que se estén utilizando.

Esta forma de descripción de estilos ofrece a los desarrolladores el control total sobre el estilo y formato de los documentos.

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

Al crear una página web, se utiliza en primer lugar el lenguaje *HTML/XHTML* para marcar los contenidos, es decir, para designar la función de cada elemento dentro de la página: párrafo, titular, texto destacado, tabla, lista de elementos, etc.

1.8.2 ¿Para qué sirve?

CSS se utiliza para dar estilo a documentos *HTML* o *XHTML*, separando el contenido de la presentación.

CSS permite a los desarrolladores web controlar el estilo y el formato de múltiples páginas web al mismo tiempo, cualquier cambio en el estilo marcado para un elemento en la *CSS* afectará a todas las páginas vinculadas a esa *CSS* en las que aparezca ese elemento.

1.8.3 ¿Cómo Funciona *CSS*?

CSS funciona a base de reglas, es decir, declaraciones sobre el estilo de uno o más elementos, las hojas de estilo están compuestas por una o más de esas reglas aplicadas a un documento *HTML* o *XHTML*.

La regla tiene dos partes: *un selector* y la *declaración*, a su vez la declaración está compuesta por una *propiedad* y el *valor* que se le asigne.

input

```
{
        color: #E2F0FF;
}
```

input es el selector

{ color: #E2F0FF; } es la declaración

El *selector* funciona como enlace entre el documento y el estilo, especificando los elementos que se van a ver afectados por esa declaración.

La *declaración* es la parte de la regla que establece cuál será el efecto.

En el ejemplo anterior, el selector de etiqueta (*input*) indica que todos los elementos *input* se verán afectados por la declaración donde se establece que la *propiedad color* va a tener el *valor* de *#E2F0FF*.

Todos los elementos *input* del documento o documentos que estén vinculados a esa hoja de estilos heredarán la configuración de este estilo definido.

1.8.4 Principales ventajas de *CSS*

- **Control del diseño:**

CSS es muy útil para separar el contenido del diseño, siendo esto muy esencial cuando se quiere cambiar un aspecto del diseño de un sitio web, ya que sin hojas de estilo se tendría que cambiar página a página dicho aspecto.

Y sin embargo, cuando se han definido hojas de estilo, se puede cambiar dicho aspecto modificando únicamente la hoja de estilos, con lo que se consigue además un gran ahorro de tiempo, una mayor uniformidad en el diseño.

- **Redefinición de etiquetas:**

Esto quiere decir que se puede redefinir el aspecto de visualización de una etiqueta, con la hoja de estilos se puede lograr por ejemplo con una sola línea de código que todos los encabezamientos <*span* /> tengan un color o un tamaño determinado, o sólo un *span* específico.

Pero si no se usa hojas de estilo, se tendría que definir todo el segmento de código declarativo de estilos cada vez que se use <*span* />.

- **Uso de etiquetas para su misión:**

Gracias a poder maquetar mediante el uso de hojas de estilo, se puede usar cada etiqueta para lo que realmente es útil.

Por ejemplo <*font-family*>, <*font-size*>, para definiciones de estilos de texto.

Por ejemplo <*table*> se usará para lo que realmente fue creada, para presentar datos tabulados.

Y se puede dar un uso más adecuado a <*img*> y librarse del socorrido *gif* de un pixel para controlar el espaciado entre los elementos en la visualización.

- **Maquetación y accesibilidad:**

Se pueden colocar elementos con mayor precisión, es común cuando no se usan hojas de estilo recurrir a tablas para la maquetación, pese a que esto implica un problema de accesibilidad (los contenidos para una persona con discapacidades visuales en realidad están desordenados).

Además de este problema, es complicado lograr el punto exacto donde se quiere colocar algo y se genera una cantidad de código excesiva, que debía ser reutilizado en todas las páginas, mientras que con las hojas de estilo implica menos problemas y menos código.

Del mismo modo, aquel usuario con un navegador que no soporta *CSS* seguirá viendo el documento en un formato más lineal independiente de su navegador.

- **Respeto a los estándares:**

El sitio web será más respetuoso con los estándares de desarrollo web con el uso de *CSS*, lo que implica que será más funcional en los diferentes navegadores, el código será más sencillo y se tendrá documentación más precisa para trabajar los documentos.

- **Tamaño:**

Se logrará reducir el tamaño de los ficheros, disminuyendo el ancho de banda que se consume (disminuyendo costos por tanto) y acercando los recursos de la web a los usuarios que agradecerán tener de manera más rápida lo que realmente interesa, el contenido.

- **Visibilidad:**

Los robots de los motores de búsqueda que usan los navegadores en modo texto como *Lynx*, definiendo el diseño en la hoja de estilo se le estará ahorrando muchísimo trabajo al buscador ya que podrá indexar los documentos con mayor rapidez, lo que es un criterio favorable en la valoración del mismo.

1.8.5 Desventaja de CSS

- Quizá una desventaja sea que la hoja de estilos al encontrase en un archivo externo, al visitar la página por primera vez, en el caso de conexiones de internet sumamente lentas, se comenzará a cargar el sitio sin su estructura gráfica, lo que lo hará más lineal y poco atractivo visualmente, aunque en unos pocos segundos adquirirá su verdadero aspecto.

1.9 Introducción a lenguaje *Javascript* + compatibilidad con ECMAScript 262 5ta. Edición.

1.9.1 ¿Qué es *Javascript*?

- *Javascript* es un lenguaje de *scripting* basado en objetos, utilizado para acceder a los mismos a nivel de aplicación cliente.
- *Javascript* es un lenguaje de programación que se utiliza principalmente para crear páginas web dinámicas, validación de formularios de datos, acceso y validación de cualquiera de los controles web, pero todo a nivel de aplicación cliente.

1.9.2 Parámetros a tener en cuenta a la hora de utilizar *Javascript*.

- *Javascript* es un lenguaje de programación interpretado, por lo que no es necesario compilar los programas para ejecutarlos.
- *Javascript* puede incluirse en cualquier documento *HTML* o *XHTML*, manteniendo parámetros de compatibilidad en el navegador web del cliente, sin importar con qué tipo de documento este asociado ya sea con sintaxis de programación *PHP*, *ASP* o *JSP*.
- Incluir código directamente en una estructura *HTML* es una práctica invasiva y no recomendada.
- El método correcto que define la *W3C* es incluir *Javascript* como un archivo externo, tanto por cuestiones de accesibilidad y velocidad en la navegación.

1.9.3 Javascript y su relación con *DOM* (*Document Object Model*)

DOM permite a los programadores web acceder y manipular las páginas *XHTML* como si fueran documentos *XML*, de hecho, *DOM* se diseñó originalmente para manipular de forma sencilla los documentos *XML*.

- **Árbol de nodos**

Sin embargo, para poder utilizar las utilidades de *DOM*, es necesario "*transformar*" la página original.

Una página *HTML* o *XHTML* normal no es más que una sucesión de caracteres, por lo que es un formato muy difícil de manipular, por ello, los navegadores web transforman automáticamente todas las páginas web en una estructura más eficiente de manipular.

DOM transforma todos los documentos *XHTML* en un conjunto de elementos llamados nodos, que están interconectados y que representan los contenidos de las páginas web y las relaciones entre ellos, por su aspecto, la unión de todos los nodos se llama "*árbol de nodos*".

