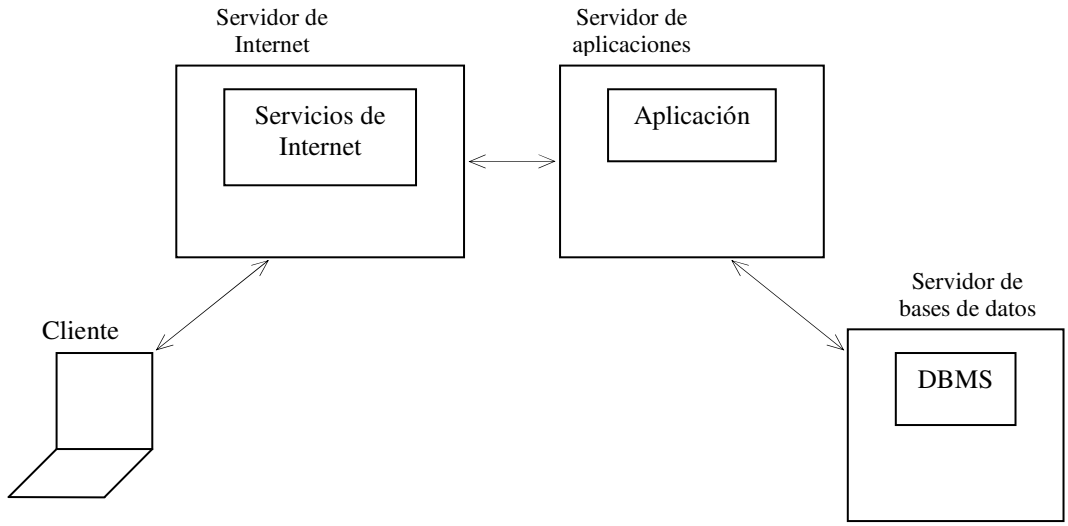
4 Aplicaciones multicapa

4.1 Introducción

Se le da este nombre a las aplicaciones que tienen al menos tres niveles de componentes de software. El siguiente diagrama ejemplifica esta arquitectura:



Características principales

El procesamiento de la aplicación sigue siendo distribuido como en el caso de aplicaciones tipo cliente-servidor. Esto es, el procesamiento se lleva a cabo por varios componentes (unidades) de software.

En este esquema normalmente se tiene: cliente delgado (normalmente un navegador), servidor de Internet (Web), (opcionalmente) servidor de aplicaciones y servidor de bases de datos.

Ubicación de las lógicas de procesamiento:
- Presentación: en el cliente delgado.
- Negocio: en el servidor de Internet, pudiendo ser conjuntamente con el servidor de aplicaciones.
- Datos: en el servidor de bases de datos.

– **Ventajas**:
  - Facilidad para el mantenimiento de la lógica del negocio: cambios en las reglas sólo se hacen en la(s) capa(s) intermedia(s).
  - Se descarga de trabajo al DBMS.
  - No es necesario instalar actualizaciones en los clientes.
  - Puede permitir la integración de ambientes de cómputo heterogéneos.
– **Desventajas**:
  - El tráfico en la red puede aumentar por la cantidad de solicitudes que se hacen desde los clientes.
  - Estas aplicaciones son más complejas que las de cliente-servidor
  - El diálogo entre las tres capas es más avanzado que las típicas instrucciones de SQL.
  - Se deben manipular posibles fallas como: errores en la red, pérdidas de conexiones o cuellos de botella.
  - Se requiere un mayor grado de análisis y planeación.

Componentes de las aplicaciones de Internet

Aquí se describen brevemente las características principales de las diferentes unidades de software que componen a este tipo de aplicaciones.

Clientes

Normalmente los componentes en esta capa son navegadores de Internet cuya función básica es desplegar páginas enviadas desde un servidor de Internet, pudiendo, en ocasiones, capturar datos dados por los usuarios y tener programación.

Servidores de Internet (Web servers)

Son los servidores que se encargan de administrar las páginas que son enviadas a los navegadores, pudiendo, en muchas ocasiones, llevar a cabo procesamiento de código en algún lenguaje de programación. También pueden proveer seguridad.

Sus características principales son:
– Tener como función primaria el administrar las páginas de una aplicación y enviarlas a los navegadores.
– Poder realizar labores como: procesar información de usuarios, proveer seguridad y ejecutar código.
– Hacer labores complementarias a las de un servidor de aplicaciones, aunque varias de sus funciones se pueden traslapar.

Servidores de aplicaciones

Es software especializado que puede llevar a cabo todo el procesamiento de la lógica del negocio de la aplicación. Es un software más poderoso que un servidor de Internet pudiendo efectuar procesamiento transaccional de la información, manejar varias conexiones al mismo tiempo a un DBMS, hacer un balanceo de carga cuando se tiene acceso simultáneo a varios DBMS, y otras funciones especializadas relacionadas con este tipo de aplicaciones. Muchos fabricantes ya están integrando los servidores de Internet con los de aplicaciones en un solo software.

Servidor de bases de datos

Es la parte relacionada con el DBMS que se encarga de llevar el control de las bases de datos que son accedidas por estas aplicaciones. En capítulos previos se han cubierto sus principales funcionalidades.

4.2 Aplicaciones de Internet con Visual C#

En esta sección se describen algunas cuestiones importantes de las aplicaciones de Internet, en general, dándole énfasis a las aplicaciones desarrolladas con Visual C#.

4.2.1 Características básicas

Se puede usar Visual Studio .NET para crear **aplicaciones de Internet** (**Web applications**). Estas aplicaciones, también conocidas como **aplicaciones basadas en Internet** (**Web-based applications**), pueden usar Visual C# en combinación con la **tecnología ASP .NET**, de Microsoft, para crear **contenido de Internet** (**Web content**).

Web content es un término que se usa para englobar a datos, documentos HTML (o XML) e imágenes que pueden ser vistos en un navegador de Internet.

ASP.NET permite la creación de proyectos con formas de Internet (Web forms) y servicios de Internet (Web Services).