La siguiente página *XHTML* sencilla ilustra lo mencionado anteriormente:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta content="text/html; charset=utf-8" http-equiv="Content-Type" />
<title>Página sencilla</title>
</head>
<body>
<p>Esta página es <strong>muy sencilla</strong></p>
</body>
</html>
```

Figura 13. Página *XHTML* sencilla.

Se trasforma en el siguiente árbol de nodos *DOM*:

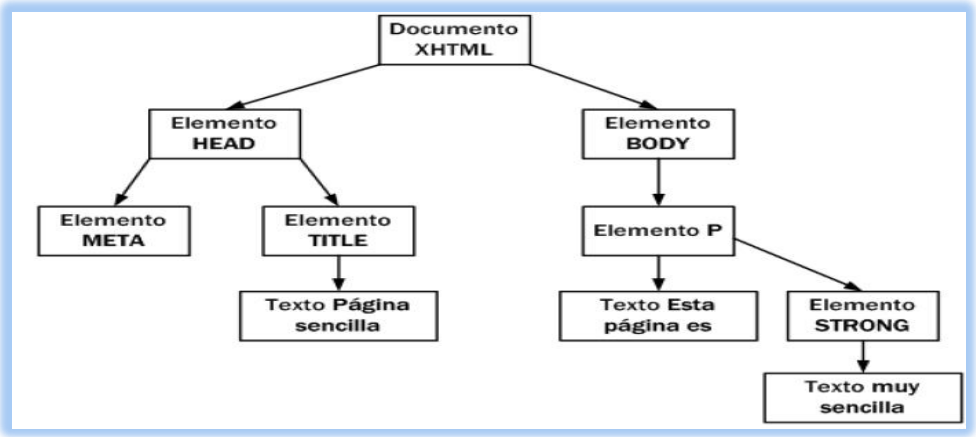

Figura 14. Transformación de una página *XHTML* sencilla en un árbol de nodos *DOM*.

En el esquema anterior, cada rectángulo representa un nodo *DOM* y las flechas indican las relaciones entre nodos, dentro de cada nodo, se ha incluido su tipo y su contenido.

La raíz del árbol de nodos de cualquier página *XHTML* siempre es la misma: un nodo de tipo especial denominado "*Document*".

A partir de ese nodo raíz, cada etiqueta *XHTML* se transforma en un nodo de tipo "*Element*", la conversión de etiquetas en nodos se realiza de forma jerárquica.

De esta forma, del nodo raíz solamente pueden derivar los nodos HEAD y BODY.

A partir de esta derivación inicial, cada etiqueta *XHTML* se transforma en un nodo que se deriva del nodo correspondiente a su "*etiqueta padre*".

Como se puede suponer, las páginas *XHTML* habituales producen árboles con miles de nodos, aun así, el proceso de transformación es rápido y automático.

1.9.4 Tipos de nodos *DOM*

La especificación completa de *DOM* define 12 tipos de nodos, aunque las páginas *XHTML* habituales se pueden manipular manejando solamente cuatro o cinco tipos de nodos:

- *Document*, nodo raíz del que derivan todos los demás nodos del árbol.
- *Element*, representa cada una de las etiquetas *XHTML*, se trata del único nodo que puede contener atributos y el único del que pueden derivar otros nodos.
- *Attr*, se define un nodo de este tipo para representar cada uno de los atributos de las etiquetas *XHTML*, es decir, uno por cada par *atributo = valor*.
- *Text*, nodo que contiene el texto encerrado por una etiqueta *XHTML*.
- *Comment*, representa los comentarios incluidos en la página *XHTML*.

Los otros tipos de nodos existentes que no se van a considerar son *DocumentType*, *CDataSection*, *DocumentFragment*, *Entity*, *EntityReference*, *ProcessingInstruction* y *Notation*.

1.9.5 Modelo de eventos en javascript

Los eventos hacen posible que los usuarios transmitan información a los programas.

De esta forma, cuando se produce cualquier evento, *javascript* ejecuta su función asociada, este tipo de funciones se denominan "*event handlers*" en inglés y suelen traducirse por "*manejadores de eventos*".

1.9.6 Modelo básico de eventos

Este modelo simple de eventos se introdujo para la versión 4 del estándar *HTML* y se considera parte del nivel más básico de *DOM*, aunque sus características son limitadas, es el único modelo que es compatible en todos los navegadores y por tanto, el único que permite crear aplicaciones que funcionan de la misma manera en todos los navegadores.

1.9.7 Modelo de eventos estándar

Las versiones más avanzadas del estándar *DOM* (*DOM nivel 2*) definen un modelo de eventos completamente nuevo y mucho más poderoso que el original.

Todos los navegadores modernos lo incluyen, salvo *Internet Explorer 7* e inferiores.

1.9.8 Modelo de eventos de *Internet Explorer 7* e inferiores

Internet Explorer 7 e inferiores utiliza su propio modelo de eventos, que es similar pero incompatible con el modelo estándar, se utilizó por primera vez en *Internet Explorer 4* y Microsoft decidió seguir utilizándolo en el resto de versiones, a pesar de que la empresa había participado en la creación del estándar *DOM* que define el modelo de eventos estándar.

1.9.9 Jerarquía del modelo básico de eventos

El nombre de cada evento se construye mediante el prefijo ***on***, seguido del nombre en inglés de la acción asociada al evento.

A continuación se resume los eventos más importantes definidos por *javascript*:

Eventos *DOM* Nivel 1	
on**blur**	on**load**
on**change**	on**mousedown**
on**click**	on**mousemove**
on**dblclick**	on**mouseout**
on**focus**	on**mouseover**
on**keydown**	on**reset**
on**keypress**	on**resize**
on**keyup**	on**select**
on**unload**	on**submit**

1.9.10 Ventajas de la utilización de *javascript*

- *Páginas web dinámicas:* Mediante la utilización de *javascript* se consiguen interfaces de usuario elegantes, flexibles y dinámicas.
- *Velocidad de transferencia:* El ejecutar la lógica de validación de formularios, validación en la manipulación de objetos, validación de datos ingresados por el usuario; al ejecutar toda esta lógica a nivel de aplicación cliente se consigue la optimización eminente de recursos en la compilación de la aplicación, y por ende se mejora sustancialmente la velocidad de transferencia de información y procesamiento a nivel de servidor.

- *Validaciones dinámicas:* Con la utilización de un "*scripting language*" como lo es *javascript* se consigue la validación de información ingresada por el usuario en tiempo real, estableciéndose así una comunicación asíncrona con el servidor.
- Evitar carga de procesamiento de información en el servidor web.

1.9.11 Posibles desventajas al utilizar *javascript*

- Es necesario enviar toda la lógica al browser, esto en ocasiones puede ser un volumen substancial (*100Kb o 200Kb*) y presentar un retardo adicional para aquellos usuarios con conexiones de internet sumamente lentas (módem-*28 Kbps*).
- Debido a que la lógica es enviada al "*browser*", es posible que un usuario con suficientes conocimientos altere dicha lógica, esto puede ser un problema crítico especialmente al validar información.
- El lenguaje es muy intolerable con errores, basta un ' (apóstrofe) o) (paréntesis) adicional en una librería o declaración, para que 1000 o 2000 líneas de código cesen de funcionar.

1.10 .NET Framework

1.10.1 Evolución funcional y .NET Framework

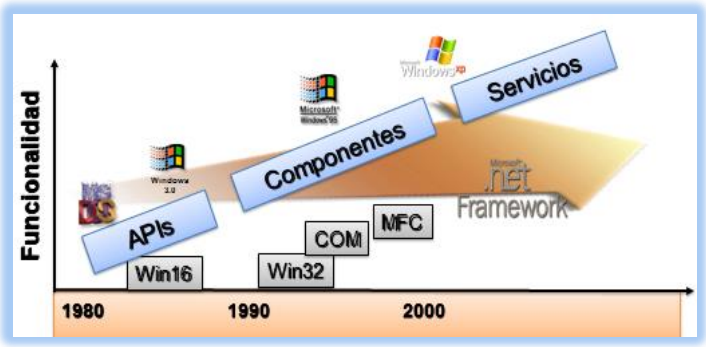

Figura 15. Evolución funcional y .NET Framework.

1.10.2 ¿Qué NO es .NET?

- .NET no es un sistema operativo.
- .NET no es un entorno de desarrollo.
- .NET no es un servidor de aplicaciones.
- .NET no es un lenguaje de programación.
- .NET no es un producto empaquetado que se pueda comprar como tal.

1.10.3 ¿Qué es .NET?

- Plataforma de desarrollo compuesta de:
  - Entorno de ejecución (*Runtime*).
  - Bibliotecas de funcionalidad (*Class Library*).
  - Lenguajes de programación.
  - Compiladores.
  - Herramientas de desarrollo (*IDE & Tools*).
  - Guías de arquitectura.
- La evolución de la plataforma *COM*.

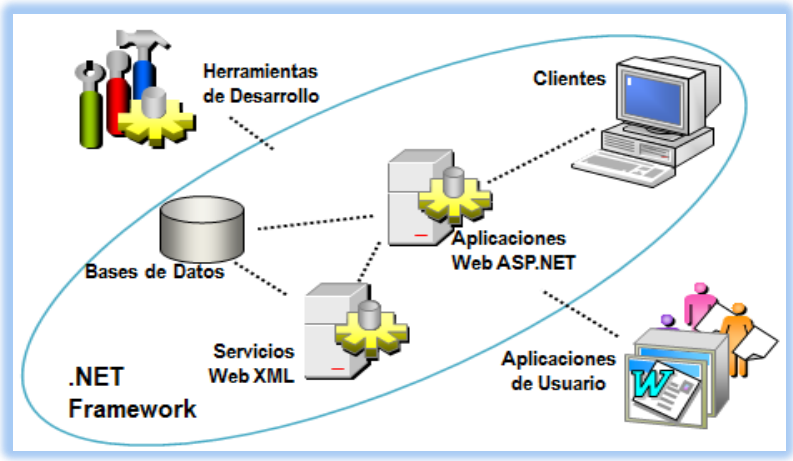

Figura 16. ¿Qué es .NET Framework?

1.10.4 Características de .NET

- Plataforma de ejecución intermedia.
- 100% orientada a objetos.
- Multilenguaje.
- Plataforma empresarial de misión crítica.
- Modelo de programación único para todo tipo de aplicaciones y dispositivos de hardware.
- Se integra fácilmente con aplicaciones existentes desarrolladas en plataformas Microsoft.
- Se integra fácilmente con aplicaciones desarrolladas en otras plataformas.

1.10.5 Extensibilidad de .NET

- .NET Framework no es una "*caja negra*".
- Sus clases pueden ser extendidas a través del mecanismo de herencia.
  - A diferencia de *COM*, usamos y extendemos las clases en sí mismas, no en un "*wrapper*".
- Herencia entre distintos lenguajes.

1.10.6 Ventajas de .NET

- Unifica los modelos de programación.
- Simplifica aún más el desarrollo.
- Provee un entorno de ejecución robusto y seguro.
- Es independiente del lenguaje de programación.
- Interoperabilidad con código existente.
- Simplifica la instalación y administración de las aplicaciones.
- Es extensible.

1.10.7 Interoperabilidad de .NET

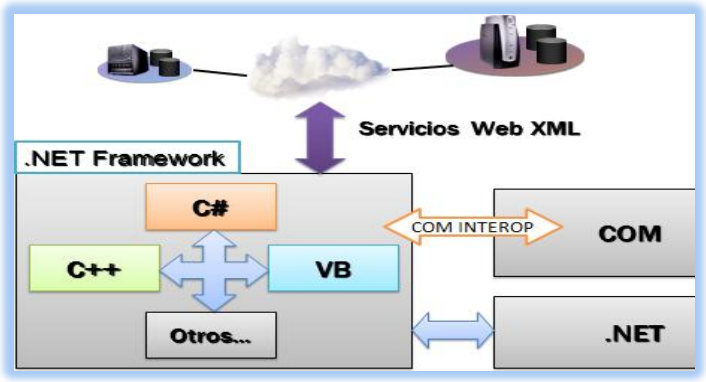

Figura 17. Interoperabilidad de .NET Framework.

1.10.8 .NET como plataforma de ejecución intermedia

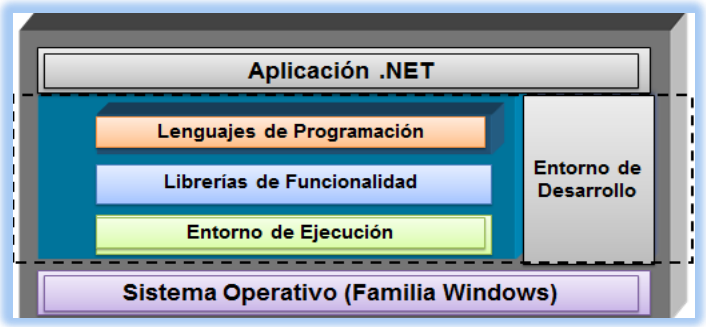

Figura 18. .NET como plataforma de ejecución intermedia.

1.10.9 .NET como evolución de COM

- *Entorno de ejecución (Runtime)*
  - ✓ COM: Windows.
  - ✓ .NET: Common Language Runtime.
- *Librerías de funcionalidad*
  - ✓ COM: Algunas (ADO, FSO, etc.).
  - ✓ .NET: Muy extensa (.NET Framework Class Library).
- *Lenguajes de programación*
  - ✓ COM: VB, C++, VFP, ASP, J++.
  - ✓ .NET: Common Language Specification.
- *Entorno de desarrollo integrado (IDE)*
  - ✓ COM: Uno para cada lenguaje.
  - ✓ .NET: Uno independiente del lenguaje (VS.NET).

1.10.10 ¿Dónde instalar .NET Framework?

* *Sólo si la aplicación es distribuída.*

	Cliente	Servidor
Aplicación de Escritorio	✓	✓*
Aplicación Web		✓
Aplicación de Consola	✓	✓*
Aplicación Móvil	.NET Compact Framework	

Figura 19. ¿Dónde instalar .NET Framework?

1.10.11 Arquitectura de .NET Framework

Los programas de *C#* se ejecutan en .NET Framework, un componente que forma parte integral de Windows y que incluye un sistema de ejecución virtual denominado *Common Language Runtime* (*CLR*) y un conjunto unificado de bibliotecas de clases.

CLR es la implementación comercial de Microsoft de *CLI* (*Common Language Infrastructure*), un estándar internacional que constituye la base para crear entornos de ejecución y desarrollo en los que los lenguajes y las bibliotecas trabajan juntos sin ningún problema.

El código fuente escrito en *C#* se compila en un *lenguaje intermedio* (*IL*) conforme con la especificación *CLI*, el código de lenguaje intermedio y recursos tales como mapas de bits y cadenas se almacenan en disco en un archivo ejecutable denominado ensamblado, cuya extensión es *.exe* o *.dll* generalmente, un ensamblado contiene un manifiesto que proporciona información sobre los tipos, la versión, la referencia cultural y los requisitos de seguridad del ensamblado.

Cuando se ejecuta un programa de *C#*, el ensamblado se carga en *CLR*, con lo que se pueden realizar diversas acciones en función de la información del manifiesto.

A continuación, si se cumplen los requisitos de seguridad, *CLR* realiza una compilación *Just In Time* (*JIT*) para convertir el código de *lenguaje intermedio* en *instrucciones máquina nativas*. *CLR* también proporciona otros servicios relacionados con la recolección automática de elementos no utilizados, el control de excepciones y la administración de recursos, el código ejecutado por *CLR* se denomina algunas veces "*código administrado*", en contraposición al "*código no administrado*" que se compila en *lenguaje máquina nativo* destinado a un sistema específico.

En el diagrama siguiente se muestran las relaciones en tiempo de compilación y en tiempo de ejecución de los archivos de código fuente de lenguaje *C#*, las bibliotecas de clases de .NET Framework, los ensamblados y *CLR*.

.

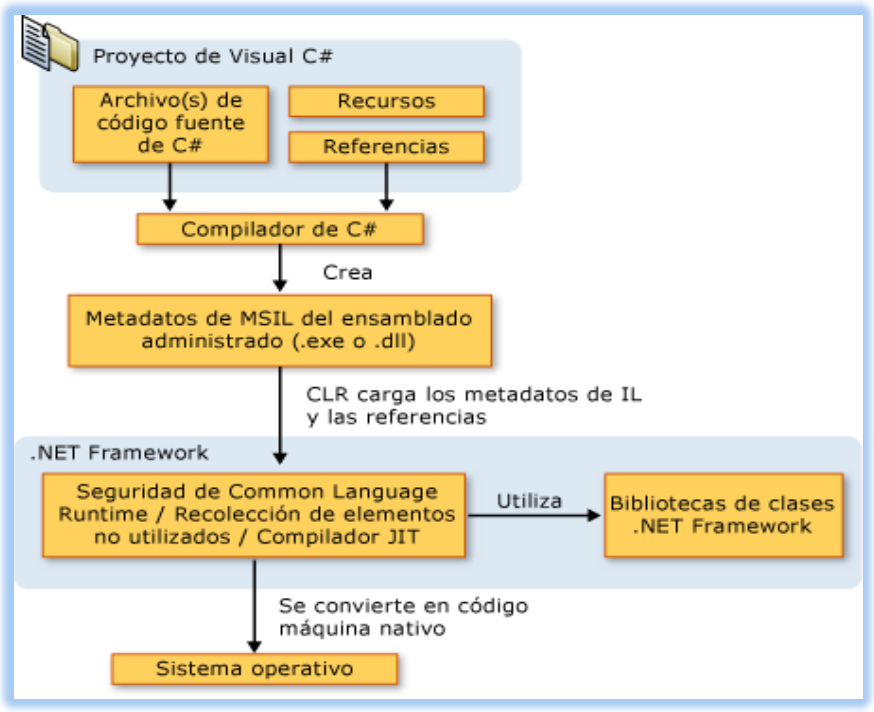

Figura 20. Relación en tiempo de compilación y ejecución de .NET Framework.

La *interoperabilidad* del lenguaje es una característica clave de .NET Framework.

Como el código de *lenguaje intermedio* generado por el compilador de *C#* cumple la especificación de tipos común (*CTS*), este código generado en *C#* puede interactuar con el código generado en las versiones .NET de *Visual Basic*, *Visual C++* o cualquiera de los más de 20 lenguajes conformes a *CTS*.

Un único ensamblado puede contener varios módulos escritos en diferentes lenguajes .NET, y los tipos admiten referencias entre sí como si estuvieran escritos en el mismo lenguaje.

Además de los servicios en tiempo de ejecución, .NET Framework también incluye una amplia biblioteca de más de 4.000 clases organizadas en espacios de nombres que proporcionan una gran variedad de funciones útiles para la entrada y salida de archivos, la manipulación de cadenas, el análisis *XML*, los controles de los formularios *Windows Forms* y muchas tareas más.

La aplicación de *C#* típica utiliza continuamente la biblioteca de clases de .NET Framework para el tratamiento de las tareas comunes de "*infraestructura*".

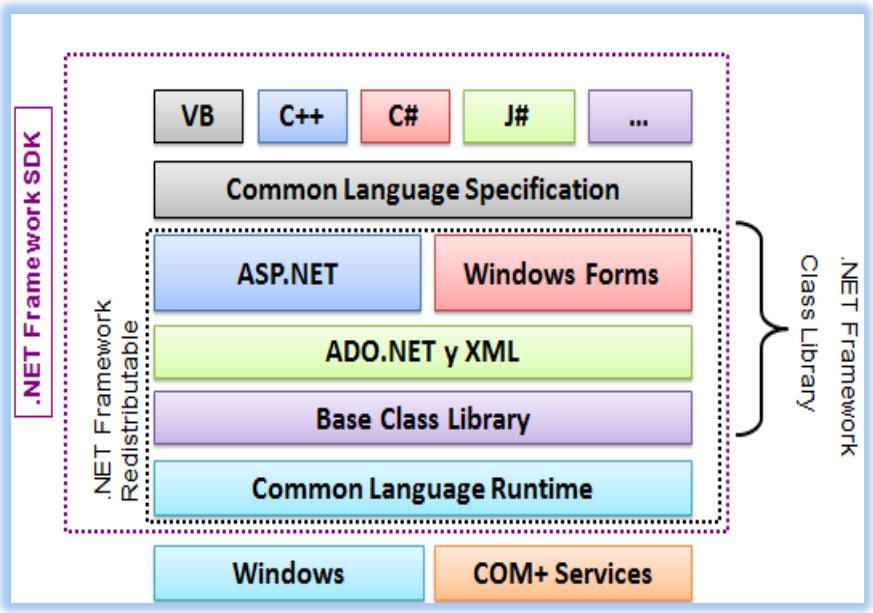

Figura 21. Arquitectura de .NET Framework.

1.10.12 *CLR* (*Common Language Runtime*)

- *CLR* es el motor de ejecución (*runtime*) de .NET Framework.
- Ofrece servicios automáticos tales como:
  - Administración de memoria.
  - Seguridad de código administrado:
    - Conversión de tipos.
    - Inicialización de variables.
    - Indexación de arreglos fuera de sus límites.
  - Versionamiento.

1.10.13 Características del *CLR*

- Compilación *Just-In-Time* (*JIT*).
- Gestión automática de memoria (*Garbage Collector*).
- Gestión de errores consistente (*excepciones*).
- Ejecución basada en componentes (*Assemblies*).
- Gestión de seguridad.
- Multithreading.

1.10.14 Proceso de compilación del *CLR*

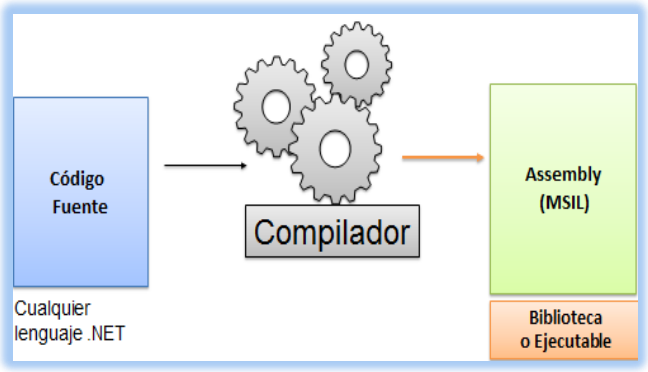

Figura 22. Proceso de compilación del *CLR*.

1.10.15 Modelo de ejecución del *CLR*

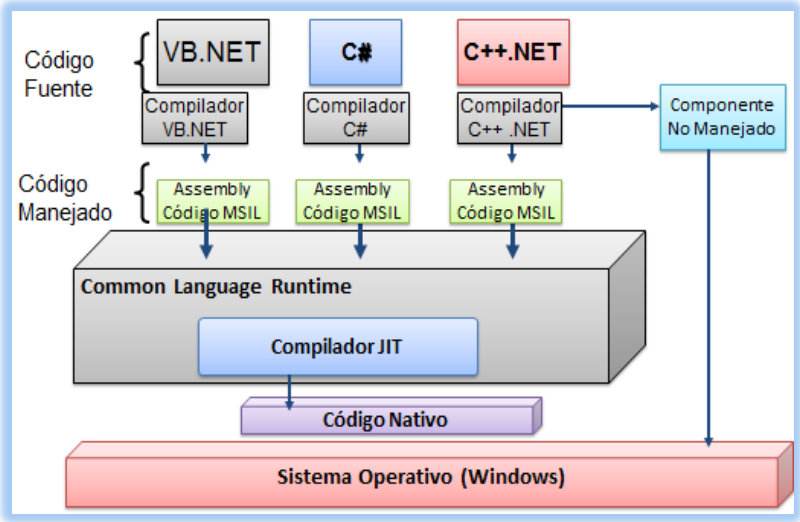

Figura 23. Modelo de ejecución del *CLR*.

1.10.16 ¿Qué es el Metadata?

- *Describe a un assembly*
  - Identifica: nombre, versión, referencia cultural, llaves públicas.
  - Que tipos son exportados.
  - A que otros *assemblys* hace referencia.
  - Permisos que se necesitan para la ejecución.
- *Descripción de tipos*
  - Nombre, visibilidad, clases base, interfaces que implementa.
  - Miembros (métodos, campos, propiedades, eventos, tipos anidados).
- *Sentencias declarativas*
  - Atributos definidos por el usuario.
  - Atributos definidos por el compilador.
  - Atributos definidos por el framework.

1.10.17 ¿Qué son los Application Domains?

- Procesos virtuales dentro del *CLR*.
  - Se ejecutan dentro de un proceso del sistema operativo.
  - Un proceso del sistema operativo puede contener varios *AppDomains*.
  - Más eficiente que múltiples procesos del sistema operativo.
  - Más eficiente en el intercambio de contexto de ejecución.
  - Un *Assembly* y sus tipos son siempre cargados dentro de un *AppDomain*.
- Provee una frontera para: fallos, tipos, seguridad.

1.10.18 Arquitectura de *ADO.NET*

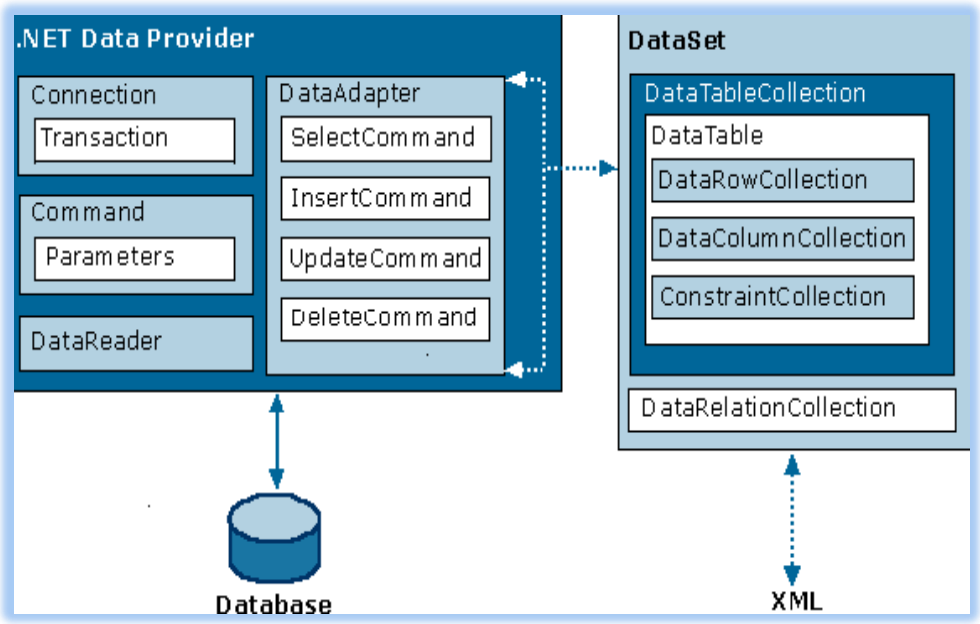

Figura 24. Arquitectura de *ADO.NET*.

1.11 Toolbox *ASP.NET* con el Framework 2.0

✔ Controles Estándar

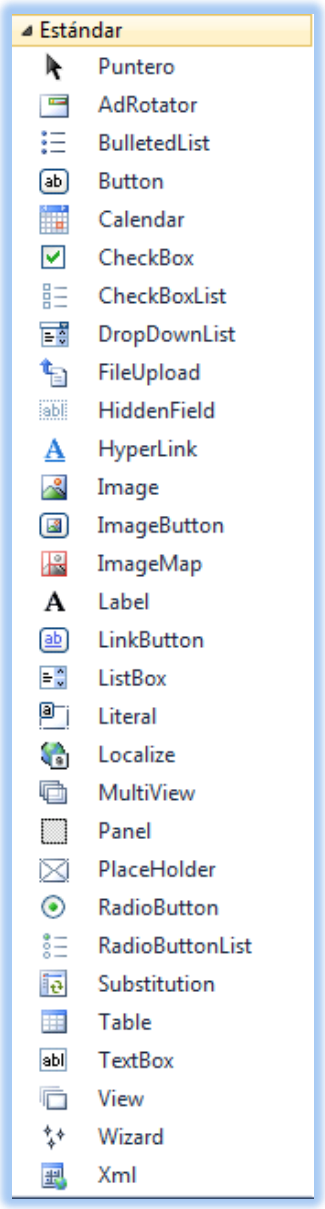

Figura 25. Controles Estándar: *ASP.NET* con el Framework 2.0.

Control	Descripción
AdRotator	Muestra un titular de anuncio en una página web.
BulletedList	Crea un control que genera una lista de elementos con formato de viñetas.
Button	Muestra un botón de comando en una página web.
Calendar	Muestra un calendario de un solo mes que permite al usuario seleccionar fechas y desplazarse al mes siguiente o al mes anterior.
Checkbox	Muestra una casilla que permite al usuario seleccionar una condición *true* o *false*.
CheckboxList	Crea un grupo de casillas de selección múltiple cuya creación podría realizarse de forma dinámica enlazando el control a un origen de datos.
DropDownList	Representa un control que permite al usuario seleccionar un único elemento de una lista desplegable.
FileUpload	Muestra un control de cuadro de texto y un botón de búsqueda que permite a los usuarios seleccionar un archivo para cargarlo al servidor.
HiddenField	Representa un campo oculto que se utiliza para almacenar un valor no mostrado.
HyperLink	Control que muestra un vínculo a otra página web.
Image	Muestra una imagen en una página web.
ImageButton	Control que muestra una imagen y responde a los clics del mouse en la imagen.
ImageMap	Crea un control que muestra una imagen en una página, cuando se hace clic en una región de la zona activa definida dentro del control *System.Web.UI.WebControls.ImageMap* el control genera una devolución de datos al servidor o navega a una dirección URL especificada.
Label	Representa un control de etiqueta que muestra texto en una página web.
LinkButton	Muestra un control de botón de tipo hipervínculo en una página web.
ListBox	Representa un control de cuadro de lista que permite la selección de uno o varios elementos.
Literal	Reserva una ubicación en la página web para mostrar texto estático.

Localize	Reserva una ubicación en la página web para mostrar en ella texto estático adaptado.
MultiView	Representa un control que actúa como contenedor de un grupo de controles *System.Web.UI.WebControls.View*.
Panel	Representa un control que actúa como contenedor de otros controles.
PlaceHolder	Almacena los controles de servidor agregados dinámicamente en la página web.
RadioButton	Representa un control de botón de opción.
RadioButtonList	Representa un control de lista que encapsula un grupo de controles de botón de opción.
Substitution	Para una página web dinámica cuyo resultado se almacena en caché, especifica una sección de dicha página donde no se aplica el almacenamiento en caché, en esa ubicación, el contenido dinámico se recupera y se sustituye en el control.
Table	Muestra una tabla en una página web.
TextBox	Muestra un control de cuadro de texto para la entrada de datos del usuario.
View	Representa un control que actúan como contenedor de un grupo de controles contenidos en un control *System.Web.UI.WebControls.MultiView*.
Wizard	Proporciona navegación y una interface de usuario (*UI*) para recopilar datos relacionados en varios pasos.
XML	Muestra un documento *XML* sin formato o que utiliza Extensible Stylesheet Language Transformations (*XSLT*).

✔ Controles de Datos

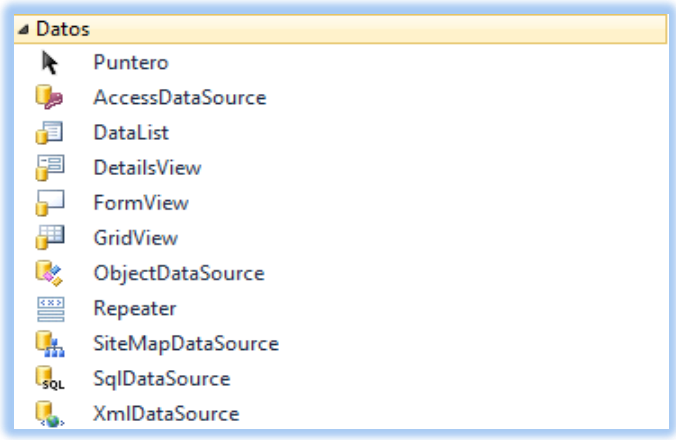

Figura 26. Controles de Datos: *ASP.NET* con el Framework 2.0.

Control	Descripción
AccessDataSource	Representa una base de datos de Microsoft Access para su uso con controles enlazados a datos.
DataList	Control de lista enlazada a datos que muestra los elementos mediante el uso de plantillas.
DetailsView	Muestra los valores de un único registro de un origen de datos en una tabla, donde cada fila de la tabla representa un campo del registro, el control *System.Web.UI.WebControls.DetailsView* pemite editar, eliminar e insertar registros.
FormView	Muestra los valores de un registro individual de un origen de datos utilizando plantillas definidas por el usuario, el control *System.Web.UI.WebControls.FormView* permite editar, eliminar e insertar registros.
GridView	Muestra los valores de un origen de datos en una tabla donde cada columna representa un campo y cada fila representa un registro. El control *System.Web.UI.WebControls.GridView* permite seleccionar, ordenar, eliminar, modificar e insertar registros.

ObjectDataSource	Representa un objeto comercial que proporciona datos a los controles enlazados a datos en las arquitecturas de aplicaciones web multi-nivel.
Repeater	Control de lista con enlace a datos que permite especificar un diseño personalizado aplicando la misma plantilla a cada uno de los elementos mostrados en la lista.
SiteMapDataSource	Proporciona un control de origen de datos que los controles de servidor web y otros controles pueden utilizar para enlazarse a los datos del mapa jerárquico de un sitio.
SqlDataSource	Representa una base de datos *SQL* para controles enlazados a datos.
XMLDataSource	Representa un origen de datos *XML* para controles enlazados a datos.

✔ Controles de Validación

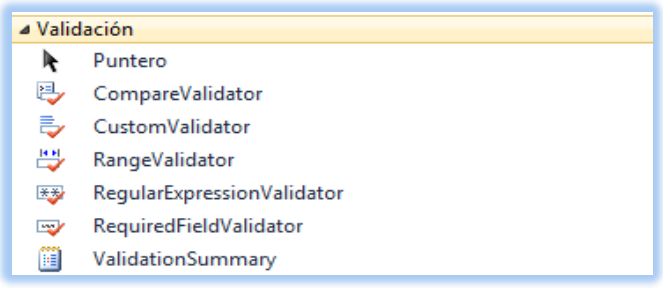

Figura 27. Controles de Validación: *ASP.NET* con el Framework 2.0.

Control	Descripción
CompareValidator	Compara el valor especificado por el usuario en un control de entrada con el valor especificado en otro control de entrada o con un valor constante.
CustomValidator	Realiza una validación definida por el usuario en un control de entrada.
RangeValidator	Comprueba si el valor de un control de entrada está comprendido en un intervalo especificado de valores.

RegularExpressionValidator	Comprueba si el valor de un control de entrada asociado coincide con el modelo especificado por una expresión regular.
RequiredFieldValidator	Convierte el control de entrada asociado en un campo obligatorio.
ValidationSummary	Muestra un resumen de todos los errores de validación en línea en una página web, en un cuadro de mensaje o en ambos.

✔ Controles de Navegación

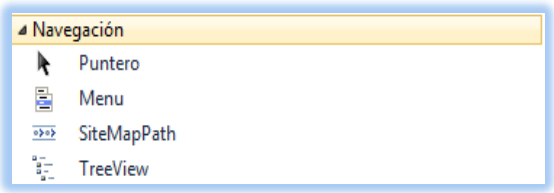

Figura 28. Controles de Navegación: *ASP.NET* con el Framework 2.0.

Control	Descripción
Menu	Muestra un menú en una página web *ASP.NET*.
SiteMapPath	Muestra un conjunto de hipervínculos de texto o imagen que permite a los usuarios explorar un sitio web más fácilmente utilizando una cantidad mínima de espacio de la página.
TreeView	Muestra datos jerárquicos, como una tabla de contenido, en una estructura de árbol.

✔ Controles de Inicio de Sesión

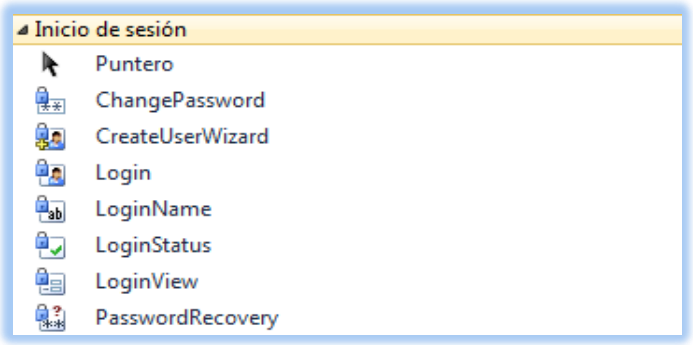

Figura 29. Controles de Inicio de Sesión: *ASP.NET* con el Framework 2.0.

Control	Descripción
ChangePassword	Proporciona una interface de usuario que permite que los usuarios modifiquen su contraseña en un sitio web.
CreateUserWizard	Proporciona una interface de usuario para crear nuevas cuentas de usuario del sitio web.
Login	Proporciona los elementos de la interface de usuario para iniciar sesión en un sitio web.
LoginName	Muestra el valor de la propiedad *System.Web.UI.Page.User.Identity.Name.*
LoginStatus	Detecta el estado de autenticación del usuario y alterna el estado de un vínculo para iniciar o cerrar sesión en un sitio web.
LoginView	Muestra la plantilla de contenido correspondiente a un usuario concreto a partir de su estado de autenticación y de su subscripción a funciones.
PasswordRecovery	Proporciona elementos de la interface de usuario que permiten a un usuario recuperar o restablecer una contraseña perdida y recibirla por correo electrónico.

✔ Elementos Web

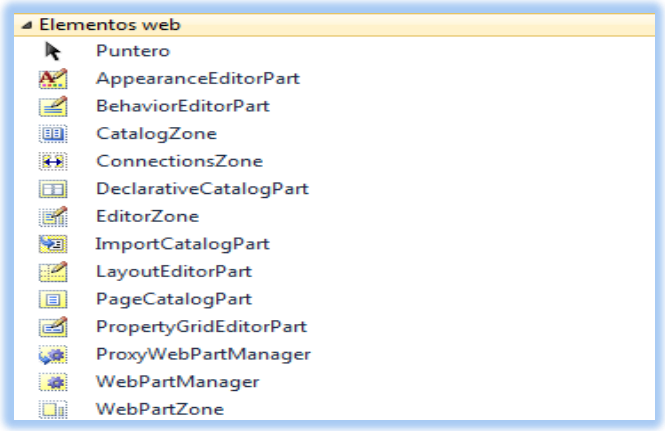

Figura 30. Elementos Web: *ASP.NET* con el Framework 2.0.

Control	Descripción
AppearanceEditorPart	Proporciona un control de editor que permite a los usuarios finales modificar varias propiedades de interface de usuario (*UI*) en un control *System.Web.UI.WebControls.WebParts.WebPart* asociado. *Esta clase no se puede heredar.*
BehaviorEditorPart	Proporciona un control de editor que permite a los usuarios finales modificar varias propiedades de interface de usuario (*UI*) en un control *System.Web.UI.WebControls.WebParts.WebPart* asociado. *Esta clase no se puede heredar.*
CatalogZone	Sirve como control primario del conjunto de controles de elementos web para alojar controles *System.Web.UI.WebControls.WebParts.CatalogPart* de una página web.
ConnectionsZone	Proporciona una interface de usuario que permite a los usuarios establecer conexiones entre controles *System.Web.UI.WebControls.WebParts.WebPart* y otros controles de servidor que residen en zonas *System.Web.UI.WebControls.WebParts.WebPartZoneBase.*

DeclarativeCatalogPart	Permite a los desarrolladores agregar un catálogo de elementos *System.Web.UI.WebControls.WebParts.WebPart* u otros controles de servidor a una página web en el formato declarativo de persistencia de página. *Esta clase no se puede heredar.*
EditorZone	Sirve como control primario en el conjunto de controles de elementos web para alojar los controles *System.Web.UI.WebControls.WebParts.EditorPart* de una página web.
ImportCatalogPart	Importa un archivo de descripción para un control *System.Web.UI.WebControls.WebParts.WebPart* (u otro control de servidor *ASP.NET* utilizando como control *System.Web.UI.WebControls.WebParts.WebPart*), de forma que los usuarios puedan agregar el control a una página web con una configuración definida. *Esta clase no se puede heredar.*
LayoutEditorPart	Proporciona un control de editor que permite a los usuarios finales editar varias propiedades de interface de usuario (*UI*) orientadas al diseño en un control *System.Web.UI.WebControls.WebParts.WebPart* asociado. *Esta clase no se puede heredar.*
PageCatalogPart	Proporciona un catálogo que mantiene referencias a todos los controles *System.Web.UI.WebControls.WebParts.WebPart* (y otros controles de servidor incluidos en zonas *System.Web.UI.WebControls.WebParts.WebPart.WebPartZoneBase*) que un usuario ha cerrado en una página de elementos web única, lo que permite a los usuario volver a agregar los controles cerrados a la página. *Esta clase no se puede heredar.*
PropertyGridEditorPart	Proporciona un control de editor que permite a los usuarios finales editar las propiedades personalizadas en un control *System.Web.UI.WebControls.WebParts.WebPart* o de servidor asociado. *Esta clase no se puede heredar.*

ProxyWebPartManager	Proporciona a los desarrolladores un mecanismo para declarar conexiones estáticas en una página de contenido cuando se ha declarado un control *System.Web.UI.WebControls.WebParts.WebPartManager* en la página principal asociada a las páginas de contenido.
WebPartManager	Sirve como clase central del conjunto de controles de elementos web y administra todos los controles de elementos web, la funcionalidad y los eventos que se producen en una página web.
WebPartZone	Sirve como control primario en el conjunto de controles de elementos web para alojar los controles *System.Web.UI.WebControls.WebParts.WebPart* de una página web.

- **Elementos HTML**

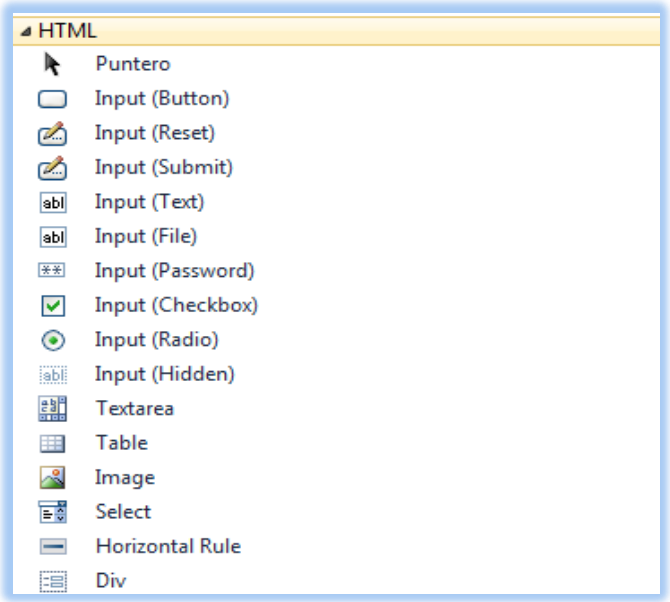

Figura 31. Elementos *HTML*: *ASP.NET* con el Framework 2.0.

Control	Descripción
Input (Button)	Permite el acceso mediante programación a los elementos *HTML* <input type="button">, <input type="submit"> y <input type="reset"> en el servidor.
Input (Reset)	Permite el acceso mediante programación al elemento *HTML* <input type="reset"> en el servidor.
Input (Submit)	Permite el acceso mediante programación al elemento *HTML* <input type="submit"> en el servidor.
Input (Text)	Permite el acceso mediante programación a los elementos *HTML* <input type="text"> y <input type="password"> en el servidor.
Input (File)	Permite el acceso mediante programación al elemento *HTML* <input type="file"> en el servidor.
Input (Password)	Permite el acceso mediante programación al elemento *HTML* <input type="password"> en el servidor.
Input (Checkbox)	Permite el acceso mediante programación al elemento *HTML* <input type="checkbox"> en el servidor.
Input (Radio)	Permite el acceso mediante programación al elemento *HTML* <input type="radio"> en el servidor.
Input (Hidden)	Permite el acceso mediante programación al elemento *HTML* <input type="hidden"> en el servidor.
Textarea	Permite el acceso mediante programación al elemento *HTML* <textarea> en el servidor.
Table	Permite el acceso mediante programación al elemento *HTML* <table> en el servidor.
Image	Permite el acceso mediante programación al elemento *HTML* <input type="image"> en el servidor.
Select	Permite el acceso mediante programación al elemento *HTML* <select> en el servidor.
Horizontal Rule	Permite el acceso mediante programación al elemento *HTML* <hr> en el servidor.
Div	Permite el acceso mediante programación al elemento .

1.12 Evolución del Toolbox *ASP.NET* con el Frameworks 3.0

En esta versión de la plataforma de .NET Framework se han mantenido todos los controles detallados anteriormente para la plataforma de .NET Framework 2.0, no obstante con algunas actualizaciones y mejoras en cada uno de los controles.

Para esta plataforma solamente se han agregado los siguientes ***nuevos controles*** *al Toolbox de ASP.NET:*

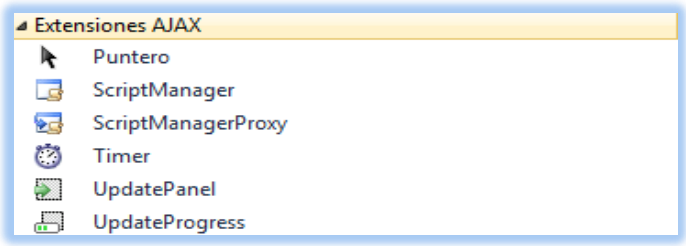

Figura 32. Extensiones *AJAX*: *ASP.NET* con el Framework 3.0.

Control	Descripción
ScriptManager	Administra las bibliotecas de scripts y los archivos de script *AJAX* de *ASP.NET*, la representación parcial de la página y la generación de la clase que hace las funciones de proxy cliente para los servicios web y de aplicación.
ScriptManagerProxy	Habilita los componentes anidados como páginas de contenido y controles de usuario que agreguen scripts y referencias de servicio a las páginas cuando ya se ha definido un control *System.Web.UI.ScriptManager* en un elemento primario.
Timer	Realiza las devoluciones de datos en páginas web asíncronas o síncronas en un intervalo definido.
UpdatePanel	Habilita las secciones de una página para que se representen parcialmente sin devoluciones de datos.
UpdateProgress	Proporciona comentarios visuales en el explorador cuando se actualiza el contenido de uno o más controles *System.Web.UI.UpdatePanel.*

1.13 Evolución del Toolbox *ASP.NET* con el Frameworks 3.5

En esta versión de la plataforma de .NET Framework se han mantenido todos los controles detallados anteriormente para la plataforma de .NET Framework 2.0 y .NET Framework 3.0, no obstante con algunas actualizaciones y mejoras en cada uno de los controles.

Para esta plataforma solamente se han agregado los siguientes ***nuevos controles*** *al Toolbox de ASP.NET:*

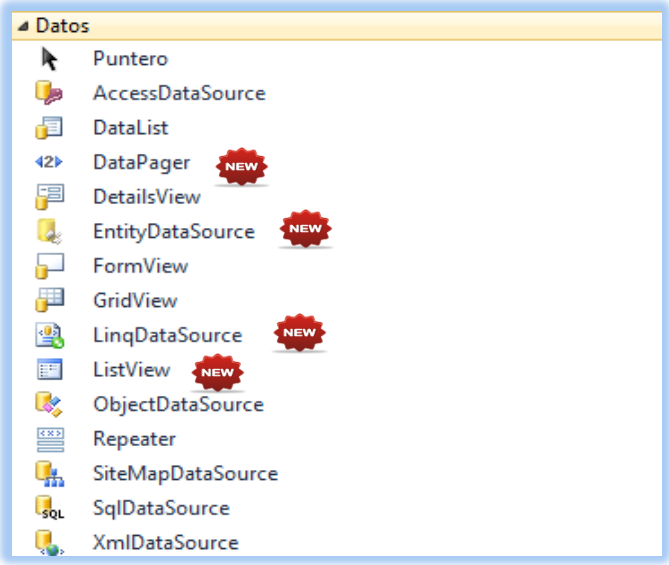

Figura 33. ***Nuevos*** Controles de Datos: *ASP.NET* con el Framework 3.5.

Control	Descripción
DataPager	Proporciona funcionalidad de paginación para los controles enlazados a datos que implementan la interface *System.Web.UI.WebControls.IpageableItemContainer*, como el control *System.Web.UI.WebControls.ListView.*
EntityDataSource	Representa un modelo de entidad de datos (*EDM*) a controles enlazados a datos en una aplicación *ASP.NET*.

LinkDataSource	Permite el uso de *Language-Integrated Query* (*LINK*) en una página web *ASP.NET* a través de texto de marcado para recuperar y modificar los datos de un objeto de datos.
ListView	Muestra los valores de un origen de datos utilizando plantillas definidas por el usuario. El control *System.Web.UI.WebControls.ListView* permite a los usuarios seleccionar, ordenar, eliminar, modificar e insertar registros.

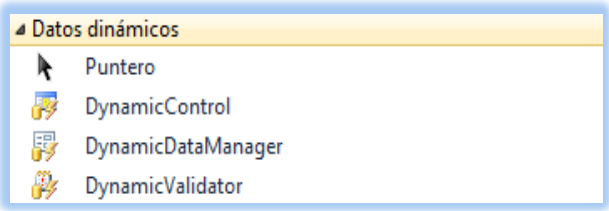

Figura 34. Controles de Datos Dinámicos: *ASP.NET* con el Framework 3.5.

Control	Descripción
DynamicControl	Muestra el contenido definido para el campo en los controles enlazados a datos con plantilla, utilizando las características de datos dinámicos de *ASP.NET*.
DynamicDataManager	Habilita el comportamiento dinámico para los controles web de *ASP.NET* que admiten datos dinámicos de *ASP.NET*.
DynamicValidator	Exige y detecta excepciones que se inician en un modelo de datos y muestra el error.

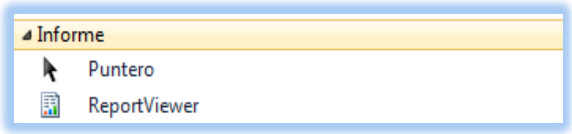

Figura 35. Control de Informe: *ASP.NET* con el Framework 3.5.

Control	Descripción
ReportViewer	Encapsula métodos y propiedades utilizados para diseñar y presentar reportes dinámicos.

1.14 Evolución del Toolbox *ASP.NET* con el Frameworks 4

En esta versión de la plataforma de .NET Framework se han mantenido todos los controles detallados anteriormente para la plataforma de .NET Framework 2.0, .NET Framework 3.0 y .NET Framework 3.5 no obstante con algunas actualizaciones y mejoras en cada uno de los controles.

Para esta plataforma solamente se han agregado los siguientes ***nuevos controles*** *al Toolbox de ASP.NET:*

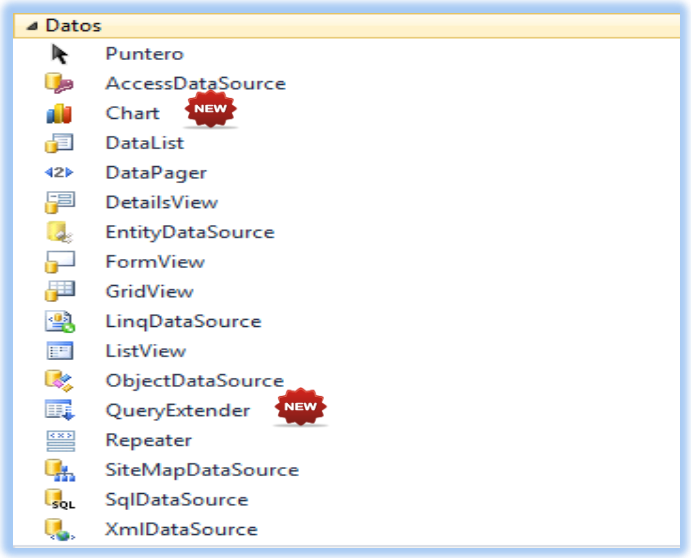

Figura 36. ***Nuevos*** Controles de Datos: *ASP.NET* con el Framework 4.

Control	Descripción
Chart	Sirve como clase raíz del control Chart. Esta clase expone todas las propiedades, métodos y eventos del control Chart Web.
QueryExtender	Habilita el filtrado de los datos de un origen de datos sin una cláusula *Where* explícita en el origen de datos.

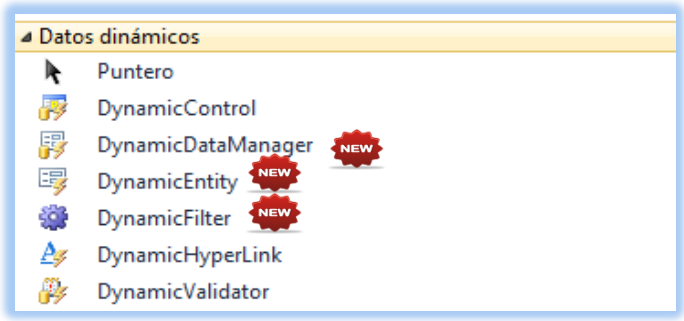

Figura 37. ***Nuevos*** Controles de Datos Dinámicos: *ASP.NET* con el Framework 4.

Control	Descripción
DynamicEntity	Proporciona un control *ASP.NET* que representa una entidad para su uso por parte de datos dinámicos de *ASP.NET*.
DynamicFilter	Muestra la interface de usuario para filtrar filas de tabla mediante una columna especificada.
DynamicHyperlink	Muestra vínculos a acciones de tabla como editar, eliminar e insertar.

1.15 Técnicas Avanzadas *CSS* (Hojas de estilo en cascada)

1.15.1 Selectores *CSS* 2.1

✔ **Selector universal**

Se utiliza para seleccionar todos los elementos de la página, el siguiente ejemplo elimina el margen y el relleno de todos los elementos *HTML*.

```
*
{
        margin: 0;
        padding: 0;
}
```

✔ **Selector de tipo o etiqueta**

Selecciona todos los elementos de la página cuya etiqueta *HTML* coincide con el valor del selector, el siguiente ejemplo selecciona todos los *h1* de la página:

```
h1
{
        propiedad: valor;
}
```

✔ **Selector descendente**

Selecciona los elementos que se encuentran dentro de otros elementos, un elemento es descendiente de otro cuando se encuentra entre las etiquetas de apertura y de cierre del otro elemento, el selector del siguiente ejemplo selecciona todos los elementos <*span*> de la página que se encuentren dentro de un elemento <*p*>:

p span

{

propiedad: valor;

}

✔ Selector de clase

Si se considera el siguiente código *HTML* de ejemplo:

<body>

* *Descripción 1* *

* *Descripción 2* *

* *Descripción 3* *

</body>

¿Cómo se pueden aplicar estilos *CSS* sólo al primer párrafo?, el selector universal (*) no se puede utilizar porque selecciona todos los elementos de la página, el selector de tipo o etiqueta (*p*) tampoco se puede utilizar porque seleccionaría todos los párrafos, por último, el selector descendente (*body p*) tampoco se puede utilizar porque todos los párrafos se encuentran en el mismo sitio.

A continuación, se crea en el archivo *CSS* una nueva regla llamada destacado con todos los estilos que se van a aplicar al elemento, para que el navegador no confunda este selector con los otros tipos de selectores, se prefija el valor del atributo *class* con un punto (**.**) tal y como muestra el siguiente ejemplo:

.*destacado*

{

propiedad: valor;

}

✔ Selectores de ID

En ocasiones, es necesario aplicar estilos *CSS* a un único elemento de la página, aunque puede utilizarse un selector de clase para aplicar estilos a un único elemento, existe otro selector más eficiente en este caso.

El selector de *ID* permite seleccionar un elemento de la página a través del valor de su atributo *id*, este tipo de selectores sólo seleccionan un elemento de la página porque el valor del atributo *id* no se puede repetir en dos elementos diferentes de una misma página.

La sintaxis de los selectores de *ID* es muy parecida a la de los selectores de clase, salvo que se utiliza el símbolo de almohadilla (**#**) en vez del punto (**.**) como prefijo del nombre de la regla *CSS*:

```
#destacado
{
        propiedad: valor;
}
```

✔ Selector de hijos

Se trata de un selector similar al selector descendente, pero muy diferente en su funcionamiento, se utiliza para seleccionar un elemento que es hijo directo de otro elemento y se indica mediante el "signo de mayor que" (>):

```
p > span
{
        propiedad: valor;
}
```

```
<p><span>Texto1</span></p>
<p><a href="#"><span>Texto2</span></a></p>
```

En el ejemplo anterior, el selector *p > span* se interpreta como "cualquier elemento <*span*> que sea hijo directo de un elemento <*p*>", por lo que el primer elemento <*span*> cumple la condición del selector, sin embargo, el segundo elemento <*span*> no la cumple porque es descendiente pero no es hijo directo de un elemento <*p*>.

✔ Selector adyacente

El selector adyacente utiliza el signo + y su sintaxis es:

elemento1 + *elemento2* { ... }

La explicación del comportamiento de este selector no es sencilla, ya que selecciona todos los elementos de tipo *elemento2* que cumplan las dos siguientes condiciones:

- ✓ *elemento1* y *elemento2* deben ser hermanos, por lo que su elemento padre debe ser el mismo.
- ✓ *elemento2* debe aparecer inmediatamente después de *elemento1* en el código *HTML* de la página.

En el siguiente ejemplo:

```
h1 + h2
{
    propiedad: valor
}
<body>
    <h1>Titulo1</h1>
    <h2>Subtítulo</h2>
    <h2>Otro subtítulo</h2>
</body>
```

Los estilos del selector *h1 + h2* se aplican al primer elemento <*h2*> de la página, pero no al segundo <*h2*>, ya que:

- ✓ El elemento padre de <*h1*> es <*body*>, el mismo padre que el de los dos elementos <*h2*>, así, los dos elementos <*h2*> cumplen la primera condición del selector adyacente.

- ✓ El primer elemento <*h2*> aparece en el código *HTML* justo después del elemento <*h1*>, por lo que este elemento <*h2*> también cumple la segunda condición del selector adyacente.

- ✓ Por el contrario, el segundo elemento <*h2*> no aparece justo después del elemento <*h1*>, por lo que no cumple la segunda condición del selector adyacente y por tanto no se le aplican los estilos de *h1 + h2*.

✔ Selector de atributos

El último tipo de selectores avanzados lo forman los selectores de atributos, que permiten seleccionar elementos *HTML* en función de sus atributos y/o valores de esos atributos.

Los cuatro tipos de selectores de atributos son:

- ✓ *elemento[nombre_atributo]*, selecciona los elementos que tienen establecido el atributo llamado nombre_atributo, independientemente de su valor.

- ✓ *elemento[nombre_atributo=valor]*, selecciona los elementos que tienen establecido un atributo llamado nombre_atributo con un valor igual a valor.

- ✓ *elemento[nombre_atributo~=valor]*, selecciona los elementos que tienen establecido un atributo llamado nombre_atributo y al menos uno de los valores del atributo es valor.

- ✓ *elemento[nombre_atributo|=valor]*, selecciona los elementos que tienen establecido un atributo llamado nombre_atributo y cuyo valor es una serie de palabras separadas con guiones, pero que comienza con valor, este tipo de selector sólo es útil para los atributos de tipo *lang* que indican el idioma del contenido del elemento.

1.15.2 Novedades en los selectores de *CSS* 3

En primer lugar, CSS 3 añade tres nuevos selectores de atributos:

- ✔ *elemento[atributo^="valor"]*, selecciona todos los elementos que disponen de ese atributo y cuyo valor comienza exactamente por la cadena de texto indicada.

- ✔ *elemento[atributo$="valor"]*, selecciona todos los elementos que disponen de ese atributo y cuyo valor termina exactamente por la cadena de texto indicada.

- ✔ *elemento[atributo*="valor"]*, selecciona todos los elementos que disponen de ese atributo y cuyo valor contiene la cadena de texto indicada.

1.15.3 pseudo-elementos en *CSS* 2.1

- ***:first-line***, selecciona la primera línea del texto de un elemento.
- ***:first-letter***, selecciona la primera letra del texto de un elemento.
- ***:before***, selecciona la parte anterior al contenido de un elemento para insertar nuevo contenido generado.
- ***:after***, selecciona la parte posterior al contenido de un elemento para insertar nuevo contenido generado.

1.15.4 pseudo-elementos en *CSS* 3

- ***::first-line***, selecciona la primera línea del texto de un elemento.
- ***::first-letter***, selecciona la primera letra del texto de un elemento.
- ***::before***, selecciona la parte anterior al contenido de un elemento para insertar nuevo contenido generado.
- ***::after***, selecciona la parte posterior al contenido de un elemento para insertar nuevo contenido generado.
- ***::selecion***, selecciona el texto que ha seleccionado un usuario con su ratón o teclado.

1.15.5 pseudo-clases en *CSS* 3

Las mayores novedades de *CSS 3* se producen en las *pseudo-clases*, ya que se añaden 12 nuevas, entre las cuales se encuentran:

- *elemento:nth-child(numero)*, selecciona el elemento indicado pero con la condición de que sea el hijo enésimo de su padre, este selector es útil para seleccionar el segundo párrafo de un elemento, el quinto elemento de una lista, etc.

- *elemento:nth-last-child(numero)*, idéntico al anterior pero el número indicado se empieza a contar desde el último hijo.

- *elemento:empty*, selecciona el elemento indicado pero con la condición de que no tenga ningún hijo, la condición implica que tampoco puede tener ningún contenido de texto.

- *elemento:first-child* y *elemento:last-child*, seleccionan los elementos indicados pero con la condición de que sean respectivamente los primeros o últimos hijos de su elemento padre.

- *elemento:nth-of-type(numero)*, selecciona el elemento indicado pero con la condición de que sea el enésimo elemento hermano de ese tipo.

- *elemento:nth-last-of-type(numero)*, idéntico al anterior pero el número indicado se empieza a contar desde el último hijo.

1.15.6 Box Model en CSS

El modelo de cajas o "*box model*" es seguramente la característica más importante del lenguaje de hojas de estilos *CSS*, ya que condiciona el diseño de todas las páginas web.

El "*box model*" es el comportamiento de *CSS* que hace que todos los elementos incluidos en una página *HTML* se representen mediante cajas rectangulares, *CSS* permite controlar el aspecto de todas las cajas, las cajas de una página se crean automáticamente.

Cada vez que se inserta una etiqueta o elemento en la página, se crea una nueva caja rectangular que encierra los contenidos del elemento.

Cada una de las cajas está formada por seis partes, tal y como se muestra en la siguiente imagen tridimensional:

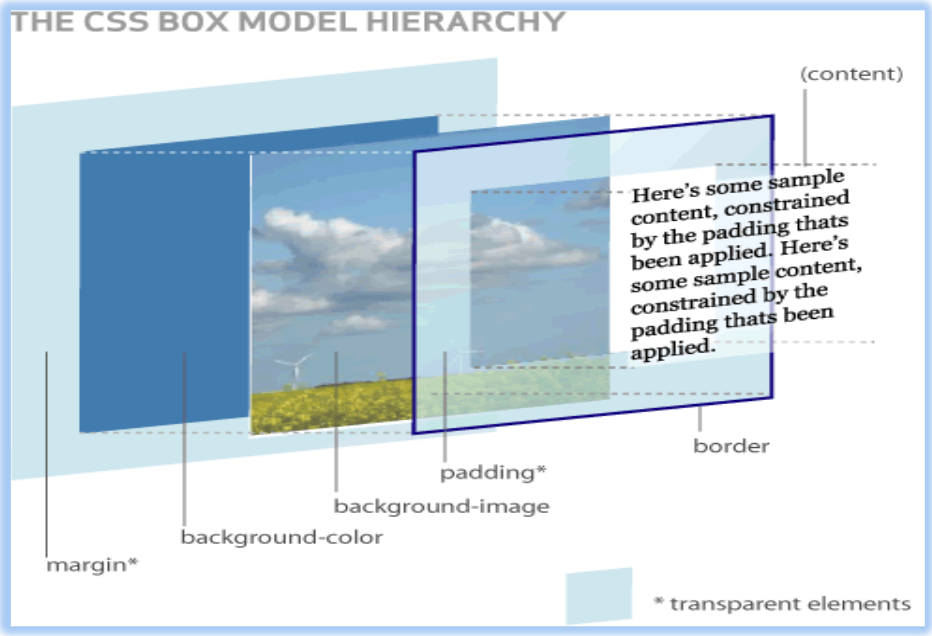

Figura 38. Box Model en CSS.

Las partes que componen cada caja y su orden de visualización desde el punto de vista del usuario son las siguientes:

- *Contenido (**content**):* se trata del contenido *HTML* del elemento (las palabras de un párrafo, una imagen, el texto de una lista de elementos, etc.)

- *Relleno (**padding**):* espacio libre opcional entre el contenido y el borde que lo encierra.

- *Borde (**border**):* línea que encierra completamente el contenido y su relleno.

- *Imagen de fondo (**background-image**):* imagen que se muestra por detrás del contenido y el espacio de relleno.

- *Color de fondo (**background-color**):* color que se muestra por detrás del contenido y el espacio de relleno.

- *Margen (**margin**):* espacio libre entre la caja y las posibles cajas adyacentes.

1.15.7 Posicionamiento y visualización

Para cumplir con el modelo de cajas presentado en la sección anterior, los navegadores crean una caja para representar a cada elemento de la página *HTML*.

Los factores que se tienen en cuenta para generar cada caja son:

- Las propiedades *width* y *height* de la caja (si están establecidas).
- El tipo de cada elemento *HTML* (*elemento de bloque o elemento en línea*).
- Posicionamiento de la caja (*normal*, *relative*, *absolute*, *fixed* o *float*).

- Las relaciones entre elementos (dónde se encuentra cada elemento, elementos descendientes, etc).
- Otro tipo de información, como por ejemplo el tamaño de las imágenes y el tamaño de la ventana del navegador.

1.15.8 El estándar de *CSS* define cinco modelos para posicionar una caja:

- *Posicionamiento normal o estático (**static**):* se trata del posicionamiento que utilizan los navegadores si no se indica lo contrario.

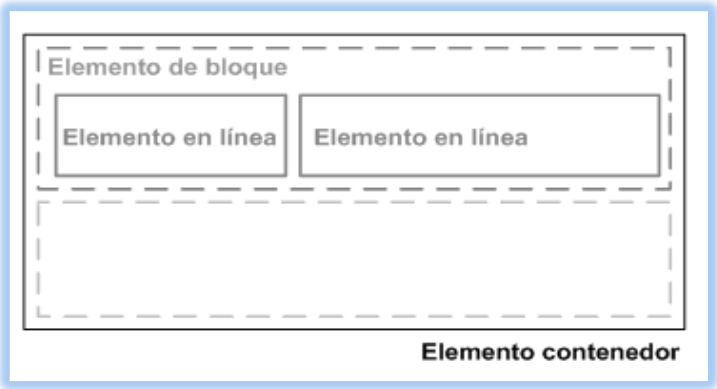

Figura 39. Posicionamiento de cajas CSS: normal o estático (***static***).

- *Posicionamiento relativo (**relative**):* variante del posicionamiento normal que consiste en posicionar una caja según el posicionamiento normal y después desplazarla respecto de su posición original.

Figura 40. Posicionamiento de cajas CSS: relativo (***relative***).

- *Posicionamiento absoluto (**absolute**):* la posición de una caja se establece de forma absoluta respecto de su elemento contenedor y el resto de elementos de la página ignoran la nueva posición del elemento.

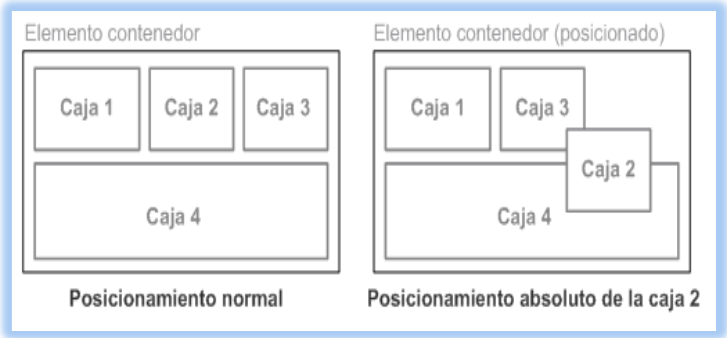

Figura 41. Posicionamiento de cajas CSS: absoluto (***absolute***).

- *Posicionamiento fijo (**fixed**):* variante del posicionamiento absoluto que convierte una caja en un elemento inamovible, de forma que su posición en la pantalla siempre es la misma independientemente del resto de elementos e independientemente de si el usuario sube o baja la página en la ventana del navegador.

- *Posicionamiento flotante (**float**):* se trata del modelo más especial de posicionamiento, ya que desplaza las cajas todo lo posible hacia la izquierda o hacia la derecha de la línea en la que se encuentran.

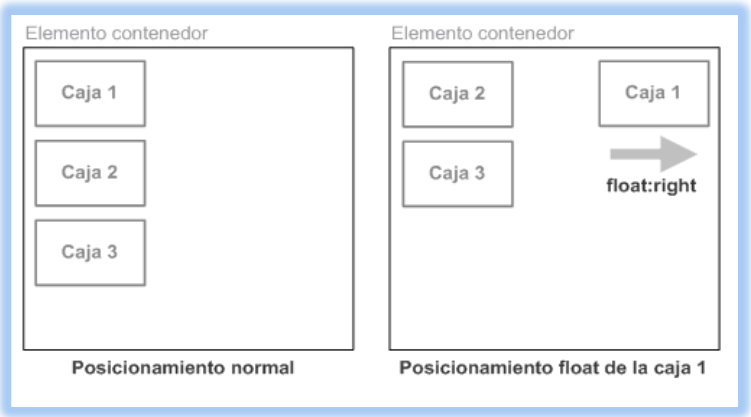

Figura 42. Posicionamiento de cajas CSS: flotante (***float***).

1.15.9 Propiedades display y visibility

Las propiedades *display* y *visibility* controlan la visualización de los elementos.

Las dos propiedades permiten ocultar cualquier elemento de la página, habitualmente se utilizan junto con *javascript* para crear efectos dinámicos como mostrar y ocultar determinados textos o imágenes cuando el usuario pincha sobre ellos.

La propiedad *display* permite ocultar completamente un elemento haciendo que desaparezca de la página, como el elemento oculto no se muestra, el resto de elementos de la página se mueven para ocupar su lugar.

Por otra parte, la propiedad *visibility* permite hacer invisible un elemento, lo que significa que el navegador crea la caja del elemento pero no la muestra, en este caso, el resto de elementos de la página no modifican su posición, ya que aunque la caja no se ve, sigue ocupando sitio.

La siguiente imagen muestra la diferencia entre ocultar la caja número 5 mediante la propiedad *display* o hacerla invisible mediante la propiedad *visibility*:

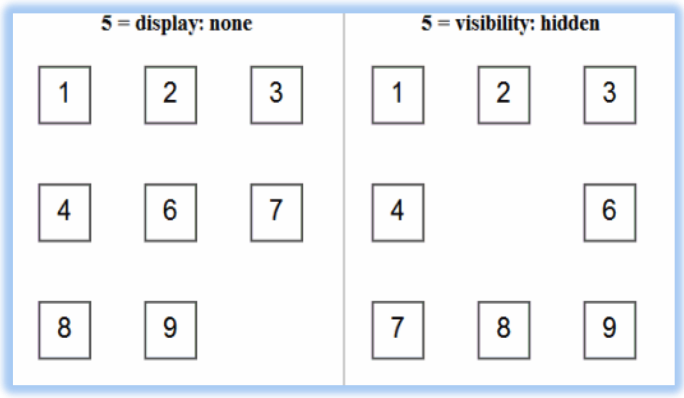

Figura 43. Propiedades avanzadas *CSS*: *display* y *visibility*

1.15.10 Propiedad z-index

Además de posicionar una caja de forma horizontal y vertical, *CSS* permite controlar la posición tridimensional de las cajas posicionadas.

De esta forma, es posible indicar las cajas que se muestran delante o detrás de otras cajas cuando se producen solapamientos.

La posición tridimensional de un elemento se establece sobre un tercer eje llamado *Z* y se controla mediante la propiedad *z-index*, utilizando esta propiedad es posible crear páginas complejas con varios niveles o capas.

El valor más común de la propiedad *z-index* es un número entero.

Aunque la especificación oficial permite los números negativos, en general se considera el número 0 como el nivel más bajo.

Cuanto más alto sea el valor numérico, más cerca del usuario se muestra la caja.

Un elemento con *z-index:* 10 se muestra por encima de los elementos con *z-index*: 8 o *z-index*: 9, pero por debajo de los elementos con *z-index*: 20 o *z-index*: 50.

La siguiente imagen muestra un ejemplo de uso de la propiedad *z-index*:

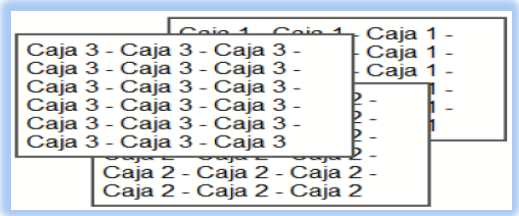

Figura 44. Propiedad avanzada *CSS*: *z-index*

1.15.11 Uso de una de las técnicas más avanzadas: sprites

El primer paso en la utilización de la técnica de sprites consiste en crear una imagen grande que incluya las imágenes individuales.

Como los iconos son cuadrados de tamaño *32px*, se crea una imagen de *32px* x *128px* para el ejemplo:

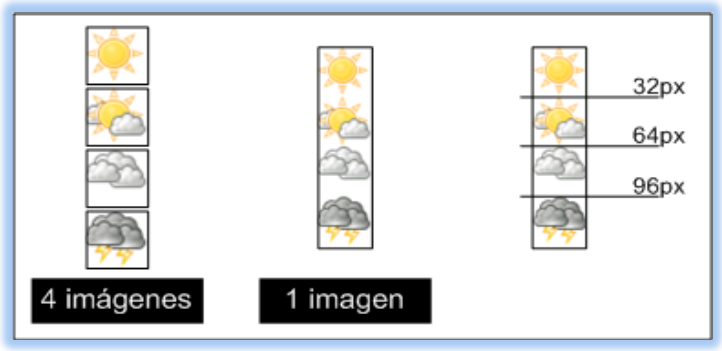

Figura 45. Uso de una de las técnicas más avanzadas *CSS*: sprites.

Para facilitar el uso de esta técnica, todas las imágenes individuales ocupan el mismo sitio dentro de la imagen grande, de esta forma, los cálculos necesarios para desplazar la imagen de fondo se simplifica al máximo.

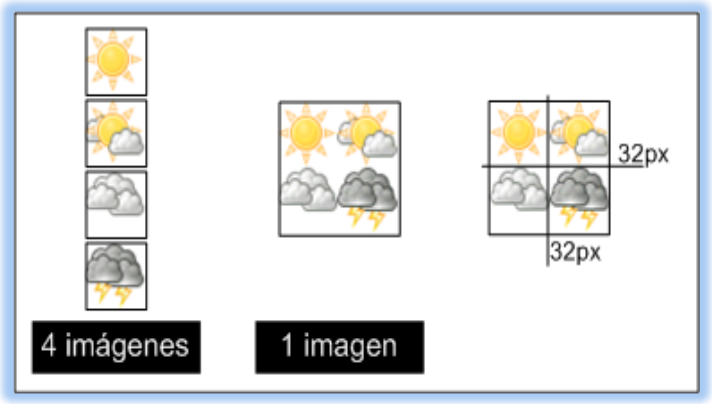

Figura 46. Uso de una de las técnicas más avanzadas *CSS*: sprites.

Aparentemente, utilizar este nuevo sprite sólo implica que la imagen de fondo se debe desplazar también horizontalmente:

A continuación se describe el código CSS requerido para este ejemplo:

```
#localidad1, #localidad2, #localidad3, #localidad4
{
        padding-left: 38px;
        height: 32px;
        line-height: 32px;
        background-image: url("images/sprite.png");
        background-repeat: no-repeat;
}
#localidad1
{
        background-position: 0 0;
}
#localidad2
{
        background-position: -32px 0;
}
#localidad3
{
        background-position: 0 -32px;
}
#localidad4
{
        background-position: -32px -32px;
}
```

El problema del sprite anterior es que cuando una imagen tiene a su derecha o a su izquierda otras imágenes, estas también se ven:

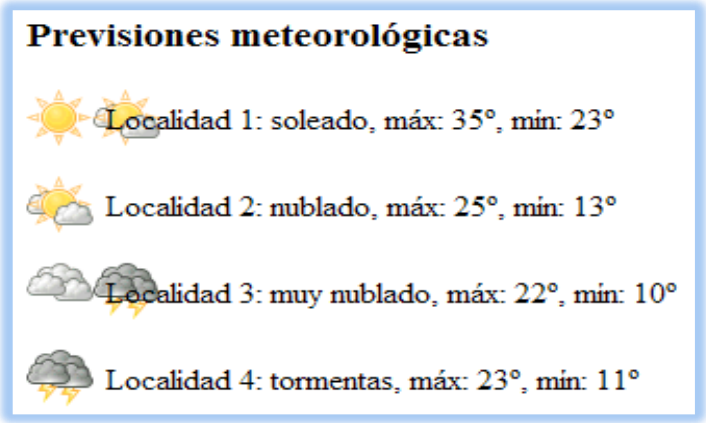

Figura 47. Uso de una de las técnicas más avanzadas *CSS*: sprites.

Conclusión: La solución más loable es limitar su tamaño para que no se visualicen las imágenes que se encuentran cerca.

La mayoría de sitios web profesionales utilizan sprites para mostrar sus imágenes de adorno.

La siguiente imagen muestra el sprite del sitio web *YouTube*:

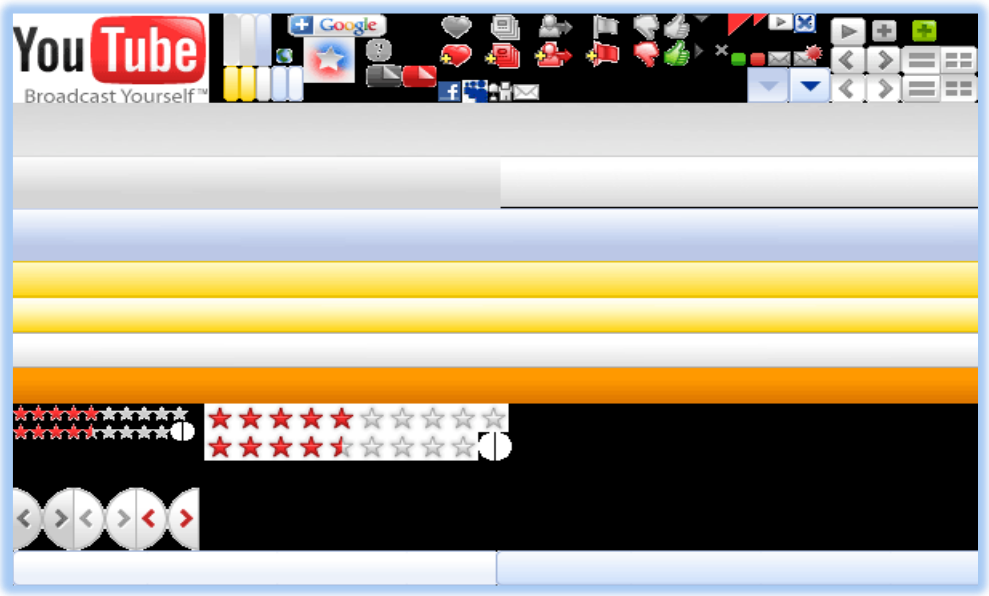

Figura 48. Sprite del sitio web profesional *YouTube*.

La siguiente imagen muestra un sprite complejo que incluye 241 imágenes que *sólo ocupa 42 KB de espacio de almacenamiento:*

Figura 49. Sprite completo con una colección de 241 imágenes.

1.16 *Callback* (Devolución de llamadas asíncronas)

Una devolución de llamada o retrollamada (en inglés: *callback*) es una función que recibe como argumento la dirección o puntero de otra función, cuando la retrollamada es llamado este recurre al puntero de la función y la ejecuta, esto permite desarrollar capas de abstracción de código genérico a bajo nivel que pueden ser llamadas desde una subrutina (o función) definida en una capa de mayor nivel.

Usualmente, el código de alto-nivel inicia con el llamado de alguna función, definida a bajo-nivel, pasando a esta un puntero, o un puntero inteligente (conocido como *handle*), de alguna función, mientras la función de bajo-nivel se ejecuta, esta puede ejecutar la función pasada como puntero para realizar alguna tarea, en otro escenario, las funciones de bajo nivel registran las funciones pasadas como un *handle* y luego pueden ser usadas de modo asincrónico.

Una retrollamada puede ser usada como una aproximación simple al *polimorfismo* y a la programación genérica, donde el comportamiento de una función puede ser

dinámicamente determinado por el paso de punteros a funciones o *handles* a funciones de bajo nivel que aunque realicen tareas diferentes los argumentos sean compatibles entre sí.

Esta es una técnica de mucha importancia por lo que se la llama código reutilizable.

1.16.1 Características de *Callback*

Para entender los motivos por usar retrollamadas, se puede considerar como ejemplo el problema de realizar varias operaciones arbitrarias en una lista, una opción puede ser iterar sobre la lista, o también realizar alguna operación sobre cada uno de los elementos de la lista, en la práctica, la solución más común, pero no ideal, es utilizar iteradores como un bucle *for*, *for each*) que deberá ser duplicado en cada lugar del código donde sea necesario, más aún, si la lista es actualizada por un proceso asíncrono (por ejemplo, si un elemento es añadido o eliminado), el iterador podría ser corrupto durante el paso a través de la lista.

Una alternativa podría ser crear una nueva biblioteca de funciones que ejecute la tarea deseada con la sincronización apropiada en cada caso, esta propuesta aún requiere que cada nueva función de la biblioteca contenga el código para ir a través de la lista.

Esta solución no es aceptable para bibliotecas genéricas que tengan como objetivo varias aplicaciones; *el desarrollador de la biblioteca de clases no puede anticiparse a las necesidades de cada aplicación, y el desarrollador de las aplicaciones no debería necesitar conocer los detalles de la implementación de la biblioteca de clases*.

En este caso las retrollamadas resuelven estos problemas, un procedimiento es escribir el paso a través de una lista que provee a la aplicación del código para ir a través de la lista y operando sobre cada elemento.

Hay una distinción clara entre la biblioteca de clases y la aplicación sin sacrificar la flexibilidad, *una retrollamada puede también considerarse un tipo de rutina enlazada por referencia.*

1.16.2 Funciones *Callback*

Una función de devolución de llamada es código de una aplicación administrada que ayuda a las funciones no administradas de un archivo *dll* a completar sus tareas.

Las llamadas a una función de devolución de llamada se pasan indirectamente desde una aplicación administrada, a través de una función de un archivo *dll*, y de vuelta a la implementación administrada, algunas de las muchas funciones de archivos *dll* a las que se llama con la invocación de plataforma requieren una función de devolución de llamada en código administrado para ejecutarse correctamente.

Para llamar a la mayoría de las funciones de archivos *dll* desde *código administrado*, se crea una definición administrada de la función y, a continuación, se la llama.

Para utilizar una función de un archivo *dll* que requiere una función de devolución de llamada hay que llevar a cabo ciertos pasos adicionales, en primer lugar, hay que determinar si la función requiere una devolución de llamada estudiando la documentación de la función, después, hay que crear la función de devolución de llamada en la aplicación administrada, finalmente, se llama a la función del archivo *dll*, pasándole un puntero a la función de devolución de llamada como argumento.

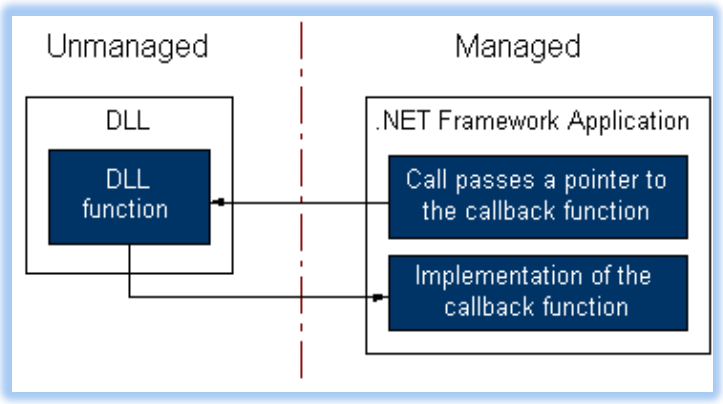

Figura 50. Esquema de la interoperabilidad de las funciones *Callback*.

Las funciones de devolución de llamada son ideales para utilizarlas en situaciones en las que una tarea se realiza repetidas veces, otro uso frecuente es con funciones de enumeración, como *EnumFontFamilies*, *EnumPrinters* y *EnumWindows* en la *API* Win32.

La función *EnumWindows* enumera todas las ventanas existentes en su equipo, llamando a la función de devolución de llamada para realizar una tarea en cada ventana.

1.16.3 Ejemplo de una función *Callback*

```
/*
    Ω_____________________________________Ð
    |                                     |
    |    http://www.magodeveloper.com/    |
    |                                     |
    |  © Todos los derechos reservados.   |
    |_____________________________________|
*/

Type.registerNamespace("Mago.Developer.Client");

Mago.Developer.Client.CallBack = function ()
{
    Mago.Developer.Client.CallBack.initializeBase(this);
    this.async = true;
    this.targetID = null;
    this.updateViewState = true;
    this.callbackDelegates = [];
}

Mago.Developer.Client.CallBack.prototype = {
    initialize: function ()
    {
        Mago.Developer.Client.CallBack.callBaseMethod(this, 'initialize');
    },
     registerMethod: function (methodName, returnType, successfulDelegate,
errorDelegate)
```

```
    {
        Array.add(this.callbackDelegates, { "method": methodName, "returnType":
returnType, "onsuccess": successfulDelegate, "onerror": errorDelegate });
        this[methodName] = function ()
        {
            var args = [];
            for (var i = 0; i < arguments.length; i++)
                Array.add(args, arguments[i]);
            this.invokeServerCall({ "method": methodName, "params": args });
        }
    },
    invokeServerCall: function (serverArgs)
    {
        if (String.isNullOrEmpty(this.get_targetId()))
            throw "Target ID no puede ser nulo.";
        var registedMethod = null;
        for (var i = 0; i < this.callbackDelegates.length; i++)
        {
            if (this.callbackDelegates[i].method == serverArgs.method)
            {
                registedMethod = this.callbackDelegates[i];
                break;
            }
        }
        if (registedMethod == null)
            throw "No se ha encontrado un método válido que se haya
registrado.";
        uniqueID = this.get_targetId().replace(/\_/g, '$');
        jsonStr = null;
        if (serverArgs != null)
            jsonStr =
Sys.Serialization.JavaScriptSerializer.serialize(serverArgs);

        theFormPostData = null;
        WebForm_InitCallback();
        registedMethod.updateViewState = this.get_updateViewState();
```

```
        WebForm_DoCallback(uniqueID, jsonStr, this.callbackSuccessfulHandler,
registedMethod, this.callbackErrorHandler, this.async);
    },
    callbackErrorHandler: function (response, context)
    {
        context.onerror(response);
    },
    callbackSuccessfulHandler: function (response, context)
    {
        data = response.split(";");
        if (context.updateViewState)
            $get("VIEWSTATE").value = data[0];
        var result = null;
        if (context.returnType == "json")
        {
            if (!String.isNullOrEmpty(data[1]))
                result =
Sys.Serialization.JavaScriptSerializer.deserialize(data[1]);
        }
        else
            result = data[1];
        context.onsuccess(result);
    },
    makeServerCall: function (method, params)
    {
        if (method == "" || method == undefined || method == null)
            throw "El método es nulo o no ha sido registrado correctamente.";
        invokeServerCall(params != undefined && params != null ? { "method":
method, "params": Sys.Serialization.JavaScriptSerializer.serialize(params)} : {
"method": method });
    },
    get_targetId: function ()
    {
        return this.targetID;
    },
```

```
    set_targetId: function (value)
    {
        this.targetID = value;
    },
    get_async: function ()
    {
        return this.async;
    },
    set_async: function (value)
    {
        this.async = value;
    },
    get_updateViewState: function ()
    {
        return this.updateViewState;
    },
    set_updateViewState: function (value)
    {
        this.updateViewState = value;
    },
    dispose: function ()
    {
        Mago.Developer.Client.CallBack.callBaseMethod(this, 'dispose');
    }
}
Mago.Developer.Client.CallBack.registerClass('Mago.Developer.Client.CallBack',
Sys.Component);
if (typeof (Sys) !== 'undefined') Sys.Application.notifyScriptLoaded();
```

1.16.4 *Callback* en las aplicaciones cliente

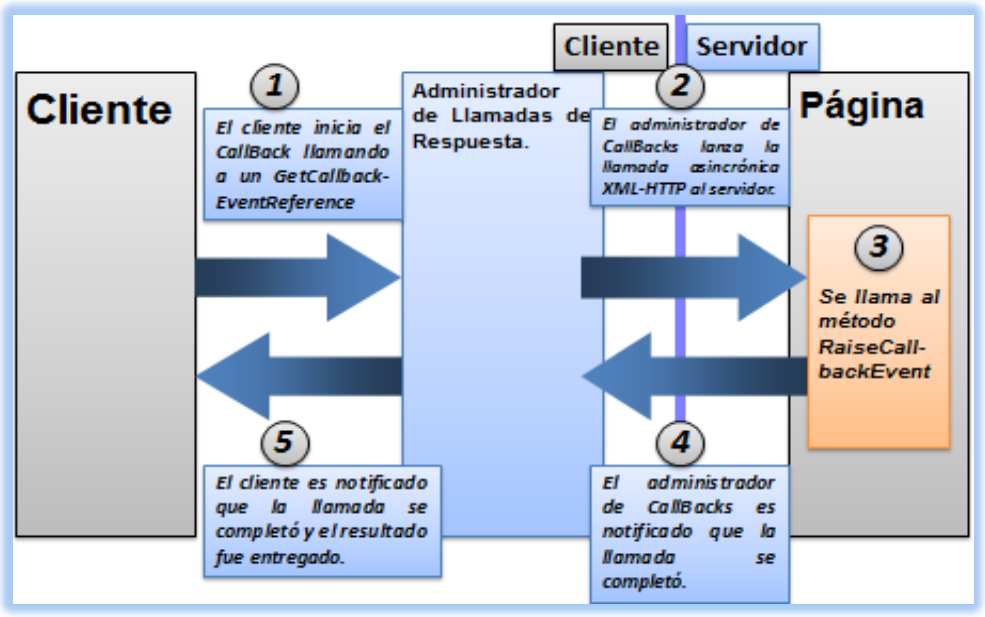

Figura 51. *Callback* en las aplicaciones cliente.

1.17 Enfoque aplicativo en el desarrollo de biblioteca de clases (*componente*).

1.17.1 Configurar las propiedades de un proyecto de biblioteca de clases.

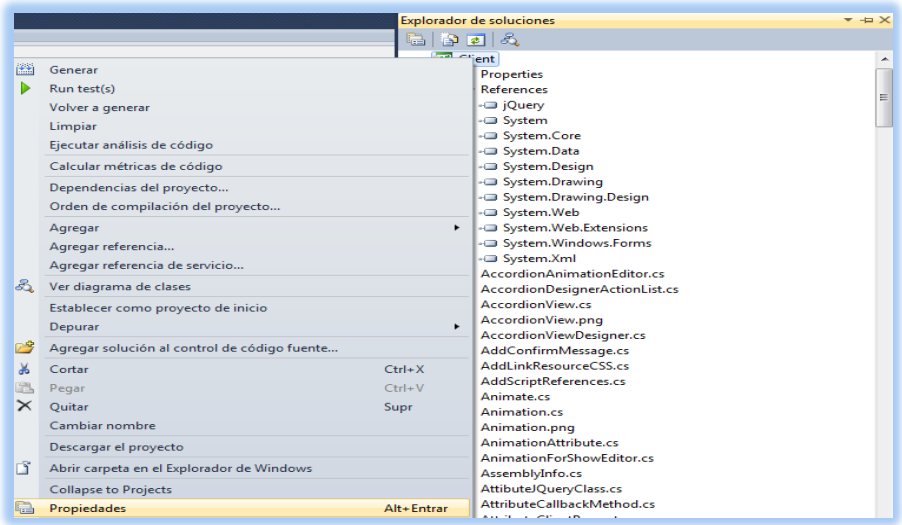

Figura 52. Configurando las propiedades de un proyecto de biblioteca de clases: *Paso 1*.

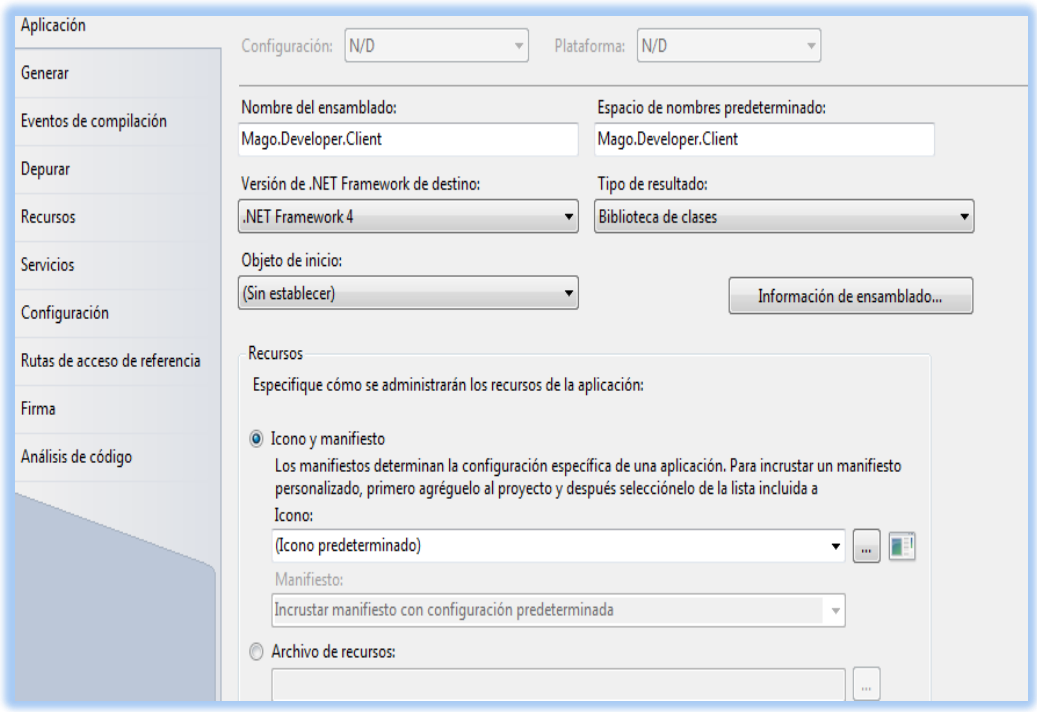

Figura 53. Configurando las propiedades de un proyecto de biblioteca de clases: *Paso 2*.

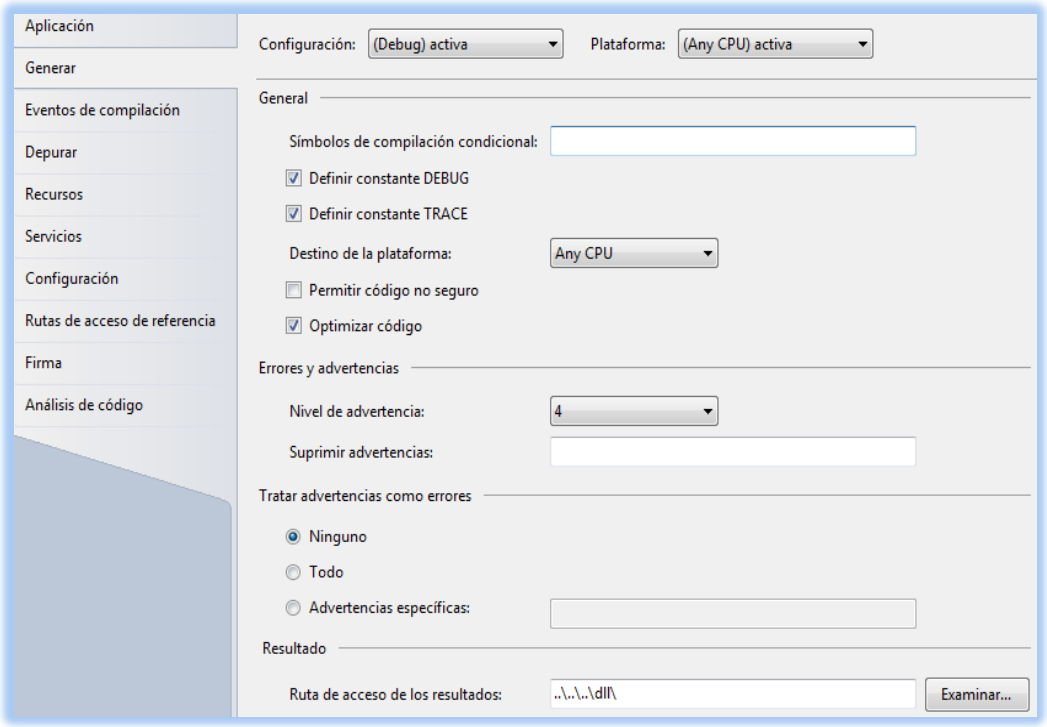

Figura 54. Configurando las propiedades de un proyecto de biblioteca de clases: *Paso 3*.

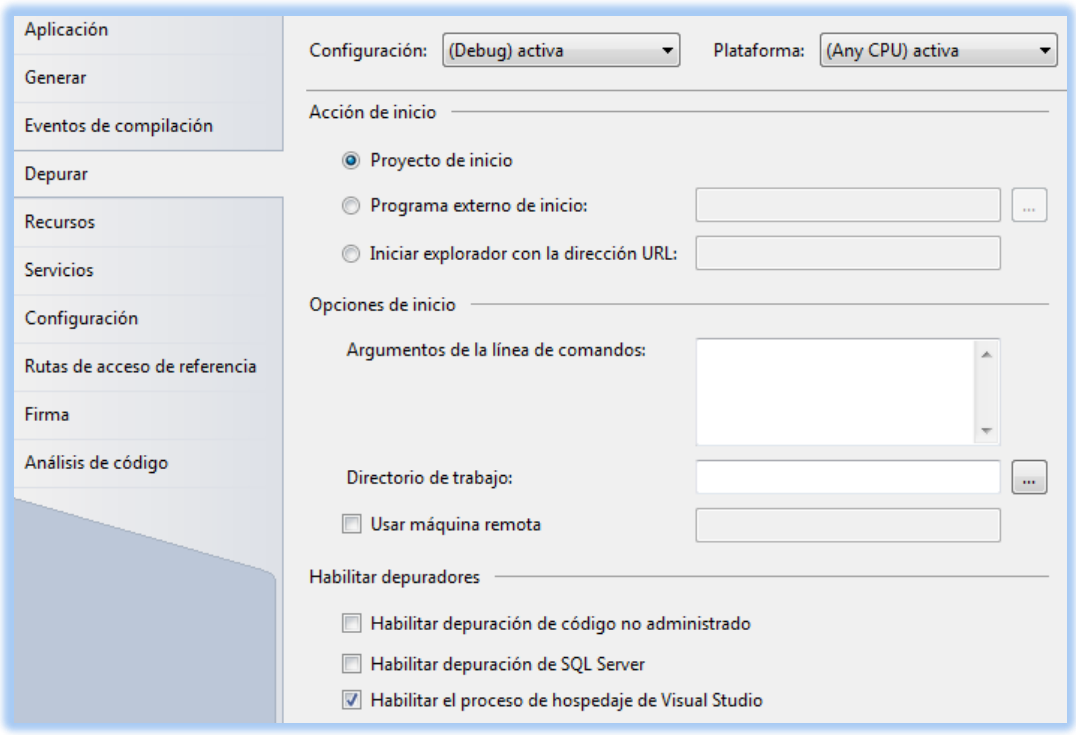

Figura 55. Configurando las propiedades de un proyecto de biblioteca de clases: *Paso 4*.

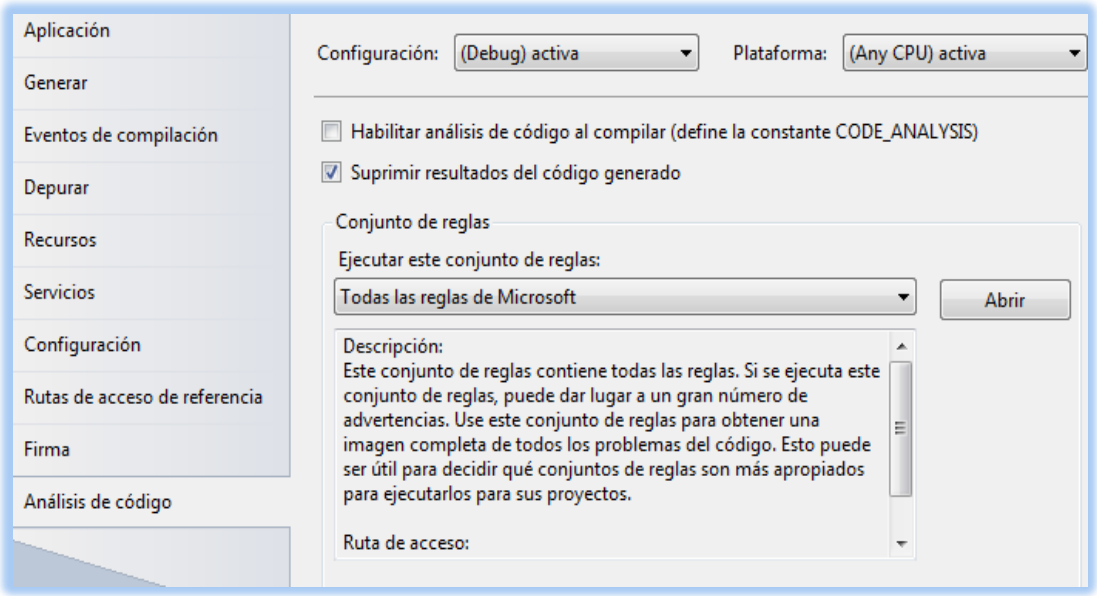

Figura 56. Configurando las propiedades de un proyecto de biblioteca de clases: *Paso 5*.

1.17.2 Agregar referencias a un proyecto de biblioteca de clases (*componente*).

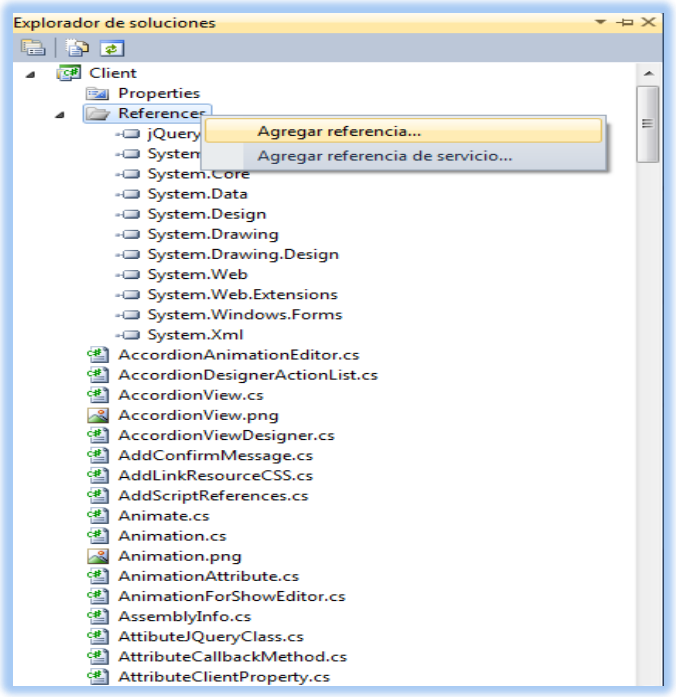

Figura 57. Agregando referencias a un proyecto de biblioteca de clases: *Paso 1*.

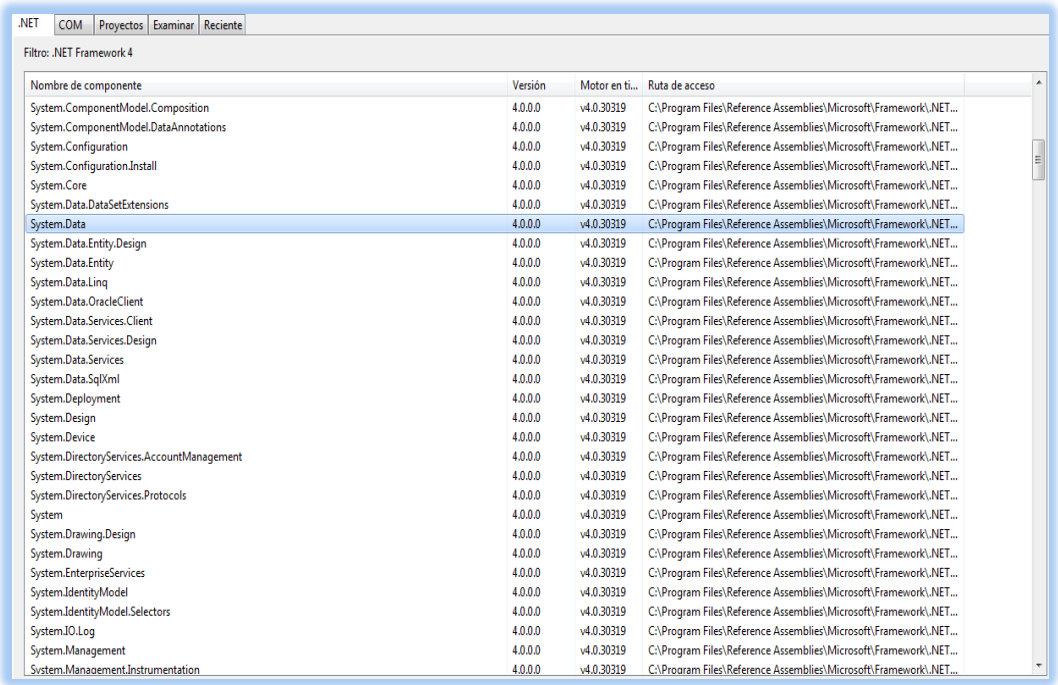

Figura 58. Agregando referencias a un proyecto de biblioteca de clases: *Paso 2*.

1.17.3 Configuración básica del archivo de ensamblado (*AssemblyInfo.cs*).

```
/*
   Ω_____________________________________Ð
    |                                     |
    |    http://www.magodeveloper.com/    |
    |                                     |
    |  © Todos los derechos reservados.   |
    |_____________________________________|
*/


using System;
using System.Web.UI;
using System.Reflection;
using System.Diagnostics;
using System.Runtime.InteropServices;
using System.Runtime.CompilerServices;

[assembly: SuppressIldasm()]
[assembly: ComVisible(false)]
[assembly: CompilationRelaxations(8)]
[assembly: AssemblyVersion("1.0.0.0")]
[assembly: AssemblyFileVersion("1.0.0.0")]
[assembly: AssemblyTitle("Mago.Developer.Client")]
[assembly: TagPrefix("Mago.Developer.Client", "mago")]
[assembly: Guid("6D68928E-4400-455E-87A2-7B2E7FBE283B")]
[assembly: AssemblyTrademarkAttribute("Mago Developer")]
[assembly: AssemblyProduct("Mago.Developer.Client 2011")]
[assembly: AssemblyDefaultAlias("Mago.Developer.Client")]
[assembly: AssemblyCompany("http://www.magodeveloper.com/")]
[assembly:
Debuggable(DebuggableAttribute.DebuggingModes.IgnoreSymbolStoreSequencePoints)]
[assembly: AssemblyCopyright("\x00a9 Copyright 2011 Todos los derechos
reservados. (http://www.magodeveloper.com/)")]
```

1.17.4 Configuración básica del archivo de ensamblado (*AssemblyInfo.cs*) con recursos incrustados (imágenes, iconos, archivos *css*, archivos *javascript*).

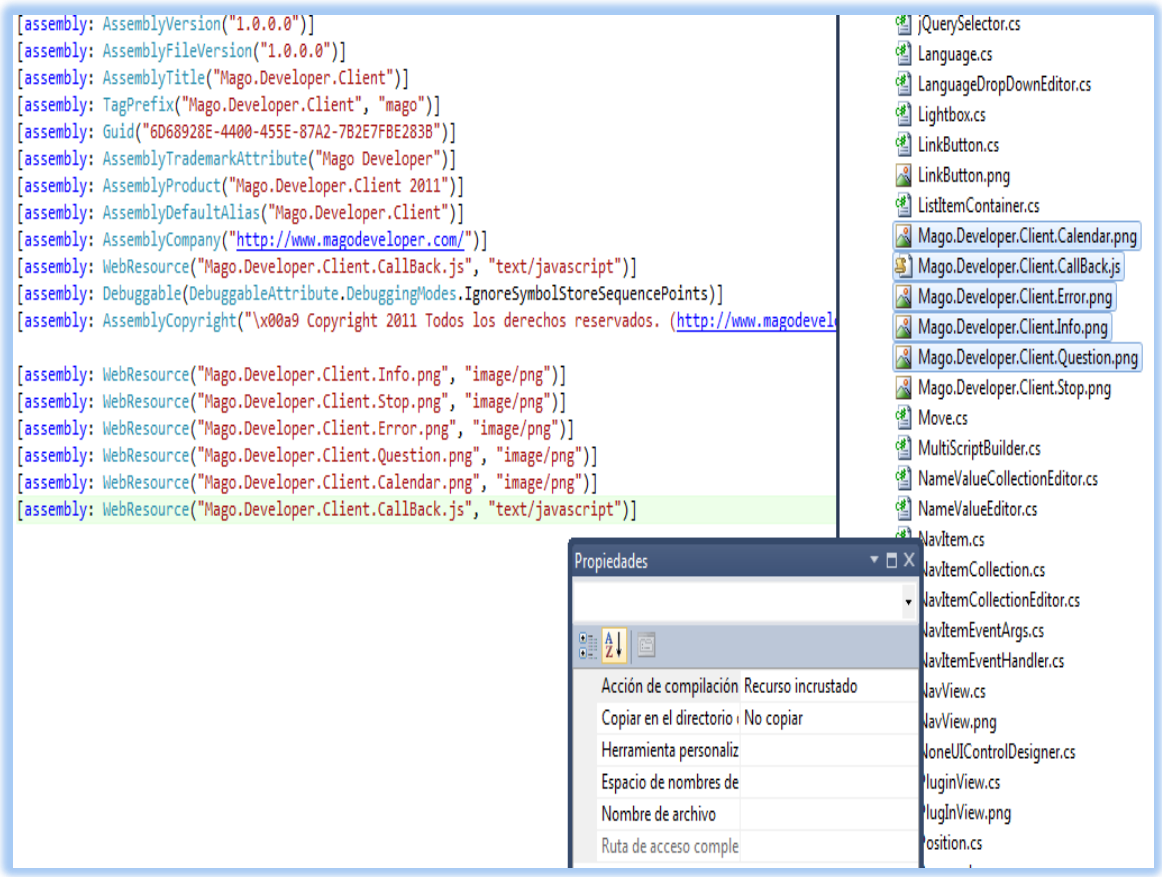

Figura 59. Agregando recursos incrustados en un ensamblado.

1.17.5 Ejemplo de un archivo de código fuente, esta es una clase base para agregar y quitar dinámicamente elementos de una colección en tiempo de diseño para un componente asociado.

```
/*
   Ω_____________________________________Ð
    |                                     |
    |    http://www.magodeveloper.com/    |
    |                                     |
    |  © Todos los derechos reservados.   |
    |_____________________________________|
*/

using System;
using System.Text;
using System.Web.UI;
using System.Collections;

namespace Mago.Developer.Client
{
    /// <summary>
    ///
    /// </summary>
    /// <typeparam name="T"></typeparam>
    public class NameValueCollectionEditor<T> : CollectionBase, IStateManager,
INameValueCollectionEditor<T> where T : NameValueEditor
    {
        private bool isTrackingViewState;
        /// <summary>
        ///
        /// </summary>
        /// <param name="item"></param>
        public void Add(T item)
        {
            if (!Contains(item))
                InnerList.Add(item);
            else
                throw new Exception("El item especificado ya existe.");
```

```
    if (!(this as IStateManager).IsTrackingViewState)
        return;
    (item as IStateManager).TrackViewState();
    item.SetViewStateDirty();
}
/// <summary>
///
/// </summary>
/// <param name="name"></param>
/// <param name="value"></param>
public void Add(string name, string value)
{
    T instance = Activator.CreateInstance<T>();
    instance.Name = name;
    instance.Value = value;
    Add(instance);
}
/// <summary>
///
/// </summary>
/// <param name="name"></param>
/// <returns></returns>
public bool Contains(string name)
{
    return this[name] != null;
}
/// <summary>
///
/// </summary>
/// <param name="item"></param>
/// <returns></returns>
public bool Contains(T item)
{
   return InnerList.Contains(item);
}
```

```
        /// <summary>
        ///
        /// </summary>
        /// <returns></returns>
        public string ToJSONOfString()
        {
            StringBuilder sb = new StringBuilder();
            foreach (T item in InnerList)
            {
                if (sb.Length > 0)
                    sb.Append(",");
                sb.Append(String.Format("{0}:\"{1}\"", item.Name, item.Value));
            }
            return sb.Length > 0 ? String.Format("{{{0}}}", sb) : "{}";
        }
        /// <summary>
        ///
        /// </summary>
        public T this[string name]
        {
            get
            {
                foreach (T item in InnerList)
                {

                    if (String.Compare(item.Name.ToLower(), name.ToLower(),
false) == 0)
                        return item;
                }

                return null;
            }
        }
```

```
/// <summary>
///
/// </summary>
public T this[int index]
{
    get
    {
        return InnerList[index] as T;
    }
}
/// <summary>
///
/// </summary>
/// <param name="item"></param>
public void Remove(T item)
{
    InnerList.Remove(item);
}
/// <summary>
///
/// </summary>
bool IStateManager.IsTrackingViewState
{
    get
    {
        return isTrackingViewState;
    }
}
/// <summary>
///
/// </summary>
/// <param name="state"></param>
void IStateManager.LoadViewState(object state)
{
    object[] bags = state as object[];
```

```
            foreach (object bag in bags)
            {
                if (bag != null)
                {
                    T item = Activator.CreateInstance<T>();
                    (item as IStateManager).LoadViewState(bag);
                    Add(item);
                }
            }
        }
        /// <summary>
        ///
        /// </summary>
        /// <returns></returns>
        object IStateManager.SaveViewState()
        {
            object[] bags = new object[Count];
            int index = 0;
            foreach (NameValueEditor item in InnerList)
                bags[index++] = (item as IStateManager).SaveViewState();
            return bags;
        }
        /// <summary>
        ///
        /// </summary>
        void IStateManager.TrackViewState()
        {
            isTrackingViewState = true;
            foreach (NameValueEditor item in InnerList)
                (item as IStateManager).TrackViewState();
        }
    }
}
```

1.17.6 Establecer atributos a una propiedad a implementarse y visualizarse en el Toolbox de *ASP.NET*.

```
using System;
using System.Web;
using System.Web.UI;
using System.Drawing;
using System.ComponentModel;
using System.Drawing.Design;
using System.Security.Permissions;
using System.ComponentModel.Design;
using Mago.Developer.Client.Design;
using System.Collections.Specialized;

public class AccordionView : JQueryMultiViewControl, IAccordionView
    {
        /// <summary>
        ///
        /// </summary>
        [Bindable(true)]
        [Themeable(true)]
        [Category(Language.Behavior)]
        [Description(Language.accordionViewAutoSizeMode)]
        public AccordionSizeModes AutoSizeMode
        {
            get
            {
                Object obj = ViewState["AutoSizeMode"];
                return (obj == null) ? AccordionSizeModes.None :
(AccordionSizeModes)obj;
            }
            set
            {
                ViewState["AutoSizeMode"] = value;
            }
        }
     }
```

Atributos de una propiedad.

Descriptores de acceso de propiedad.

1.17.7 Establecer atributos a un control (*clase*) a implementarse y visualizarse en el Toolbox de *ASP.NET*.

```
/*
    Ω_____________________________________Ð
    |                                     |
    |    http://www.magodeveloper.com/    |
    |                                     |
    |  © Todos los derechos reservados.   |
    |_____________________________________|
*/

using System;
using System.Web;
using System.Web.UI;
using System.Drawing;
using System.ComponentModel;
using System.Drawing.Design;
using System.Security.Permissions;
using System.ComponentModel.Design;
using Mago.Developer.Client.Design;
using System.Collections.Specialized;

namespace Mago.Developer.Client
{
    /// <summary>
    ///
    /// </summary>
    [ParseChildren(true)]
    [Designer(typeof(AccordionViewDesigner))]
    [ToolboxBitmap(typeof(AccordionView), "AccordionView.png")]
    [ToolboxData("<{0}:AccordionView runat=\"server\"></{0}:AccordionView>")]
    [AspNetHostingPermission(SecurityAction.Demand, Level =
AspNetHostingPermissionLevel.Minimal)]
    [AspNetHostingPermission(SecurityAction.InheritanceDemand, Level =
AspNetHostingPermissionLevel.Minimal)]
    [AttibuteJQueryClass(NameWebResource = "accordion", NameAssembly =
"jQuery", DisposeMethod = "destroy", ScriptWebResources = new string[] {
"ui.core.js", "ui.widget.js", "effects.core.js", "ui.accordion.js",
"plugins.easing.1.3.js" }, NameEvent = ClientRegisterEvents.ApplicationLoad)]
    public class AccordionView : JQueryMultiViewControl, IAccordionView
    {
    }
```

Atributos de un control.

1.17.8 Compilar un proyecto de biblioteca de clases desde línea de comandos.

- **Compilador de línea de comandos lenguaje *C#***
  - <Win>\Microsoft.NET\Framework\<version>\csc.exe.
  - Csc /out:XX /target:YY “Archivo1.cs” “Archivo2.cs”.
- **Compilador de línea de comandos lenguaje *VB.NET***
  - <Win>\Microsoft.NET\Framework\<version>\vbc.exe.
  - vbc /out:XX /target:YY “Archivo1.vb” “Archivo2.vb”.

1.17.9 Algunas opciones útiles:

✓ **/out:<file>**	Nombre del archivo de salida.
✓ **/target:**exe/winexe/***library***	Consola/Windows/***dll***.
✓ **/reference:**<file list>	*Assemblies* de referencia.
✓ **/doc:**<file>	Archivo de documentación.
✓ **/debug**[+|-]	Emitir info de *DEBUG*.
✓ **/main:**<type>	Determina la clase que posee el *Entry Point* (ignora los otros posibles).
✓ **/lib:**<file list>	Directorios de librerías.

1.17.10 (*Recomendado):* Compilar un proyecto de biblioteca de clases desde línea de comandos para generar el archivo de biblioteca y el archivo de documentación *XML*.

```
@echo off
setlocal
set ASSEMBLY_NAME=Mago.Developer.Client
if exist %SYSTEMROOT%\Microsoft.NET\Framework\v4.0.30319 goto net40
if exist %SYSTEMROOT%\Microsoft.NET\Framework\v3.5 goto net35
if exist %SYSTEMROOT%\Microsoft.NET\Framework\v3.0 goto net30
if exist %SYSTEMROOT%\Microsoft.NET\Framework\v2.0.50727 goto net20
if exist %SYSTEMROOT%\Microsoft.NET\Framework\v1.1.4322 goto net11
if exist %SYSTEMROOT%\Microsoft.NET\Framework\v1.0.3705 goto net10
echo Error: .NET Framework v4.0.30319 ó v3.5 ó v3.0 ó v2.0.50727 ó v1.1.4322 ó
v1.0.3705 es requerida.
echo.
goto end
:net10
set CSC_PATH=%SYSTEMROOT%\Microsoft.NET\Framework\v1.0.3705
goto options
:net11
set CSC_PATH=%SYSTEMROOT%\Microsoft.NET\Framework\v1.1.4322
goto options
:net20
set CSC_PATH=%SYSTEMROOT%\Microsoft.NET\Framework\v2.0.50727
goto options
:net30
set CSC_PATH=%SYSTEMROOT%\Microsoft.NET\Framework\v3.0
goto options
```

```
:net35
set CSC_PATH=%SYSTEMROOT%\Microsoft.NET\Framework\v3.5
goto options
:net40
set CSC_PATH=%SYSTEMROOT%\Microsoft.NET\Framework\v4.0.30319
goto options
:options
if %1x == -debugx goto debug
goto release
:release
set OUTPATH=dll
goto compile
:debug
set OUTPATH=Debug
set DEBUG="/debug"
goto compile
:compile
@echo Compilando %ASSEMBLY_NAME%.dll Espere...
@echo.
if exist D:\%OUTPATH%\%ASSEMBLY_NAME%.* del
D:\%OUTPATH%\%ASSEMBLY_NAME%.* /Q
%CSC_PATH%\csc %DEBUG% /target:library /debug-
/doc:D:\%OUTPATH%\%ASSEMBLY_NAME%.xml
/out:D:\%OUTPATH%\%ASSEMBLY_NAME%.dll *.cs
/resource:Mago.Developer.Client.CallBack.js
/resource: Mago.Developer.Client.AccordionView.png
/resource: Mago.Developer.Client.Animation.png
/resource: Mago.Developer.Client.AutoComplete.png
/resource: Mago.Developer.Client.ButtonExtended.png
```

/resource: Mago.Developer.Client.CalendarExtended.png

/resource: Mago.Developer.Client.CheckExtended.png

/resource: Mago.Developer.Client.CollapsablePanel.png

/resource: Mago.Developer.Client.Comboview.png

/resource: Mago.Developer.Client.DialogBox.png

/resource: Mago.Developer.Client.DragView.png

/resource: Mago.Developer.Client.DropView.png

/resource: Mago.Developer.Client.EffectView.png

/resource: Mago.Developer.Client.LinkButton.png

/resource: Mago.Developer.Client.NavView.png

/resource: Mago.Developer.Client.PlugInView.png

/resource: Mago.Developer.Client.Progressbar.png

/resource: Mago.Developer.Client.ResizeView.png

/resource: Mago.Developer.Client.SelectView.png

/resource: Mago.Developer.Client.SliderView.png

/resource: Mago.Developer.Client.SortView.png

/resource: Mago.Developer.Client.TabView.png

/resource: Mago.Developer.Client.View.png

/reference: D:\%OUTPATH%\Mago.Developer.Resources.dll

echo.

echo %ASSEMBLY_NAME%.dll se ha compilado exitosamente!!!.

:end

pause

endlocal

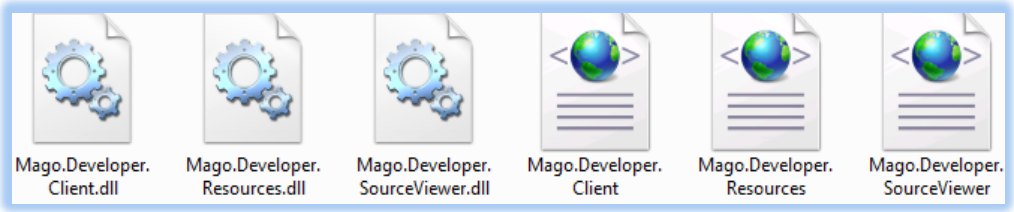

Figura 60. Generando el archivo de biblioteca y de documentación *XML*.

1.18 Enfoque Caso Aplicativo: Sistema de Gestión de Recursos Humanos *UNACH*.

1.18.1 Esquema seguido durante la etapa de análisis y toma de requerimientos previos.

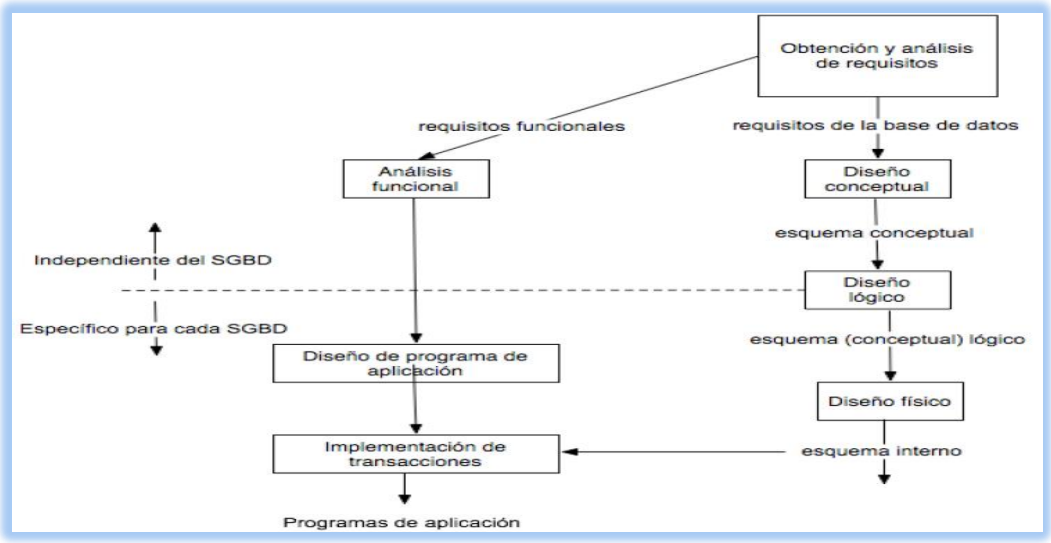

Figura 61. Esquema etapa de análisis y toma de requerimientos previos.

1.18.2 Diagrama Entidad-Relación

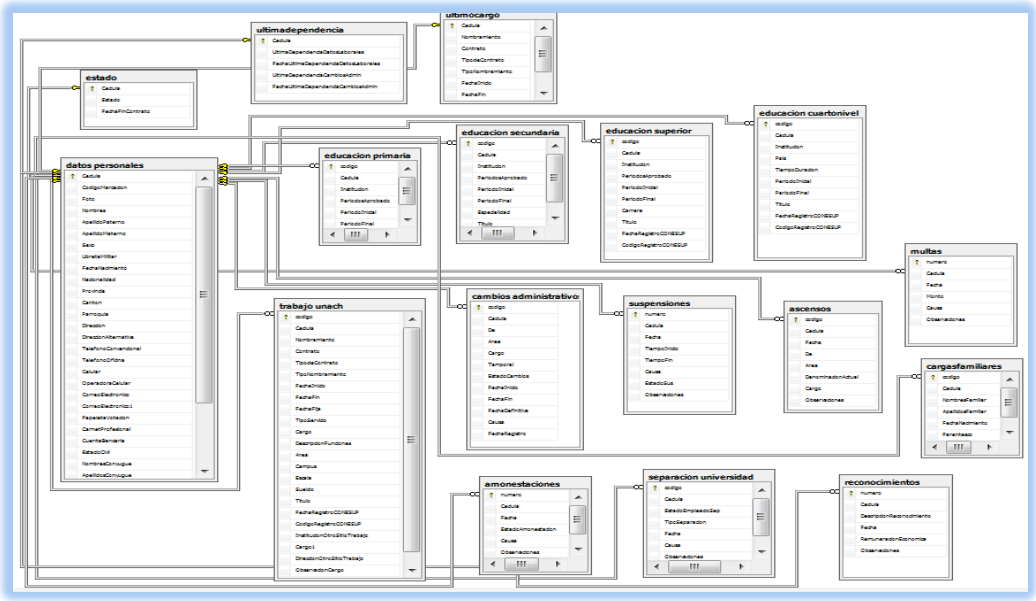

Figura 62. Diagrama Entidad-Relación.

1.18.3 Arquitectura de la Aplicación

✔ **Aplicación Web en 3 capas con *ASP.NET*.**

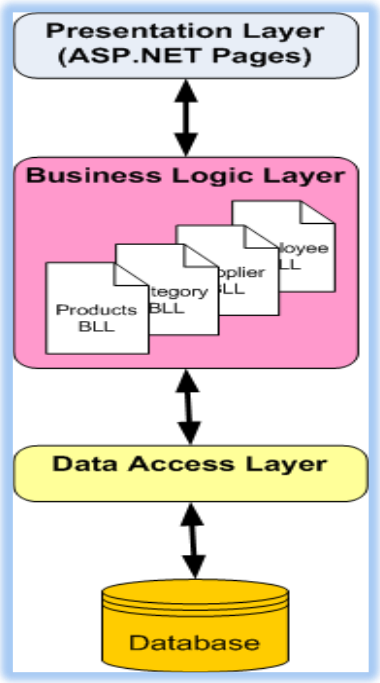

Figura 63. Modelo de una aplicación web en 3 capas con *ASP.NET*.

1.18.4 Introducción a la creación de una aplicación web en 3 capas con *ASP.NET*.

Una aplicación web en *ASP.NET* con un diseño en tres capas se basa en crear una *capa de acceso a datos* (Data Access Layer), una *capa de lógica de negocios* (Business Logic Layer) y una *capa de interface de usuario* (*Master Pages*).

1.18.5 Pasos básicos para la creación de una aplicación web en 3 capas con *ASP.NET*.

1. En el entorno de desarrollo de Microsoft Visual Studio, se deberá escoger Archivo y a continuación Nuevo, seguidamente se seleccionará Sitio Web.

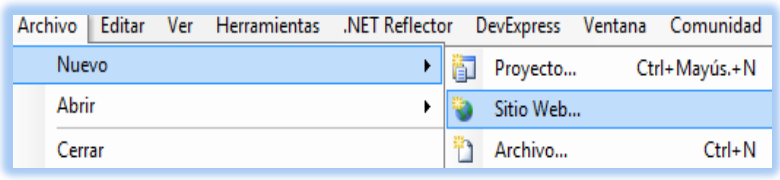

Figura 64. Creación de una aplicación web en 3 capas con *ASP.NET: Paso 1.*

2. Se deberá definir la ubicación o el tipo de alojamiento para los archivos de aplicación, en este caso se seleccionará que la aplicación estará alojada en el directorio virtual (*C:/inetpub/wwwroot/)* en el caso de que el servidor web sea *IIS* (Internet Information Services), o sí se utiliza Abys Web Server la aplicación estaría alojada en el directorio virtual (*C:/Abyss Web Server/htdocs/*).

3. También se deberá definir el tipo de lenguaje de programación a utilizar, y para ello se ha seleccionado el lenguaje Visual C#.

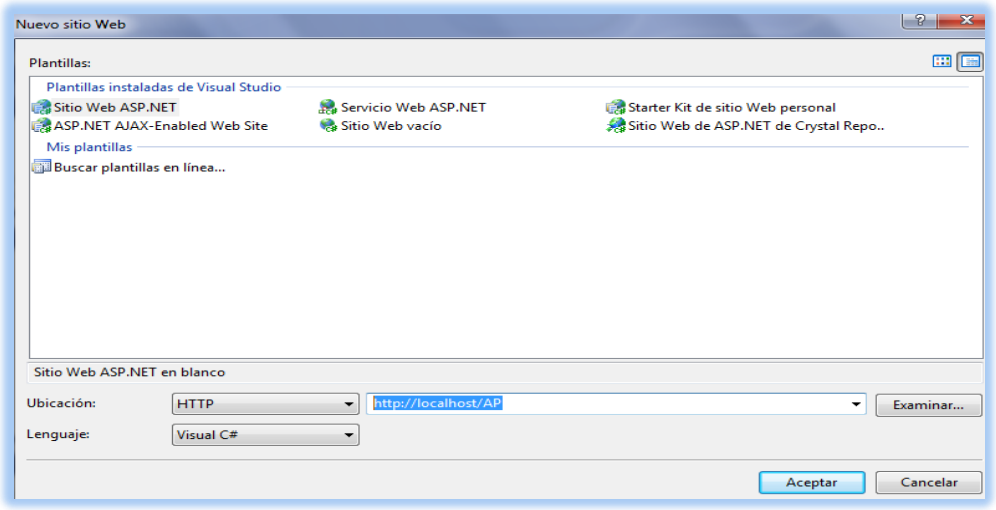

Figura 65. Creación de una aplicación web en 3 capas con *ASP.NET: Paso 2.*

1.18.6 Creación de la capa de acceso a datos en *ASP.NET*

(Data Access Layer) – (*DAL*)

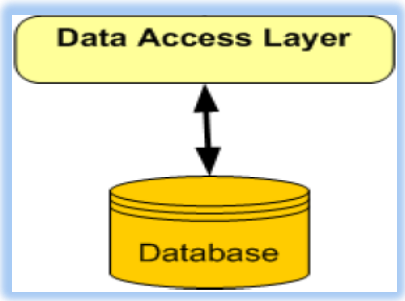

Figura 66. Creación de la capa de acceso a datos en *ASP.NET*.

Primeramente se establece la conexión al gestor de base de datos SQL Server.

Nota: Todas las cadenas de conexión que se utilicen se deben definir en un archivo de configuración *Machine.config* o en un archivo de configuración *Web.config*, en este caso se va a definir todas las cadenas de conexión y demás configuraciones de aplicación en un archivo denominado *Web.config***.**

En la capa de acceso a datos se encuentran todas las referencias a la base de datos, los procedimientos almacenados o los métodos (que pueden devolver un *DataSet* o un *DataReader*).

Además de los *DataTable* también se incluyen *TableAdapters*, que son clases con métodos para rellenar el *DataSet* y *DataTable*.

1.18.7 Metodologías que se pueden utilizar para las transacciones de datos.

1. Cada método, es decir insertar, actualizar y eliminar realizan una solicitud individual que será enviada de inmediato a la base de datos.

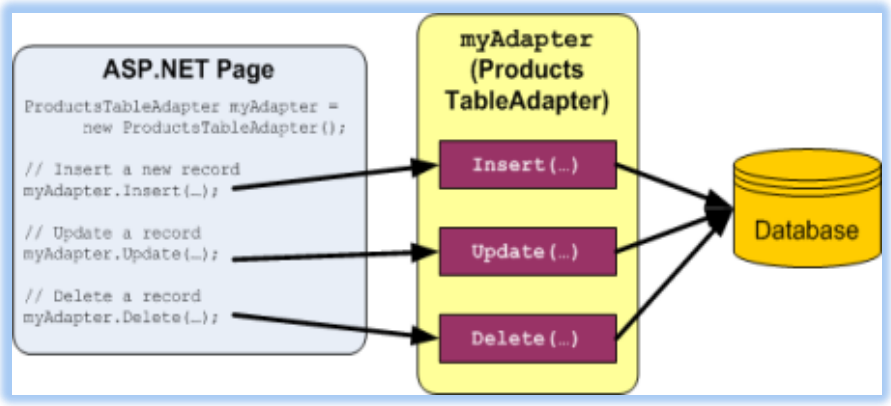

Figura 67. Primera metodología de envío de transacciones a la base de datos.

2. Todos los cambios se sincronizan con la base de datos a través de un único método de actualización que es invocado.

Figura 68. Segunda metodología de envío de transacciones a la base de datos.

1.18.8 Ejemplo (*DataSet* y *DataTable*) del módulo: Datos Personales

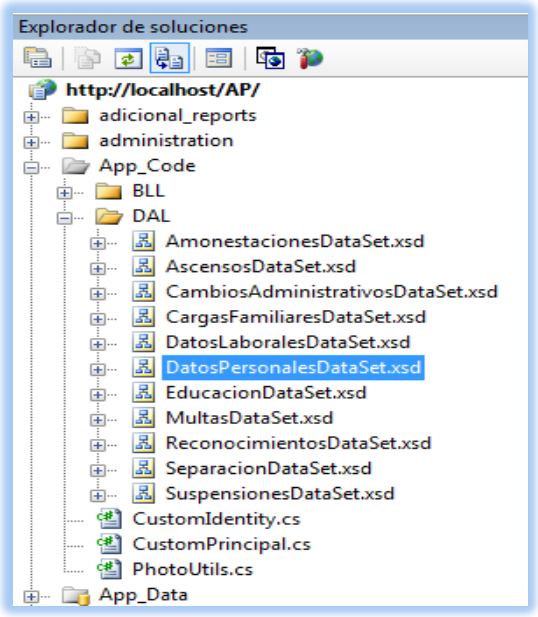

Figura 69. *DataSet* del módulo: Datos Personales.

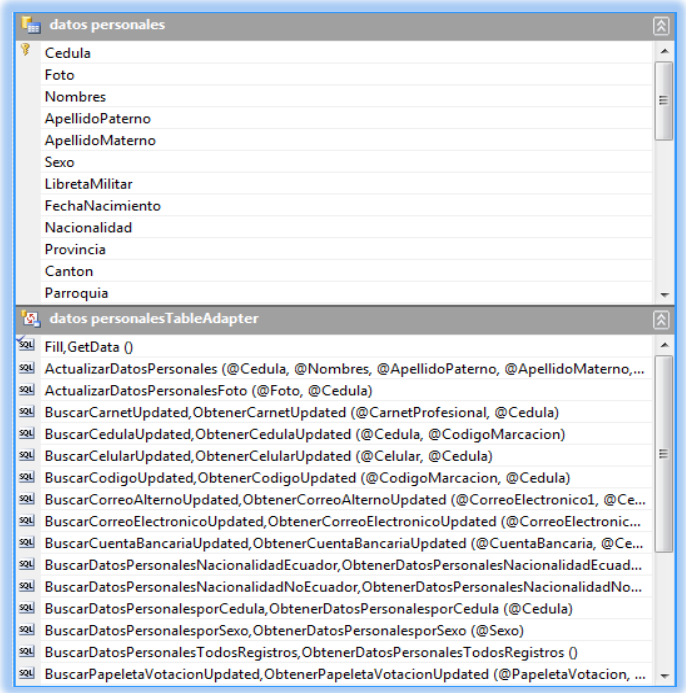

Figura 70. *DataTable* del módulo: Datos Personales.

1.18.9 Creación de la capa de lógica de negocios en *ASP.NET*

(Business Logic Layer) – (*BLL*)

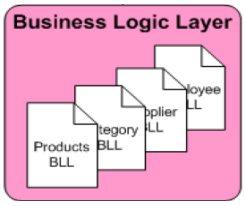

Figura 71. Creación de la capa de lógica de negocios en *ASP.NET*.

En este apartado se enfocará como centralizar las reglas de negocio mediante Business Logic Layer (*BLL*) que sirve como intermediario para el intercambio de datos entre la capa de presentación y la *DAL*.

En un mundo real de aplicación, las *BLL* pueden ser aplicadas como un proyecto de biblioteca de clases, o también como una serie de clases alojadas en la carpeta *App_Code*.

1.18.10 Creación de las *BLL* Clases

La aplicación se compone de una *BLL* por cada *TableAdapter* en la *DAL*, cada una de estas clases tendrá métodos de recuperación, inserción, actualización y supresión de los respectivos *TableAdapter* en la *DAL* si es que así lo amerita.

Para separar más limpiamente la lógica *DAL-BLL* y relacionarlas con las clases, se va a crear dos subcarpetas en la carpeta *App_Code*, *DAL* y *BLL*.

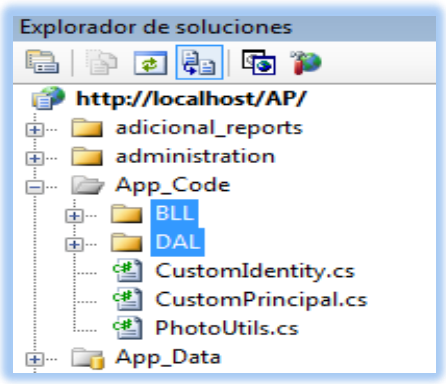

Figura 72. Carpeta *App_Code* y su relación con *DAL-BLL*.

1.18.11 Ejemplo de una clase de lógica de negocios: módulo "Datos Personales".

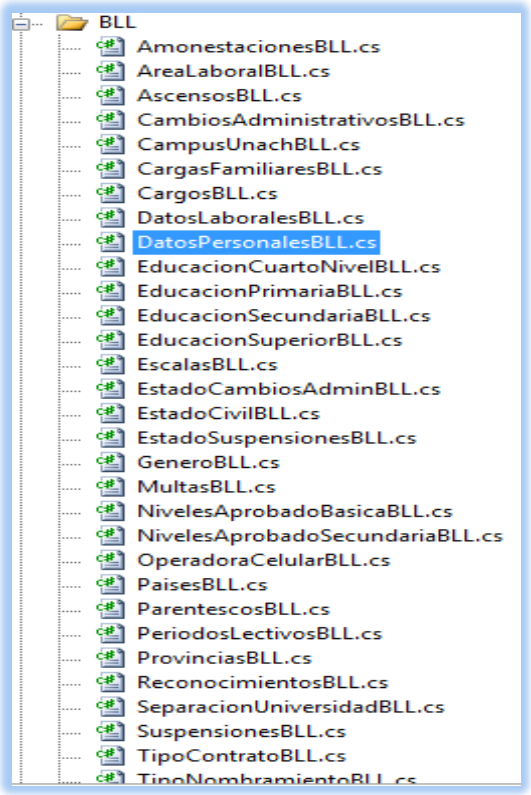

Figura 73. Clase de lógica de negocios: módulo "Datos Personales".

```
using System;
using DatosPersonalesDataSetTableAdapters;

  [System.ComponentModel.DataObject]
public class DatosPersonalesBLL
{
    private datos_personalesTableAdapter AdaptadorDatosPersonales;
    protected datos_personalesTableAdapter Adapter
    {
        get
        {
            if (AdaptadorDatosPersonales == null)
                AdaptadorDatosPersonales = new datos_personalesTableAdapter();

            return AdaptadorDatosPersonales;
        }
    }

    [System.ComponentModel.DataObjectMethodAttribute(System.ComponentModel.DataObjectMethodType.Select, false)]
    public DatosPersonalesDataSet.datos_personalesDataTable ObtenerDatosPersonalesTodosRegistros()
    {
        return Adapter.ObtenerDatosPersonalesTodosRegistros();
    }

    [System.ComponentModel.DataObjectMethodAttribute(System.ComponentModel.DataObjectMethodType.Select, false)]
    public DatosPersonalesDataSet.datos_personalesDataTable ObtenerDatosPersonalesNacionalidadEcuador()
    {
        return Adapter.ObtenerDatosPersonalesNacionalidadEcuador();
    }

[System.ComponentModel.DataObjectMethodAttribute(System.ComponentModel.DataObjectMethodType.Select, false)]
public DatosPersonalesDataSet.datos_personalesDataTable ObtenerDatosPersonalesNacionalidadNoEcuador()
{
    return Adapter.ObtenerDatosPersonalesNacionalidadNoEcuador();
}

[System.ComponentModel.DataObjectMethodAttribute(System.ComponentModel.DataObjectMethodType.Select, false)]
public DatosPersonalesDataSet.datos_personalesDataTable ObtenerDatosPersonalesporCedula(string Cedula)
{
    return Adapter.ObtenerDatosPersonalesporCedula(Cedula);
}

[System.ComponentModel.DataObjectMethodAttribute(System.ComponentModel.DataObjectMethodType.Select, false)]
public DatosPersonalesDataSet.datos_personalesDataTable ObtenerDatosPersonalesporSexo(string Sexo)
{
    return Adapter.ObtenerDatosPersonalesporSexo(Sexo);
}
[System.ComponentModel.DataObjectMethodAttribute(System.ComponentModel.DataObjectMethodType.Delete, true)]
public bool EliminacionGlobaldeDatos(string Cedula)
{
    try
    {
        int rowsAffected = Adapter.EliminacionGlobaldeDatos(Cedula);

        return rowsAffected == 1;
    }
    catch
    {
        throw;
    }
    finally
    {
        Adapter.Dispose();
    }
}
```

```
[System.ComponentModel.DataObjectMethodAttribute(System.ComponentModel.DataObjectMethodType.Insert, true)]
public bool RegistrarDatosPersonales(string Cedula, string Nombres, string ApellidoPaterno, string ApellidoMaterno, string Sexo, string Lil
{
    DatosPersonalesDataSet.datos_personalesDataTable tempsearch = Adapter.ObtenerDatosPersonalesporCedula(Cedula);
    DatosPersonalesDataSet.datos_personalesDataTable CodigoMarcacion_ = Adapter.ObtenerporCodigodeMarcacion(CodigoMarcacion);
    DatosPersonalesDataSet.datos_personalesDataTable Celular_ = Adapter.ObtenerporCelular(Celular);
    DatosPersonalesDataSet.datos_personalesDataTable CarnetProfesional_ = Adapter.ObtenerporCarnetProfesional(CarnetProfesional);
    DatosPersonalesDataSet.datos_personalesDataTable PapeletaVotacion_ = Adapter.ObtenerporPapeletadeVotacion(PapeletaVotacion);
    DatosPersonalesDataSet.datos_personalesDataTable CuentaBancaria_ = Adapter.ObtenerporCuentaBancaria(CuentaBancaria);
    DatosPersonalesDataSet.datos_personalesDataTable CorreoElectronico_ = Adapter.ObtenerporCorreoElectronico(CorreoElectronico);
    DatosPersonalesDataSet.datos_personalesDataTable CorreoElectronicoAlterno_ = Adapter.ObtenerporCorreoElectronicoAlterno(CorreoElectron:

    try
    {
        if (tempsearch.Rows.Count == 1)
        {
            throw new ArgumentException("El empleado seleccionado ya tiene registrado sus Datos Personales. Los Datos Personales son únicos
        }
        if (CodigoMarcacion_.Rows.Count == 1)
        {
            throw new ArgumentException("El código de marcación es único, este código ya se encuentra registrado.");
        }
        if (Celular_.Rows.Count == 1)
        {
            throw new ArgumentException("El número de celular es único, este número ya se encuentra registrado.");
        }
        if (CarnetProfesional_.Rows.Count == 1)
        {
            throw new ArgumentException("El carné profesional es único, este carné ya se encuentra registrado.");
        }
[System.ComponentModel.DataObjectMethodAttribute(System.ComponentModel.DataObjectMethodType.Insert, true)]
public bool RegistrarDatosPersonales(string Cedula, string Nombres, string ApellidoPaterno, string ApellidoMaterno, string Sexo, string Lil
{
    DatosPersonalesDataSet.datos_personalesDataTable tempsearch = Adapter.ObtenerDatosPersonalesporCedula(Cedula);
    DatosPersonalesDataSet.datos_personalesDataTable CodigoMarcacion_ = Adapter.ObtenerporCodigodeMarcacion(CodigoMarcacion);
    DatosPersonalesDataSet.datos_personalesDataTable Celular_ = Adapter.ObtenerporCelular(Celular);
    DatosPersonalesDataSet.datos_personalesDataTable CarnetProfesional_ = Adapter.ObtenerporCarnetProfesional(CarnetProfesional);
    DatosPersonalesDataSet.datos_personalesDataTable PapeletaVotacion_ = Adapter.ObtenerporPapeletadeVotacion(PapeletaVotacion);
    DatosPersonalesDataSet.datos_personalesDataTable CuentaBancaria_ = Adapter.ObtenerporCuentaBancaria(CuentaBancaria);
    DatosPersonalesDataSet.datos_personalesDataTable CorreoElectronico_ = Adapter.ObtenerporCorreoElectronico(CorreoElectronico);
    DatosPersonalesDataSet.datos_personalesDataTable CorreoElectronicoAlterno_ = Adapter.ObtenerporCorreoElectronicoAlterno(CorreoElectron:

    try
    {
        if (tempsearch.Rows.Count == 1)
        {
            throw new ArgumentException("El empleado seleccionado ya tiene registrado sus Datos Personales. Los Datos Personales son únicos
        }
        if (CodigoMarcacion_.Rows.Count == 1)
        {
            throw new ArgumentException("El código de marcación es único, este código ya se encuentra registrado.");
        }
        if (Celular_.Rows.Count == 1)
        {
            throw new ArgumentException("El número de celular es único, este número ya se encuentra registrado.");
        }
        if (CarnetProfesional_.Rows.Count == 1)
        {
            throw new ArgumentException("El carné profesional es único, este carné ya se encuentra registrado.");
        }
```

```
        if (PapeletaVotacion_.Rows.Count == 1)
        {
            throw new ArgumentException("El comprobante de votación es único, este comprobante ya se encuentra registrado.");
        }
        if (CuentaBancaria_.Rows.Count == 1)
        {
            throw new ArgumentException("El número de cuenta bancaria es único, este número ya se encuentra registrado.");
        }
        if (CorreoElectronico_.Rows.Count == 1)
        {
            throw new ArgumentException("La dirección de correo electrónico es única, esta dirección ya se encuentra registrada.");
        }
        if ((CorreoElectronicoAlterno_.Rows.Count == 1) && (CorreoElectronico != CorreoElectronico1))
        {
            throw new ArgumentException("La dirección de correo electrónico es única, esta dirección ya se encuentra registrada por otro
        }

        int rowsAffected = Adapter.RegistrarDatosPersonales(Cedula, Nombres, ApellidoPaterno, ApellidoMaterno, Sexo, LibretaMilitar, Fech

        return rowsAffected == 1;
    }
    catch
    {
        throw;
    }
    finally
    {
        Adapter.Dispose();
    }

}
 [System.ComponentModel.DataObjectMethodAttribute(System.ComponentModel.DataObjectMethodType.Update, true)]
 public bool ActualizarDatosPersonales(string Cedula, string Nombres, string ApellidoPaterno, string ApellidoMaterno, string Sexo, string
 {
     DatosPersonalesDataSet.datos_personalesDataTable tempsearch = Adapter.ObtenerCedulaUpdated(Cedula, CodigoMarcacion);
     DatosPersonalesDataSet.datos_personalesDataTable CodigoMarcacion_ = Adapter.ObtenerCodigoUpdated(CodigoMarcacion, Cedula);
     DatosPersonalesDataSet.datos_personalesDataTable Celular_ = Adapter.ObtenerCelularUpdated(Celular, Cedula);
     DatosPersonalesDataSet.datos_personalesDataTable CarnetProfesional_ = Adapter.ObtenerCarnetUpdated(CarnetProfesional, Cedula);
     DatosPersonalesDataSet.datos_personalesDataTable PapeletaVotacion_ = Adapter.ObtenerPapeletaVotacionUpdated(PapeletaVotacion, Cedula)
     DatosPersonalesDataSet.datos_personalesDataTable CuentaBancaria_ = Adapter.ObtenerCuentaBancariaUpdated(CuentaBancaria, Cedula);
     DatosPersonalesDataSet.datos_personalesDataTable CorreoElectronico_ = Adapter.ObtenerCorreoElectronicoUpdated(CorreoElectronico, Cedu
     DatosPersonalesDataSet.datos_personalesDataTable CorreoElectronicoAlterno_ = Adapter.ObtenerCorreoAlternoUpdated(CorreoElectronico1,

     try
     {
         if (tempsearch.Rows.Count == 0)
         {
             throw new ArgumentException("La cédula de identidad es única, esta cédula ya se encuentra registrada.");
         }
         if (CodigoMarcacion_.Rows.Count == 1)
         {
             throw new ArgumentException("El código de marcación es único, este código ya se encuentra registrado.");
         }
         if (Celular_.Rows.Count == 1)
         {
             throw new ArgumentException("El número de celular es único, este número ya se encuentra registrado.");
         }
         if (CarnetProfesional_.Rows.Count == 1)
         {
             throw new ArgumentException("El carné profesional es único, este carné ya se encuentra registrado.");
         }
```

```
        if (PapeletaVotacion_.Rows.Count == 1)
        {
            throw new ArgumentException("El comprobante de votación es único, este comprobante ya se encuentra registrado.");
        }
        if (CuentaBancaria_.Rows.Count == 1)
        {
            throw new ArgumentException("El número de cuenta bancaria es único, este número ya se encuentra registrado.");
        }
        if (CorreoElectronico_.Rows.Count == 1)
        {
            throw new ArgumentException("La dirección de correo electrónico es única, esta dirección ya se encuentra registrada.");
        }
        if ((CorreoElectronicoAlterno_.Rows.Count == 1) && (CorreoElectronico != CorreoElectronico1))
        {
            throw new ArgumentException("La dirección de correo electrónico es única, esta dirección ya se encuentra registrada por otro f
        }

        int rowsAffected = Adapter.ActualizarDatosPersonales(Cedula, Nombres, ApellidoPaterno, ApellidoMaterno, Sexo, LibretaMilitar, Fech

        return rowsAffected == 1;
    }
    catch
    {
        throw;
    }
    finally
    {
        Adapter.Dispose();
    }
}
     [System.ComponentModel.DataObjectMethodAttribute(System.ComponentModel.DataObjectMethodType.Update, false)]
     public bool ActualizarDatosPersonalesFoto(string Cedula, byte[] Foto)
     {
         try
         {
             int rowsAffected = Adapter.ActualizarDatosPersonalesFoto(Foto, Cedula);

             return rowsAffected == 1;
         }
         catch
         {
             throw;
         }
         finally
         {
             Adapter.Dispose();
         }
     }
}
```

1.18.12 Creación de la capa de presentación o interface de usuario en *ASP.NET*.

ASP.NET a diferencia de otras plataformas de desarrollo de aplicaciones web implementa el concepto de las *Master Pages*, una *Master Page* es muy similar en cuanto a las funcionalidades que proporciona un formulario *MdiContainer* en una aplicación *Windows Forms*, generalmente en una *Master Page* se implementa las funcionalidades de navegación como menú, encabezado de resumen, estado de sesión, barras de herramientas, etc.

Cuando se crea una *Master Page* se agrega por defecto un control denominado *ContentPlaceHolder*, este control es la región editable que se crea para todas las páginas web asociadas a una *Master Page* específica.

Este modelo permite a los desarrolladores definir un sitio y centralizar toda la disposición de la página, haciendo más fácil la creación de un aspecto coherente en todas las páginas que pueden actualizarse fácilmente.

Nota importante: Se pueden crear los *ContentPlaceHolder* deseados, por defecto se agrega uno, definiendo de este modo la disposición de todos los controles web en pantalla incluyendo los controles de tipo *ContentPlaceHolder*, según el diseño requerido para la interface de usuario.

La ubicación en pantalla y todos los atributos de estilo de todos los controles web en la interface de usuario están diseñados completamente aplicando normativas de diseño establecidas por la *W3C*, utilizando tecnologías de desarrollo como *CSS* y *javascript*.

1.18.13 Pasos básicos para la creación de la *Master Page.*

1. En el explorador de soluciones del proyecto se deberá hacer clic derecho y elegir agregar nuevo elemento.

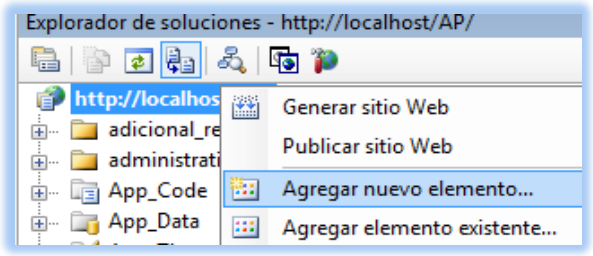

Figura 74. Pasos básicos para la creación de la *Master Page: Paso 1.*

2. El nuevo elemento a seleccionar es uno de tipo página principal.

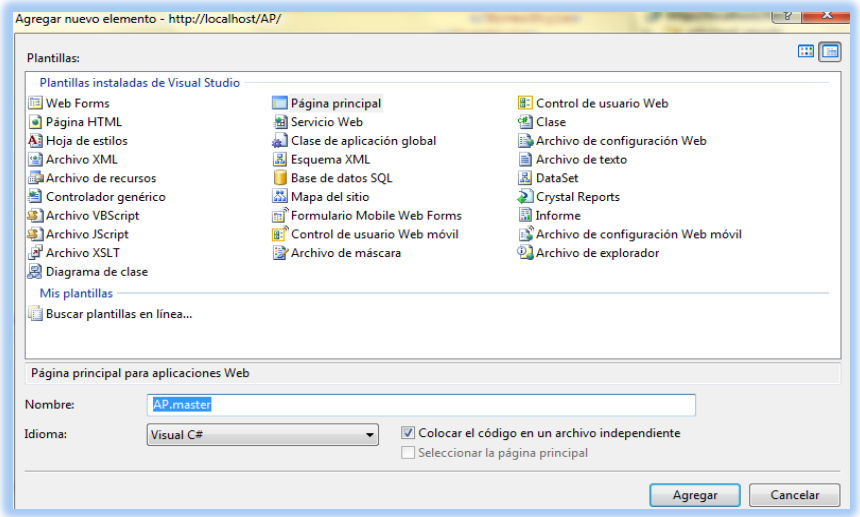

Figura 75. Pasos básicos para la creación de la *Master Page: Paso 2.*

3. Se agregará dinámicamente la página maestra que se acaba de crear en el explorador de soluciones del proyecto.

4. Ahora se mostrará cómo se está diseñando la página maestra y se hace referencia al mencionado *ContentPlaceHolder*.

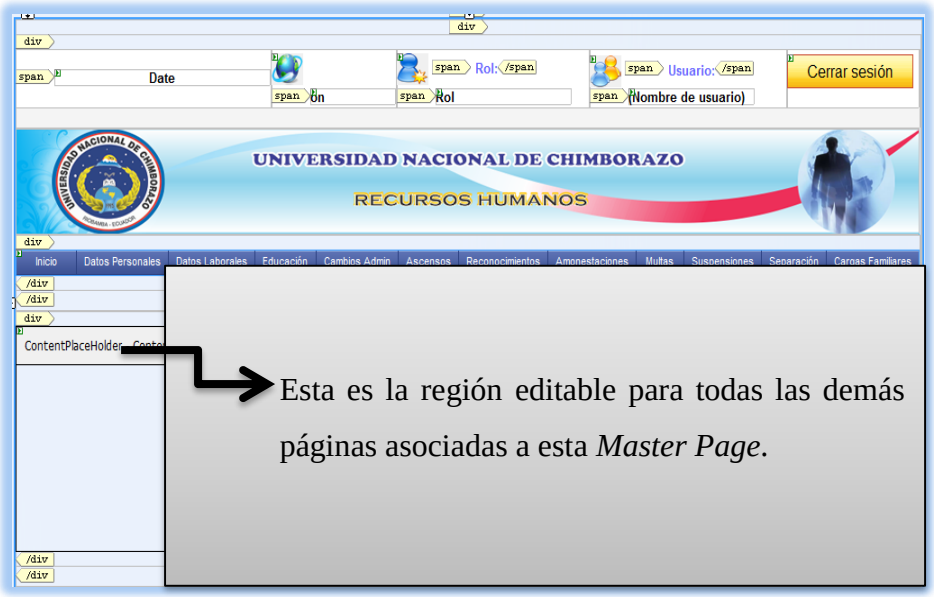

Figura 76. Pasos básicos para la creación de la *Master Page: Paso 3.*

1.18.14 Metodología de desarrollo de software

Se ha seguido la *metodología de desarrollo de software incremental*, que se basa fundamentalmente en proveer una estrategia para controlar la complejidad y los riesgos, desarrollando una parte del producto de software, reservando el resto de aspectos para el futuro, ya que ha sido indispensable hacer un detenido y minucioso análisis en relación a los requerimientos previos enfocando las necesidades actuales y futuras del sistema.

1.18.15 Aplicación de los controles más importantes del Toolbox Mago Developer Component en el Sistema de Gestión de Recursos Humanos de la Universidad Nacional de Chimborazo.

✔ **CallbackPanel**

Habilita cualquier sección de una página web para que se represente parcialmente de forma asíncrona a nivel de aplicación cliente mediante devoluciones de llamadas asíncronas, evitando el típico *postback*.

- ✓ ***Propiedades y eventos disponibles:***

```
<mago:CallbackPanel
  ChildrenAsTriggers="True | False"
  ClientIDMode="Inherit | AutoID | Predictable | Static"
  EnableViewState="True | False"
  ID=""
  OnDataBinding=""
  OnDisposed=""
  OnInit=""
  OnLoad=""
  OnPreRender=""
  OnUnload=""
  RenderMode="Block | Inline"
  UpdateMode="Always | Conditional"
  ViewStateMode="Inherit | Enabled | Disabled"
  Visible="True | False"
  <Triggers>
      <mago:AsyncPostBackTrigger ControlID="" EventName="" />
      <mago:PostBackTrigger ControlID="" />
  </Triggers>
  <ContentTemplate>
  </ContentTemplate>
</mago:CallbackPanel>
```

✔ CallbackProgress

Proporciona comentarios visuales en el explorador cuando se actualiza el contendido representado parcialmente de forma asíncrona a nivel de aplicación cliente mediante devoluciones de llamadas asíncronas, en los controles *CallbackPanel* asociados.

- ✓ ***Propiedades y eventos disponibles:***

```
<mago:CallbackProgress
AssociatedCallbackPanelID=""
ClientIDMode="Inherit | AutoID | Predictable | Static"
DisplayAfter=""
DynamicLayout="True | False"
EnableViewState="True | False"
ID=""
OnDataBinding=""
OnDisposed=""
OnInit=""
OnLoad=""
OnPreRender=""
OnUnload=""
ViewStateMode="Inherit | Enabled | Disabled"
Visible="True | False"
<ProgressTemplate>
</ProgressTemplate>
</mago:CallbackProgress>
```

✔ Autocomplete

Proporciona funcionalidades avanzadas de autocompletar contenidos, mediante sugerencias dinámicas de datos coincidentes encontrados, ya sea en un recurso remoto como una base de datos, un web service, o una colección de datos definidos por el usuario, en un control asociado, ya sea un control `<asp:TextBox></asp:TextBox>` o un control de lista desplegable como `<asp:DropDownList></asp:DropDownList>`, `<mago:Comboview></mago:Comboview>`, etc, y además soporta una colección jerárquica de eventos de aplicación cliente.

- ✓ ***Propiedades y eventos disponibles:***

```
<mago:AutoComplete
runat="server"
ClientIDMode="Inherit | AutoID | Predictable | Static"
CompletionSetCount=""
Delay=""
EnabledCache="True | False"
EnableViewState="True | False"
ID=""
IsMultipleValueSupports="True | False"
MinLength=""
OnClientChange=""
OnClientClose=""
OnClientInputFocus=""
OnClientItemSearch=""
OnClientItemSelect=""
OnClientOpen=""
OnDataBinding=""
OnDisposed=""
OnInit=""
OnLoad=""
OnPreRender=""
OnUnload=""
ServiceMethod=""
```

```
    ServicePath=""
    Source=""
    ViewStateMode="Inherit | Enabled | Disabled"
    Visible="True | False">
    <Target Selector="" TargetID="" TargetIDs="" />
</mago:AutoComplete>
```

✔ DialogBox

Muestra cuadros de diálogo completamente personalizables, elegantes y funcionales, posesionados de forma *modal*, pueden aceptar un mensaje completamente textual o cualquier contenido insertado en la sección `<BodyTemplate></BodyTemplate>` del mismo, soporte de imágenes que acompañan al mensaje presentado, es redimensionable, arrastrable en las diferentes zonas de la pantalla, soporte completo de estilos *CSS* y de su propiedad avanzada *z-index*, soporta la pulsación de la tecla *ESC* para ocultar y cancelar el mismo, *y* finalmente dispone de soporte para una colección jerárquica de eventos de aplicación cliente.

- ✓ ***Propiedades y eventos disponibles:***

```
  <mago:DialogBox
    runat="server"
    AccessKey=""
    AutoOpen="True | False"
    BackColor=""
    BgiFrame="True | False"
    BorderColor=""
    BorderStyle="Dashed | Dotted | Double | Groove | Inset | None | NotSet |
Outset | Ridge | Solid"
    BorderWidth=""
    ClientIDMode="Inherit | AutoID | Predictable | Static"
    CloseOnEscape="True | False"
    CssClass=""
    DialogButtons="Cancel | Close | None | OK"
    DialogIcon="Error | Info | None | Question | Stop"
```

```
    Enabled="True | False"
    EnableTheming="True | False"
    EnableViewState="True | False"
    Font-Bold="True | False"
    Font-Italic="True | False"
    Font-Names=""
    Font-Overline="True | False"
    Font-Size="Large | Larger | Medium | Small | Smaller | X-Large | X-Small |
XX-Large | XX-Small"
    Font-Strikeout="True | False"
    Font-Underline="True | False"
    ForeColor=""
    Height=""
    HideEffect="Blind | Bounce | Clip | Drop | Explode | Fold | Highlight |
None | Pulsate | Scale | Shake | Slide | Transfer"
    IconUrl=""
    ID=""
    IsDraggable="True | False"
    IsMultipleInstance="True | False"
    IsResizable="True | False"
    MaxHeight=""
    MaxWidth=""
    MessageText=""
    MinHeight=""
    MaxWidth=""
    OnButtonCommand=""
    OnClientBeforeClose=""
    OnClientClose=""
    OnClientDrag=""
    OnClientDragStart=""
    OnClientDragStop=""
    OnClientFocus=""
    OnClientOpen=""
    OnClientResize=""
    OnClientResizeStart=""
    OnClientResizeStop=""
```

```
    OnDataBinding=""
    OnDisposed=""
    OnInit=""
    OnLoad=""
    OnPreRender=""
    OnUnload=""
    Position="Bottom | Center | Left | LeftBottom | LeftTop | PositionAbsolute
| Right | RightBottom | RightTop | Top"
    PositionLeft=""
    PositionTop=""
    ShowEffect="Blind | Bounce | Clip | Drop | Explode | Fold | Highlight |
None | Pulsate | Scale | Shake | Slide | Transfer"
    ShowModal="True | False"
    SkinID=""
    TabIndex=""
    Title=""
    ViewStateMode="Inherit | Enabled | Disabled"
    Width=""
    ZIndex="">
    <BodyTemplate>
    </BodyTemplate>
    <Buttons>
        <mago:DialogButton  CommandArgument=""  CommandName=""  OnClientClick=""
Text="" />
    </Buttons>
    <mago:DialogButton   CommandArgument=""   CommandName=""   OnClientClick=""
Text="" />
    <Trigger Selector="" TargetID="" TargetIDs="" />
</mago:DialogBox>
```

✔ SortView

Proporciona funcionalidades avanzadas de ordenar dinámicamente cualquier tipo de elementos, combinando funcionalidades de arrastrar y colocar los elementos en donde desea visualizarlos el usuario, se puede aplicar a cualquier tipo de elementos, pero se utiliza principalmente en controles de tipo `<asp:GridView></asp:GridView>`, `<asp:DetailsView></asp:DetailsView>`, ordenando de esta forma dinámicamente las columnas que contienen los campos enlazados a un recurso remoto, como lo es una base de datos.

- ✓ ***Propiedades y eventos disponibles:***

```
<mago:SortView
    runat="server"
    AllowDropOnEmpty="True | False"
    AutoScroll="True | False"
    ClientIDMode="Inherit | AutoID | Predictable | Static"
    Container="Customize | Document | NotSet | Parent | Window"
    Cursor=""
    DisplaceX=""
    DisplaceY=""
    DragHelper="Clone | Original"
    DragHelperOpacity=""
    DragOrientation="Both | Horizontal | Vertical"
    DragStartDelay=""
    DragStartDistance=""
    EnableViewState="True | False"
    ForceHelperSize="True | False"
    ForcePlaceholderSize="True | False"
    ID=""
    OnClientSort=""
    OnClientSortActivate=""
    OnClientSortBeforeStop=""
    OnClientSortChange=""
```

```
        OnClientSortDeactivate=""
        OnClientSortOut=""
        OnClientSortOver=""
        OnClientSortReceive=""
        OnClientSortRemove=""
        OnClientSortStart=""
        OnClientSortStop=""
        OnClientSortUpdate=""
        OnDataBinding=""
        OnDisposed=""
        OnInit=""
        OnLoad=""
        OnPreRender=""
        OnUnload=""
        PlaceHolderCssClass=""
        Revert="True | False"
        ScrollSensitivity=""
        ScrollSpeed=""
        Tolerance="Intersect | Pointer"
        ViewStateMode="Inherit | Enabled | Disabled"
        ZIndex="">
</mago:SortView>
```

CAPÍTULO II

2 METODOLOGÍA

2.1 Tipo de estudio

Cuasi experimental: Porque se encuentra enfocado directamente a la comunidad de desarrolladores de sistemas informáticos basados en la web, de tal forma que este es el grupo tomado como referente.

2.1.1 Método de investigación:

***Método Deductivo*:** Para la investigación se parte de la documentación integrada en el entorno de desarrollo de Microsoft Visual Studio 2010 Ultimate (*MSDN Library*), que es la base de la cual se parte para el desarrollo del proyecto.

Se toma como referencia esencial el estudio minucioso y detenido de la plataforma de desarrollo Microsoft .NET Framework (la disponibilidad y compatibilidad entre versiones).

2.1.2 Nivel de la investigación

***Exploratorio*:** Dado los múltiples requerimientos existentes en el desarrollo de sistemas informáticos basados en la web y con el referente de que el Toolbox del entorno de desarrollo actual de *ASP.NET* es aún limitado, razón por la cual en base a la investigación se crearán nuevas herramientas que cumplan con criterios de funcionalidad, practicidad, estandarización y usabilidad.

2.1.3 Lugar en el que se realizó la investigación

Instalaciones de la Universidad Nacional de Chimborazo.

2.1.4 Período de duración de la investigación

6 meses de duración (Octubre 2010 – Marzo 2011).

2.2 Población y Muestra

2.2.1 Población:

Comunidad de desarrolladores de sistemas informáticos basados en la web.

2.2.2 Muestra:

Comunidad de desarrolladores de sistemas informáticos basados en la web de la Universidad Nacional de Chimborazo.

2.3 Operacionalización de variables

Variable	Descripción	Indicador
parámetro1	Ordenación, agrupación semántica de código y reutilización de código en forma de objetos.	Funcionalidad. Eficiencia. Reusabilidad.
parámetro2	Manipulación de objetos de servidor mediante eventos basados en performance *callback*.	Disponibilidad. Rendimiento. Tiempos de respuesta.
parámetro 3	Colección jerárquica de scripts y eventos de aplicación cliente basados en *ECMAScript 262 5ta. Edición* y performance *callback*.	Disponibilidad. Rendimiento. Tiempos de respuesta.
parámetro4	Nuevos e innovadores objetos con compatibilidad con los nuevos estándares *CSS3 y HTML5*.	Estandarización. Funcionalidad. Usabilidad.
parámetro5	Compatibilidad con múltiples servicios web y recursos *XML*.	Usabilidad. Disponibilidad.
Parámetro6	Agentes de usuario para visualización de resultados.	Disponibilidad. Compatibilidad.

2.4 Procedimientos

2.4.1 Técnicas de recolección de datos:

Documentales: (*MSDN Library*).

2.4.2 Instrumentos de recolección de datos:

Conexión de red local de internet (*MSDN Library* visualizado en *localhost*).

2.5 Procesamiento y Análisis

2.5.1 Tratamiento de la información

Los resultados obtenidos serán evaluados mediante un análisis minucioso de las limitaciones existentes en el Toolbox de desarrollo de *ASP.NET* integrado por defecto y las soluciones presentadas en el Toolbox Mago Developer Component investigado y desarrollado, en dicho análisis se contemplaran aspectos como nuevos e innovadores objetos, propiedades disponibles, eventos de servidor utilizables para la manipulación de los objetos y la aplicación de performance *callback*, colección jerárquica de eventos enviados a través de aplicaciones cliente y su optimización de rendimiento a través de performance *callback*, aplicación de los nuevos estándares de desarrollo de *GUI* de usuario aplicando las tecnologías *CSS3* y *HTML5* para la obtención de objetos de servidor elegantes, funcionales y optimizados.

2.5.2 Comprobación de Hipótesis

2.5.2.1 Ordenación y agrupación semántica de código, código reutilizable en forma de objetos.

Mediante la creación de las bibliotecas de clases (componentes) del Toolbox Mago Developer Component se agrupa y ordena jerárquicamente de forma lógica cada uno de los objetos creados, que pueden ser reutilizados cuántas veces se desee, el código generado para cada uno de los objetos es transparente para el usuario al estar agrupado en un único archivo de biblioteca (archivo *.dll*) dónde se agrupa toda la colección de clases, recursos y archivos necesario para la creación de dicho componente.

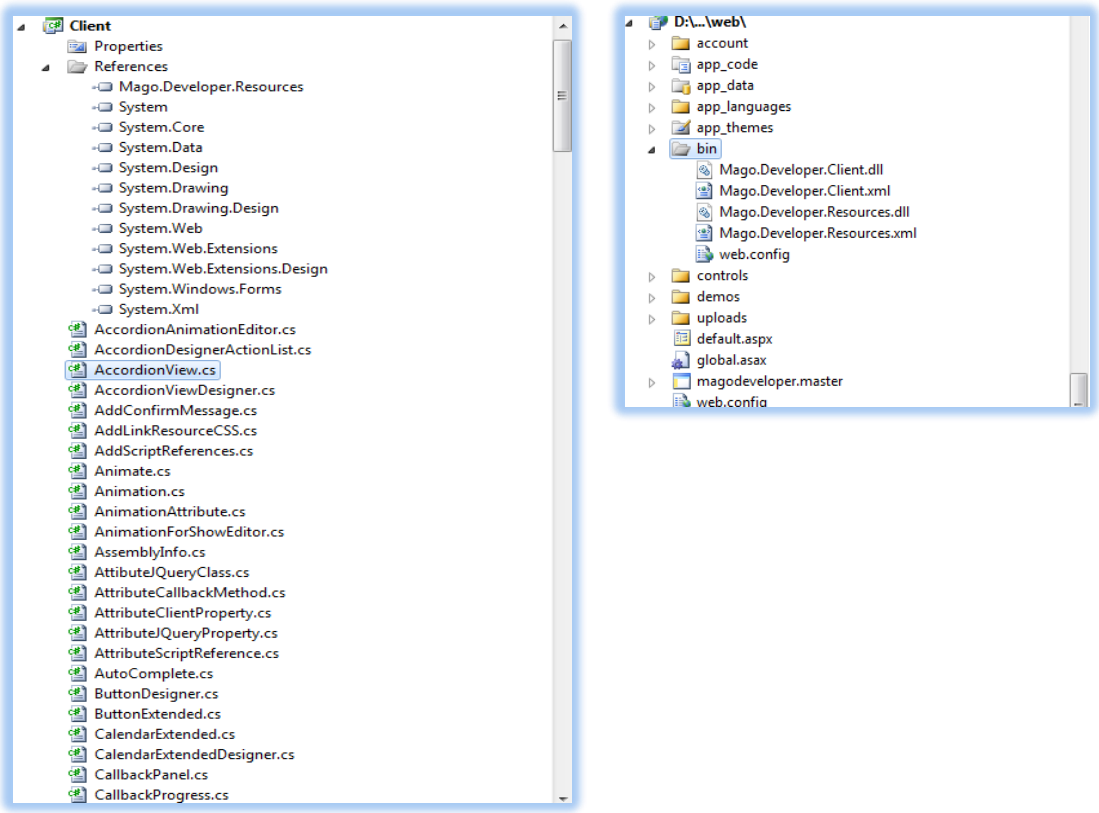

Figura 77. Diferencias ordenación semántica de código, código reutilizable en forma de objetos.

2.5.2.2 Manipulación de objetos de servidor mediante eventos basados en performance *callback*.

✔ **Toolbox *ASP.NET* por defecto.**

Funcionalidades similares estarán disponibles únicamente utilizando el Toolbox externo *AJAX* Control Toolkit.

✔ Toolbox Mago Developer Component.

En el Toolbox Mago Developer Component se han creado dos controles de servidor (*CallbackPanel* y *CallbackProgress*) que permiten ejecutar cualquier tipo de evento de servidor basado en performance *callback* (rendimiento en devoluciones de llamadas asíncronas), a nivel de aplicación cliente, evitando siempre el típico *postback*.

✓ CallbackPanel

Habilita cualquier sección de una página web para que se represente parcialmente de forma asíncrona a nivel de aplicación cliente mediante devoluciones de llamadas asíncronas, evitando el típico *postback*.

✓ CallbackProgress

Proporciona comentarios visuales en el explorador cuando se actualiza el contendido representado parcialmente de forma asíncrona a nivel de aplicación cliente mediante devoluciones de llamadas asíncronas, en los controles *CallbackPanel* asociados.

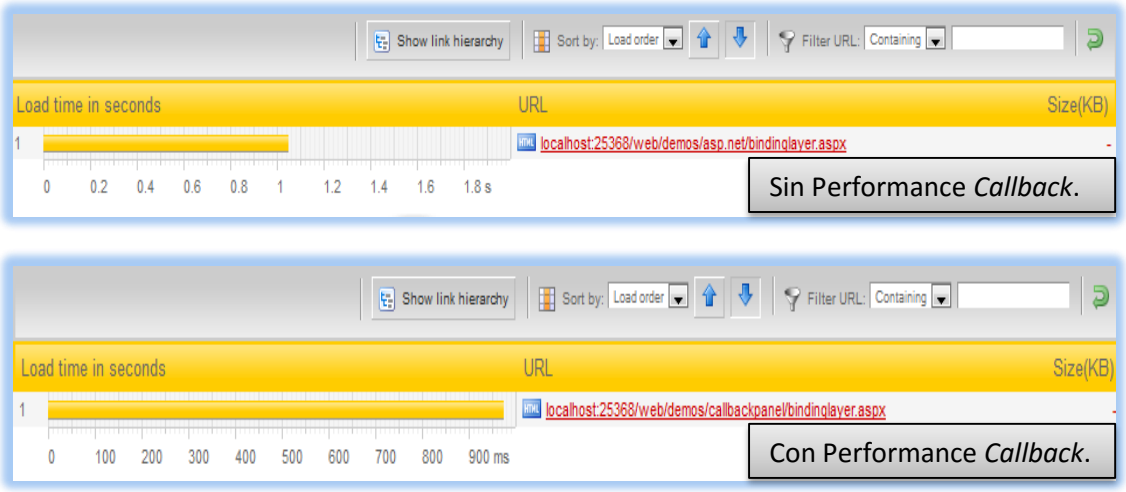

Figura 78. Comparación entre una página sin performance *callback* y otra con esta característica.

(**Fuente Tester:** http://tools.pingdom.com/)

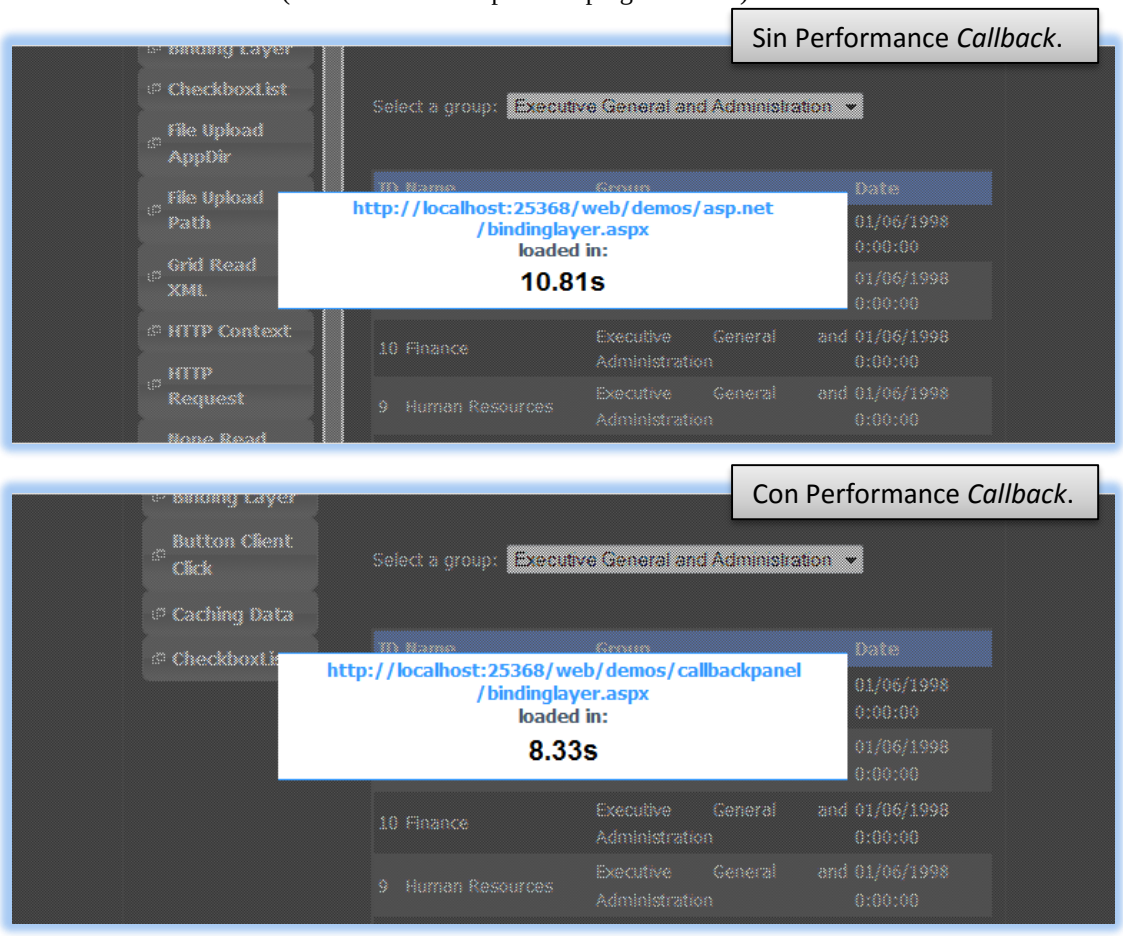

Figura 79. Comparación entre una página sin performance *callback* y otra con esta característica.

(**Fuente Tester:** http://www.webslug.info/)

2.5.2.3 Colección jerárquica de scripts y eventos de aplicación cliente basados en *ECMAScript 262* 5ta. Edición y performance *callback*.

✔ **Toolbox *ASP.NET* por defecto.**

No disponible en ninguna de las versiones del Toolbox del entorno de desarrollo de *ASP.NET*.

✔ **Toolbox Mago Developer Component.**

Se ha creado una colección jerárquica de eventos que operan a nivel de aplicación cliente para cada uno de los objetos de servidor, basados en el estándar *ECMAScript 262* 5ta. Edición que describe la especificación *DOM* (Document Object Model) de nivel 2, además se hace relevante mencionar que todos los eventos disponibles están basados en performance *callback.*

Eventos *DOM* Nivel 2				
Click	BeforeEditFocus	ControlSelect	DragEnd	Finish
Abort	BeforePaste	Copy	DragEnter	Focus
Active	BeforePrint	Cut	DragLeave	FocusIn
AfterPrint	BeforeUnload	Change	DragOver	FocusOut
AfterUpdate	BeforeUpdate	DblClick	DragStart	HashChange
BeforeActive	Blur	DataAvailable	Drop	Help
BeforeCopy	Bounce	DateSetChanged	ErrorUpdate	KeyDown
BeforeCut	CellChange	Deactivate	Error	KeyPress
BeforeDeactivate	ContextMenu	Drag	FilterChange	KeyUp

Load	MouseLeave	Page	RowsDelete	Submit
LayoutComplete	MouseWheel	Paste	RowsInstered	Unload
LoseCapture	Move	Progress	Scroll	TimeOut
Message	MoveEnd	ProgressChange	Select	
MouseDown	ResizeStart	ReadStateChange	SelectionChange	
MouseUp	RowEnter	Reset	SelectStart	
MouseOver	MoveStart	Resize	Start	
MouseMove	Offline	ResizeEnd	Stop	
MouseOut	Online	RowExit	Storage	

2.5.2.4 Creación de nuevos e innovadores objetos con compatibilidad, usabilidad y estandarización para *CSS 3* y *HTML 5.*

✔ **Toolbox Mago Developer Component.**

En el Toolbox Mago Developer component se han creado nuevos e innovadores objetos que cumplen con criterios de funcionalidad, practicidad, estandarización y usabilidad para su utilización en el Toolbox del entorno de desarrollo de *ASP.NET* basado en .NET Framework 2.0 o superior.

Se mencionarán algunos de los nuevos e innovadores controles creados en el Toolbox Mago Developer Component.

✔ **CallbackPanel**

Habilita cualquier sección de una página web para que se represente parcialmente de forma asíncrona a nivel de aplicación cliente mediante devoluciones de llamadas asíncronas, evitando el típico *postback.*

✔ **CallbackProgress**

Proporciona comentarios visuales en el explorador cuando se actualiza el contendido representado parcialmente de forma asíncrona a nivel de aplicación cliente mediante devoluciones de llamadas asíncronas, en los controles *CallbackPanel* asociados.

✔ **Autocomplete**

Proporciona funcionalidades avanzadas de autocompletar contenidos, mediante sugerencias dinámicas de datos coincidentes encontrados, ya sea en un recurso remoto como una base de datos, un web service, o una colección de datos definidos por el usuario, en un control asociado, ya sea un control `<asp:TextBox></asp:TextBox>` o un control de lista desplegable como `<asp:DropDownList></asp:DropDownList>`, `<mago:Comboview></mago:Comboview>`, etc, y además soporta una colección jerárquica de eventos de aplicación cliente.

✔ **DialogBox**

Muestra cuadros de diálogo completamente personalizables, elegantes y funcionales, posesionados de forma *modal*, pueden aceptar un mensaje completamente textual o cualquier contenido insertado en la sección `<BodyTemplate></BodyTemplate>` del mismo, soporte de imágenes que acompañan al mensaje presentado, es redimensionable, arrastrable en las diferentes zonas de la pantalla, soporte completo de estilos *CSS* y de su propiedad avanzada *z-index*, soporta la pulsación de la tecla *ESC* para ocultar y cancelar el mismo, *y* finalmente dispone de soporte para una colección jerárquica de eventos de aplicación cliente.

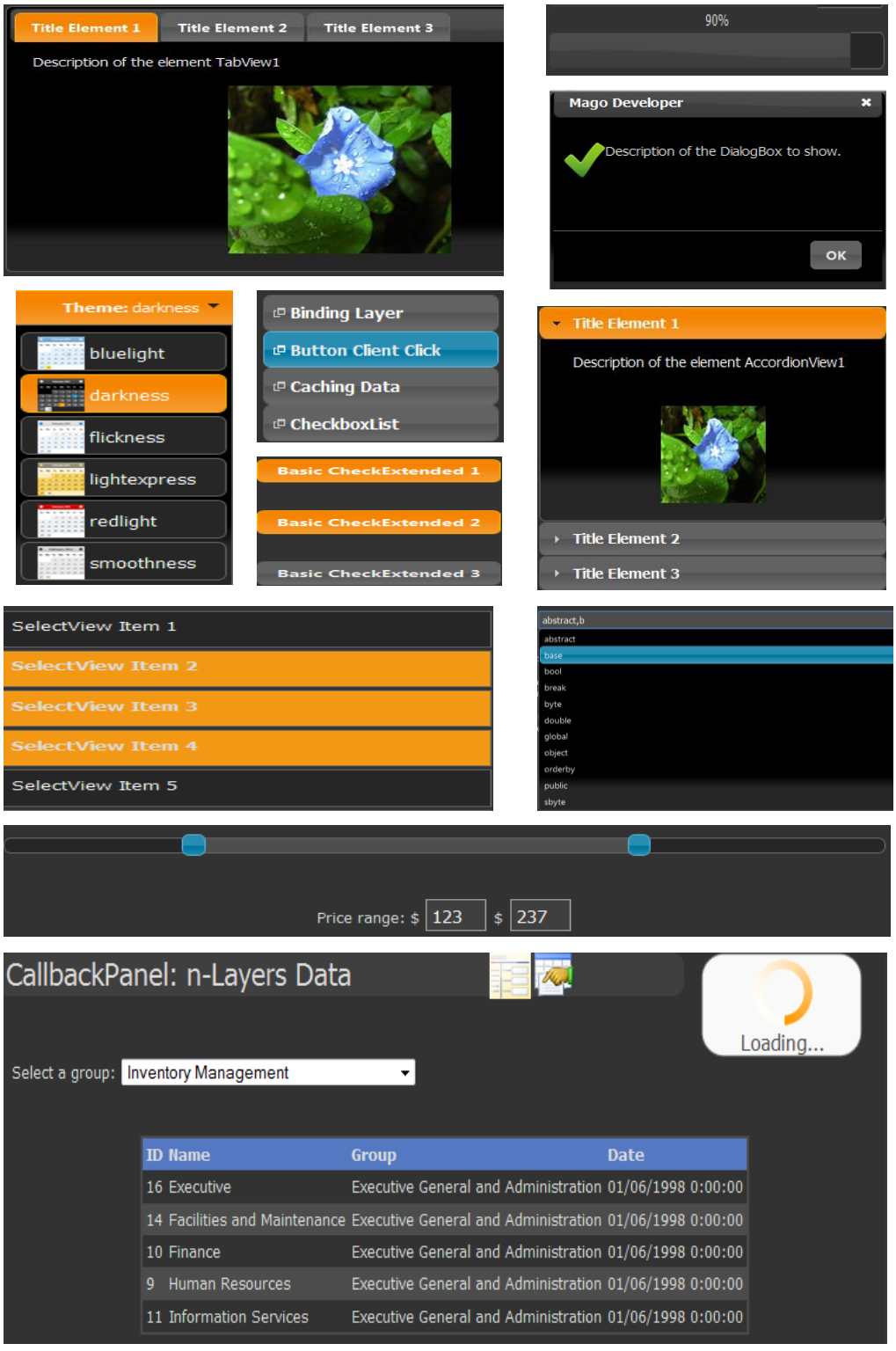

Figura 80. Nuevos e innovadores controles del Toolbox Mago Developer Component.

2.5.2.5 Compatibilidad con múltiples servicios web y recursos *XML*.

En el Toolbox Mago Developer Component se tiene compatibilidad total con recursos *XML* para los controles enlazados a recursos externos de datos, mediante las propiedades `DataSource="" DataSourceID=""` asociadas en cada uno de los controles que aceptan recursos enlazados a datos, por ejemplo el control `<mago:Comboview DataSource="" DataSourceID=""></mago:Comboview>`.

En el Toolbox Mago Developer Component se tiene compatibilidad total con recursos *web service* para los controles enlazados a recursos externos de datos, mediante las propiedades `ServiceMethod="" ServicePath=""` asociadas en cada uno de los controles que aceptan recursos enlazados a datos, por ejemplo el control `<mago:AutoComplete ServiceMethod="" ServicePath=""></mago:AutoComplete>`.

2.5.2.6 Compatibilidad con agentes de usuario para visualización de resultados.

Los controles desarrollados en el Toolbox Mago Developer Component y los más importantes controles aplicados en el Sistema de Gestión de Recursos Humanos de la Universidad Nacional de Chimborazo tienen la compatibilidad y el soporte en los cinco navegadores más importantes del mercado.

Figura 81. Navegadores web más importantes del mercado.

2011	Internet Explorer	Firefox	Chrome	Safari	Opera
Febrero	26.5 %	42.4%	24.1%	4.1%	2.5%
Enero	26.6 %	42.8%	23.8%	4.0%	2.5%
2010	Internet Explorer	Firefox	Chrome	Safari	Opera
Diciembre	27.5 %	43.5%	22.4%	3.8%	2.2%
Noviembre	28.6 %	44.0%	20.5%	4.0%	2.3%
Octubre	29.7 %	44.1%	19.2%	3.9%	2.2%
Septiembre	31.1 %	45.1%	17.3%	3.7%	2.2%
Agosto	30.7 %	45.8%	17.0%	3.5%	2.3%
Julio	30.4 %	46.4%	16.7%	3.4%	2.3%
Junio	31.0 %	46.6%	15.9%	3.6%	2.1%
Mayo	32.2 %	46.9%	14.5%	3.5%	2.2%
Abril	33.4 %	46.4%	13.6%	3.7%	2.2%
Marzo	34.9 %	46.2%	12.3%	3.7%	2.2%
Febrero	35.3 %	46.5%	11.6%	3.8%	2.1%
Enero	36.2 %	46.3%	10.8%	3.7%	2.2%

Figura 82. Estadísticas de los navegadores web más usados en el mercado.

(**Fuente:** http://norfipc.com/internet/navegadores-web.html)

CAPÍTULO III

3 RESULTADOS

- Se ha obtenido un total de 26 nuevos e innovadores controles que cumplen con criterios de funcionalidad, practicidad, estandarización y usabilidad para su utilización en el Toolbox del entorno de desarrollo de *ASP.NET* basado en .NET Framework 2.0 o superior.

- Se han utilizado los más importantes controles del Toolbox Mago Developer Component en el desarrollo del Sistema Informático de Gestión de Recursos Humanos de la Universidad Nacional de Chimborazo.

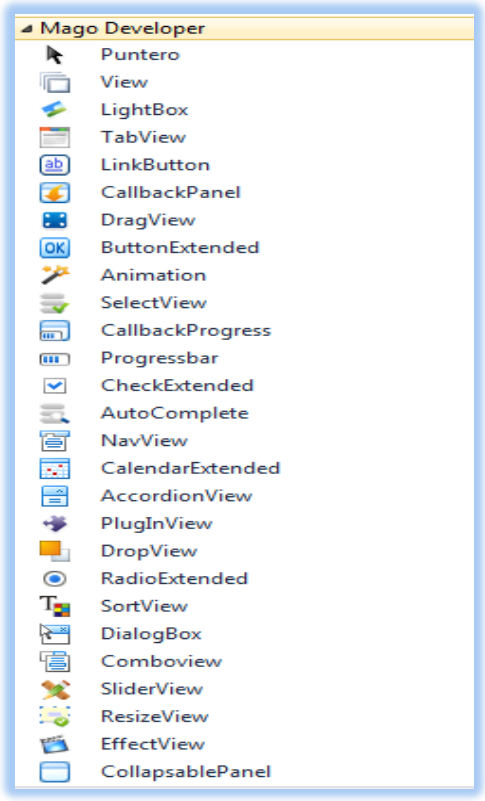

Figura 83. Toolbox Mago Developer Component.

Figura 84. Sistema de Gestión de Recursos Humanos *UNACH*.

- Para determinar las limitaciones existentes en el Toolbox de desarrollo integrado por defecto en el entorno *ASP.NET* se ha realizado un análisis objetivo de la arquitectura, servicios y evolución del mismo.

- Se ha contemplado el ámbito y la funcionalidad de cada uno de los elementos integrados en el Toolbox de desarrollo y su enfoque aplicativo en el desarrollo de sistemas informáticos basados en la web.

- Después de un minucioso análisis de los estándares de desarrollo de: (*CSS*, *HTML*, y *Javascript + compatibilidad con ECMAScript*.) se ha enfatizado en la aplicación de los nuevos estándares de las tecnologías mencionadas, con ello consiguiendo *GUI* de usuario más amigables, más coherentes, más funcionales, y más optimizadas.

- Cada uno de los elementos desarrollados en el Toolbox Mago Developer Component está optimizado para mantener los parámetros de compatibilidad con las plataformas de .NET Framework disponibles.

- Se ha establecido el performance *callback* (rendimiento en devoluciones de llamadas asíncronas) para la colección jerárquica de eventos de cada uno de los objetos de servidor.

- Se ha desarrollado un sistema informático muy fiable, adaptable y sobre todo con escalabilidad a corto y largo plazo, porque se basa en un diseño modular y en capas.

- Se han aplicado las normativas de diseño de aplicaciones web exigidas por la *W3C* (World Wide Web Consortium) tales como: validación externa, adhesión a estándares *HTML*, simplificación y accesibilidad, y las normativas *ISO* tales como: *ISO* / IEC 9126-1:2001, *ISO* / IEC 12207:2008, *ISO* / IEC 24744:2007, *ISO* / IEC CD 29148, *ISO* / IEC 16262: 2002, *ISO* / IEC DIS 16262, *ISO* / IEC 23270:2006, *ISO* / IEC TR 11580:2007, *ISO* / IEC 11581-2:2000, *ISO* / IEC 18036: 2003.

A continuación se describen los módulos desarrollados:

- Autenticación y roles de usuarios.
- Datos personales.
- Datos laborales.
- Nivel de preparación (primaria, secundaria, superior, cuarto nivel).
- Cambios y traslados administrativos.
- Ascensos.
- Reconocimientos.
- Amonestaciones.
- Multas.
- Suspensiones.
- Separación de la institución.
- Subsidio familiar.

CAPÍTULO IV

4 CONCLUSIONES Y RECOMENDACIONES

4.1 Conclusiones

- Se ha contemplado las limitaciones más elocuentes en el Toolbox de desarrollo integrado por defecto en el entorno *ASP.NET* detalladas como: compatibilidad entre las diferentes versiones de .NET Framework, objetos necesarios sin referencia alguna hasta el momento, carencia de algunas propiedades necesarias en algunos objetos, eventos y scripts enviados a través de aplicaciones cliente basados en performance *callback* (rendimiento en devoluciones de llamadas asíncronas), entre otros aspectos importantes, para ello se ha realizado un análisis objetivo de la arquitectura, servicios y evolución del mismo.

- Se ha delimitado el ámbito y la funcionalidad tanto de los elementos del Toolbox de desarrollo *ASP.NET* integrado por defecto como el Toolbox Mago Developer Component, y su enfoque aplicativo en el desarrollo de sistemas informáticos basados en la web.

- Se ha realizado un análisis funcional y evolutivo de los nuevos estándares de las tecnologías (*CSS*, *HTML*, y *Javascript* + *compatibilidad con ECMAScript*.), con lo cual se han conseguido *GUI* de usuario más amigables, más coherentes, más funcionales, más optimizadas y estandarizadas.

- Se ha establecido el performance *callback* (rendimiento en devoluciones de llamadas asíncronas) para la colección jerárquica de eventos de cada uno de los objetos de servidor.

Caso Aplicativo: Sistema de Gestión de Recursos Humanos *UNACH*.

- Se ha seleccionado la tecnología de desarrollo Web *ASP.NET* porque brinda las ventajas de: code-behind, velocidad en tiempos de respuesta, seguridad, robustez, facilidad de mantenimiento, herramientas de trabajo, cache, carpetas especializadas para una función específica, adaptación automática de código a los dispositivos que lo acceden, utilización de *Master Pages*, compatibilidad con *XML* y servicios web, utilización de múltiples lenguajes complementarios.

- La herramienta complementaria *CSS* proporciona múltiples beneficios tales como: control de diseño, redefinición de etiquetas, maquetación y accesibilidad, respeto a los estándares, usabilidad, velocidad, personalización, coherencia y portabilidad.

- Con la utilización de la tecnología *javascript* se ha obtenido las funcionalidades de: páginas web dinámicas, velocidad de transferencia, validaciones dinámicas y en tiempo real, a través de una comunicación asíncrona con el servidor y ejecutando lógica de negocios a nivel de aplicación cliente.

- El modelo de una aplicación en capas ha permitido centralizar el diseño, la lógica y la presentación, a través de lo cual se separa completamente lo que es un recurso remoto de base de datos y la interface de usuario mediante la utilización de una capa intermedia o de lógica de negocios.

- Se han establecido múltiples parámetros de seguridad para el acceso o denegación de usuarios a nivel de aplicación tales como: encriptación de datos críticos del usuario en la base de datos mediante la utilización de los algoritmos *AES* y *Hash*, expiración de sesiones o cookies de autenticación establecidas a través de períodos de inactividad entre el usuario y la aplicación, bloqueo de cuentas de usuario cuando se tengan demasiados intentos fallidos en las credenciales de acceso en la misma solicitud de una cookie de acceso.

4.2 Recomendaciones

- Se deberá hacer un detenido y minucioso análisis de la arquitectura, servicios y evolución del Toolbox de *ASP.NET* integrado por defecto para determinar que nuevos e innovadores objetos son necesarios, que actualizaciones y mejoras necesita cada uno de los mismos.

- Se deberá imprescindiblemente considerar la disponibilidad y evolución de los estándares (*CSS*, *HTML*, y *Javascript + compatibilidad con ECMAScript*.), con lo que se podrá conseguir *GUI* de usuario más amigables, más coherentes, más funcionales, más optimizadas y estandarizadas.

- Uno de los parámetros esenciales que se deberá tener en cuenta en el desarrollo de biblioteca de clases (*componente*), es mantener la compatibilidad con las plataformas de .NET Framework disponibles.

- El factor que inmutablemente mejorará y optimizará sustancialmente los recursos de una aplicación basada en la web es la utilización de performance *callback* (rendimiento en devoluciones de llamadas asíncronas), con lo que se pueden conseguir validación y manipulación de los objetos de servidor optimizadas, a nivel de aplicación cliente y en tiempo real.

Caso Aplicativo: Sistema de Gestión de Recursos Humanos *UNACH*.

- Es indispensable hacer un detenido y minucioso análisis en relación a los requerimientos previos enfocados a las necesidades actuales y futuras del sistema, para ello se debería seguir la metodología de desarrollo de software incremental, que se basa fundamentalmente en proveer una estrategia para controlar la complejidad y los riesgos, desarrollando una parte del producto de software, reservando el resto de aspectos para el futuro.

- Se podrá desarrollar una aplicación web cuando la aplicación requiera características como: interfaces de usuario dinámicas y con facilidades de diseño, desarrollo y mantenimiento, aplicaciones transparentes para el cliente, poca utilización de recursos a nivel de estación servidor, con pocas dependencias y con un altísimo grado de disponibilidad.

- Sí utilizamos *ASP.NET* como herramienta de desarrollo web brindaría los beneficios de: code-behind, velocidad en tiempos de respuesta, seguridad, robustez, facilidad de mantenimiento, herramientas de trabajo, cache, carpetas especializadas para una función específica, adaptación automática de código a los dispositivos que lo acceden, utilización de *Master Pages*, compatibilidad con *XML* y servicios web, utilización de múltiples lenguajes complementarios.

- Es imprescindible y además normalizado por *W3C* e *ISO* utilizar tecnologías que operan a nivel de aplicación cliente como lo es *javascript*, que ayuda a obtener páginas web dinámicas, a través de una comunicación asíncrona con el servidor y ejecutando lógica de negocios a nivel de aplicación cliente.

- Es necesario centralizar el diseño, la lógica de negocios y la presentación o interface de usuario, a través de este modelo se separa completamente lo que es el diseño y las referencias a los recursos remotos de datos a través de una capa intermedia o de lógica de negocios.

CAPÍTULO V

BIBLIOGRAFÍA

General.

- http://www.asp.net/learn/
- http://www.librosweb.es/css/
- http://www.librosweb.es/ajax/
- http://ajax.microsoft.com/ajax/
- http://www.w3.org/TR/CSS21/
- http://www.w3.org/TR/xhtml1/
- http://www.w3.org/TR/REC-xml/
- http://www.asp.net/learn/security/
- http://www.librosweb.es/javascript/
- http://www.cssportal.com/format-css/
- http://www.librosweb.es/css_avanzado/
- http://xml.coverpages.org/iso639a.html
- http://www.colorzilla.com/gradient-editor/
- http://www.iana.org/assignments/media-types/
- http://go.microsoft.com/fwlink/?LinkId=47811/
- http://www.iana.org/assignments/character-sets/
- http://msdn.microsoft.com/es-es/library/52f3sw5c.aspx

- http://msdn.microsoft.com/es-es/library/aa139615.aspx
- http://msdn.microsoft.com/es-es/library/dd566231.aspx
- http://msdn.microsoft.com/es-es/library/bb814937.aspx
- http://msdn.microsoft.com/es-ec/library/d186xcf0.aspx
- http://msdn.microsoft.com/es-es/library/bb972200.aspx
- http://www.w3schools.com/browsers/browsers_stats.asp
- http://es.wikipedia.org/wiki/Callback_(inform%C3%A1tica)
- http://www.desarrolloweb.com/articulos/callback-funciones-jquery.html/
- http://www.readwriteweb.es/analisis/browser-ballot-guerra-navegadores/

Específica.

- http://validator.w3.org/
- http://www.w3.org/TR/html5/
- http://jigsaw.w3.org/css-validator/
- http://msdn.microsoft.com/library/
- http://www.w3.org/TR/css3-roadmap/
- http://msdn.microsoft.com/es-es/vcsharp/
- http://www.cssportal.com/css-properties/
- Material bibliográfico del programa *Microsoft desarrollador 5 estrellas*.
- *ECMAScript Language Specification 5th Edition* / December 2009(.pdf)
- Documentación de Microsoft Visual Studio 2010 Ultimate (*MSDN Library*).

CAPÍTULO VI

APÉNDICES Y ANEXOS

Estándar de propiedades *CSS 1*, *CSS 2.1* y evolución a *CSS 3*

CSS 1 Properties | CSS 2.1 Properties | CSS 3 Properties | CSS Properties Quick Guide

Property	CSS 1	CSS 2 & 2.1	CSS 3
alignment-adjust	✗	✗	✓
alignment-baseline	✗	✗	✓
appearance	✗	✗	✓
azimuth	✗	✓	?
background	✓	✓	✓
background-attachment	✓	✓	✓
background-break	✗	✗	✓
background-clip	✗	✗	✓
background-color	✓	✓	✓
background-image	✓	✓	✓
background-origin	✗	✗	✓
background-position	✓	✓	✓
background-repeat	✓	✓	✓
background-size	✗	✗	✓
baseline-shift	✗	✗	✓
binding	✗	✗	✓
bookmark-label	✗	✗	✓
bookmark-level	✗	✗	✓
bookmark-target	✗	✗	✓
border	✓	✓	✓
border-bottom	✓	✓	✓
border-bottom-color	✗	✓	✓
border-bottom-left-radius	✗	✗	✓
border-bottom-right-radius	✗	✗	✓
border-bottom-style	✗	✓	✓
border-bottom-width	✓	✓	✓
border-break	✗	✗	✓

border-collapse	✗	✓	?
border-color	✓	✓	✓
border-image	✗	✗	✓
border-left	✓	✓	✓
border-left-color	✗	✓	✓
border-left-style	✗	✓	✓
border-left-width	✓	✓	✓
border-length	✗	✗	✓
border-radius	✗	✗	✓
border-right	✓	✓	✓
border-right-color	✗	✓	✓
border-right-style	✗	✓	✓
border-right-width	✓	✓	✓
border-spacing	✗	✓	?
border-style	✓	✓	✓
border-top	✓	✓	✓
border-top-color	✗	✓	✓
border-top-left-radius	✗	✗	✓
border-top-right-radius	✗	✗	✓
border-top-style	✗	✓	✓
border-top-width	✓	✓	✓
border-width	✓	✓	✓
bottom	✗	✓	?
box-align	✗	✗	✓
box-direction	✗	✗	✓
box-flex	✗	✗	✓
box-flex-group	✗	✗	✓
box-lines	✗	✗	✓
box-orient	✗	✗	✓
box-pack	✗	✗	✓
box-shadow	✗	✗	✓
box-sizing	✗	✗	✓

caption-side	✗	✓	?
clear	✓	✓	✓
clip	✗	✓	?
color	✓	✓	✓
color-profile	✗	✗	✓
column-break-after	✗	✗	✓
column-break-before	✗	✗	✓
column-count	✗	✗	✓
column-fill	✗	✗	✓
column-gap	✗	✗	✓
column-rule	✗	✗	✓
column-rule-color	✗	✗	✓
column-rule-style	✗	✗	✓
column-rule-width	✗	✗	✓
column-span	✗	✗	✓
column-width	✗	✗	✓
columns	✗	✗	✓
content	✗	✓	✓
counter-increment	✗	✓	✓
counter-reset	✗	✓	✓
crop	✗	✗	✓
cue	✗	✓	✓
cue-after	✗	✓	✓
cue-before	✗	✓	✓
cursor	✗	✓	✓
direction	✗	✓	?
display	✓	✓	✓
dominant-baseline	✗	✗	✓
drop-initial-after-adjust	✗	✗	✓
drop-initial-after-align	✗	✗	✓
drop-initial-before-adjust	✗	✗	✓

drop-initial-after-align	✗	✗	✓
drop-initial-before-adjust	✗	✗	✓
drop-initial-before-align	✗	✗	✓
drop-initial-size	✗	✗	✓
drop-initial-value	✗	✗	✓
elevation	✗	✓	?
empty-cells	✗	✓	?
fit	✗	✗	✓
fit-position	✗	✗	✓
float	✓	✓	✓
float-offset	✗	✗	✓
font	✓	✓	✓
font-effect	✗	✗	✓
font-emphasize	✗	✗	✓
font-emphasize-position	✗	✗	✓
font-emphasize-style	✗	✗	✓
font-size	✓	✓	✓
font-size-adjust	✗	✓2 ✗2.1	✓
font-smooth	✗	✗	✓
font-stretch	✗	✓2 ✗2.1	✓
font-style	✓	✓	✓
font-variant	✓	✓	✓
font-weight	✓	✓	✓
grid-columns	✗	✗	✓
grid-rows	✗	✗	✓
hanging-punctuation	✗	✗	✓
height	✓	✓	✓
hyphenate-after	✗	✗	✓
hyphenate-before	✗	✗	✓
hyphenate-character	✗	✗	✓
hyphenate-lines	✗	✗	✓
hyphenate-resource	✗	✗	✓

hyphens	✗	✗	✓
icon	✗	✗	✓
image-orientation	✗	✗	✓
image-resolution	✗	✗	✓
inline-box-align	✗	✗	✓
left	✗	✓	?
letter-spacing	✓	✓	✓
line-height	✓	✓	✓
line-stacking	✗	✗	✓
line-stacking-ruby	✗	✗	✓
line-stacking-shift	✗	✗	✓
line-stacking-strategy	✗	✗	✓
list-style	✓	✓	✓
list-style-image	✓	✓	✓
list-style-position	✓	✓	✓
list-style-type	✓	✓	✓
margin	✓	✓	✓
margin-bottom	✓	✓	✓
margin-left	✓	✓	✓
margin-right	✓	✓	✓
margin-top	✓	✓	✓
mark	✗	✗	✓
mark-after	✗	✗	✓
mark-before	✗	✗	✓
marker-offset	✗	✓2 ✗2.1	?
marks	✗	✓2 ✗2.1	✓
marquee-direction	✗	✗	✓
marquee-play-count	✗	✗	✓
marquee-speed	✗	✗	✓
marquee-style	✗	✗	✓
max-height	✗	✓	✓

max-width	✗	✓	✓
min-height	✗	✓	✓
min-width	✗	✓	✓
move-to	✗	✗	✓
nav-down	✗	✗	✓
nav-index	✗	✗	✓
nav-left	✗	✗	✓
nav-right	✗	✗	✓
nav-up	✗	✗	✓
opacity	✗	✗	✓
orphans	✗	✓	✓
outline	✗	✓	✓
outline-color	✗	✓	✓
outline-offset	✗	✗	✓
outline-style	✗	✓	✓
outline-width	✗	✓	✓
overflow	✗	✓	✓
overflow-style	✗	✗	✓
overflow-x	✗	✗	✓
overflow-y	✗	✗	✓
padding	✓	✓	✓
padding-bottom	✓	✓	✓
padding-left	✓	✓	✓
padding-right	✓	✓	✓
padding-top	✓	✓	✓
page	✗	✓2 ✗2.1	✓
page-break-after	✗	✓	✓
page-break-before	✗	✓	✓
page-break-inside	✗	✓	✓
page-policy	✗	✗	✓
pause	✗	✓	✓
pause-after	✗	✓	✓

pause-before	✗	✓	✓
phonemes	✗	✗	✓
pitch	✗	✓	?
pitch-range	✗	✓	?
play-during	✗	✓	?
position	✗	✓	?
presentation-level	✗	✗	✓
punctuation-trim	✗	✗	✓
quotes	✗	✓	✓
rendering-intent	✗	✗	✓
resize	✗	✗	✓
rest	✗	✗	✓
rest-after	✗	✗	✓
rest-before	✗	✗	✓
richness	✗	✓	?
right	✗	✓	?
rotation	✗	✗	✓
rotation-point	✗	✗	✓
ruby-align	✗	✗	✓
ruby-overhang	✗	✗	✓
ruby-position	✗	✗	✓
ruby-span	✗	✗	✓
size	✗	✓2 ✗2.1	✓
speak	✗	✓	?
speak-header	✗	✓	?
speak-numeral	✗	✓	?
speak-punctuation	✗	✓	?
speech-rate	✗	✓	?
stress	✗	✓	?
string-set	✗	✗	✓
tab-side	✗	✗	✓
table-layout	✗	✓	?

target	✗	✗	✓
target-name	✗	✗	✓
target-new	✗	✗	✓
target-position	✗	✗	✓
text-align	✓	✓	✓
text-align-last	✗	✗	✓
text-decoration	✓	✓	?
text-emphasis	✗	✗	✓
text-height	✗	✗	✓
text-indent	✓	✓	✓
text-justify	✗	✗	✓
text-outline	✗	✗	✓
text-replace	✗	✗	✓
text-shadow	✗	✓2 ✗2.1	✓
text-transform	✓	✓	?
text-wrap	✗	✗	✓
top	✗	✓	?
unicode-bibi	✗	✓	?
vertical-align	✓	✓	✓
visibility	✗	✓	✓
voice-balance	✗	✗	✓
voice-duration	✗	✗	✓
voice-family	✗	✓	✓
voice-pitch	✗	✗	✓
voice-pitch-range	✗	✗	✓
voice-rate	✗	✗	✓
voice-stress	✗	✗	✓
voice-volume	✗	✗	✓
volume	✗	✓	?
white-space	✓	✓	✓
white-space-collapse	✗	✗	✓

Property	CSS 1	CSS 2 & 2.1	CSS 3
voice-rate	✗	✗	✓
voice-stress	✗	✗	✓
voice-volume	✗	✗	✓
volume	✗	✓	?
white-space	✓	✓	✓
white-space-collapse	✗	✗	✓
widows	✗	✓	✓
width	✓	✓	✓
word-break	✗	✗	✓
word-spacing	✓	✓	✓
word-wrap	✗	✗	✓
z-index	✗	✓	?

Introducción al estándar *HTML 5*

HTML 5 mejora la interoperabilidad y reduce los costes de desarrollo, adaptando las normas precisas sobre la forma de manejar todos los elementos *HTML*, y cómo recuperarse de los errores.

Algunas de las nuevas características de *HTML 5* son las funciones de audio que encajan, vídeo, gráficos, almacenamiento de datos del lado del cliente, y los documentos interactivos.

El grupo de trabajo de *HTML 5* incluye *AOL*, *Apple*, *Google*, *IBM*, *Microsoft*, *Mozilla*, *Nokia*, *Opera*, y cientos de otros proveedores.

Etiquetas *HTML 5* según *W3C*

<!-- -->	Define un comentario.
<!DOCTYPE>	Define el tipo de documento.
<a>	Define un hipervínculo.
<abbr>	Define una abreviatura.
<acronym>	No es compatible con *HTML 5*.
<address>	Define un elemento de dirección.
<applet>	No es compatible con *HTML 5*.
<area>	Define un área dentro de un mapa de imágenes.
<article>	Define un artículo.
<aside>	Define el contenido aparte de los contenidos de la página.
<audio>	Define el contenido de sonido.
<b>	Define el texto en negrita.
<base>	Define una *URL* base para todos los enlaces en una página.
<basefont>	No es compatible con *HTML 5*.
<bdo>	Define la dirección de visualización de texto.
<big>	No es compatible con *HTML 5*.
<blockquote>	Define una larga cita.

<body>	Define el contenido o cuerpo de un documento *HTML*
 	Inserta un salto de línea único.
<button>	Define un botón.
<canvas>	Define gráficos.
<caption>	Define el título de una tabla.
	No es compatible con *HTML 5*.
<cite>	Define una cita.
<code>	Define el texto de código específico de un lenguaje de programación.
<col>	Define los atributos de columnas de la tabla.
<colgroup>	Define los grupos de columnas de la tabla.
<command>	Define un botón de comando.
<datalist>	Define una lista desplegable.
<dd>	Define una descripción de la definición.
<del>	Define el texto eliminado (línea horizontal que atraviesa sobre el texto).
<details>	Define los detalles de un elemento.
<dfn>	Define una definición de término.
<dir>	No es compatible con *HTML 5*.
	Define una sección en un documento.

<dl>	Define una lista de definiciones.
<dt>	Define una definición de término.
	Define texto enfatizado.
<embed>	Define contenido interactivo externo o *plugin*.
<fieldset>	Define un fieldset (establecer campos de formulario).
<figcaption>	Define el título de un elemento gráfico.
<figure>	Define un grupo de contenidos de medios, y su leyenda.
<font>	No es compatible con *HTML 5*.
<footer>	Define un pie de página de una sección o página.
<form>	Define un formulario.
<frame>	No es compatible con *HTML 5*.
<frameset>	No es compatible con *HTML 5*.
<h1> a <h6>	Define texto de encabezamiento en tamaños predefinidos desde *h1* a *h6*.
<head>	Define la información de encabezamiento sobre el documento.
<header>	Define un encabezado de una sección o página.
<hgroup>	Define la información sobre una sección de un documento.
<hr>	Define una regla horizontal.
<html>	Define un documento *html*.

<i>	Define el texto en cursiva.
<iframe>	Define una sub ventana de línea (estructura).
<img>	Define una imagen.
<input>	Define un campo de entrada.
<ins>	Define el texto insertado.
<keygen>	Define una clave generada en un formulario.
<kbd>	Define el texto del teclado.
<label>	Define una etiqueta para un control de formulario.
<legend>	Define un título en un fieldset (establecer campos de formulario).
	Define una lista de elementos.
<link>	Define un recurso de referencia.
<map>	Define una imagen del mapa.
<mark>	Define el texto marcado.
<menu>	Define un menú de la lista.
<meta>	Define la información de meta contenido.
<meter>	Define la medida dentro de un rango predefinido.
<nav>	Define los enlaces de navegación.
<noframes>	No es compatible con *HTML 5*.

<noscript>	Define una sección *noscript*.
<object>	Define un objeto incrustado.
	Define una lista ordenada.
<optgroup>	Define una opción de grupo.
<option>	Define una opción en una lista desplegable.
<output>	Define algunos tipos de salida.
<p>	Define un párrafo.
<param>	Define un parámetro para un objeto.
<pre>	Define texto preformateado.
<progress>	Define el progreso de una tarea de cualquier tipo.
<q>	Define una breve cita.
<rp>	Se utiliza en las anotaciones de *rubí* para definir lo que deben mostrar los navegadores que no soporten el elemento de *rubí*.
<rt>	Define la explicación de anotaciones *rubí*.
<ruby>	Define anotaciones *rubí*.
<s>	Define el texto (etiqueta obsoleta).
<samp>	Define el código máquina de muestra.
<script>	Define una secuencia de comandos.

<select>	Define una lista seleccionable.
<small>	Define un pequeño fragmento de texto.
<source>	Define recursos multimedia.
	Define una sección en un documento.
<strike>	No es compatible con *HTML 5*.
	Define un énfasis de texto (similar a texto en negrita).
<style>	Define estilos de los elementos *HTML*.
<sub>	Define el texto en subíndice.
<summary>	Define el encabezado de un "details" elemento.
<sup>	Define el texto en superíndice.
<table>	Define una tabla.
<tbody>	Define el cuerpo o contenido de una tabla.
<td>	Define una celda de tabla.
<textarea>	Define un área de texto.
<tfoot>	Define un pie de página de una tabla.
<th>	Define un encabezamiento o elemento cabecera de tabla.
<thead>	Define una cabecera de una tabla.
<time>	Define una fecha / hora.

<title>	Define el título del documento.
<tr>	Define una fila de la tabla.
<tt>	No es compatible con *HTML 5*.
<u>	No es compatible con *HTML 5*.
	Define una lista desordenada.
<var>	Define una variable.
<video>	Define un vídeo.
<wbr>	Define una posible línea de rotura (*line-break*).
<XMP>	No es compatible con *HTML 5*.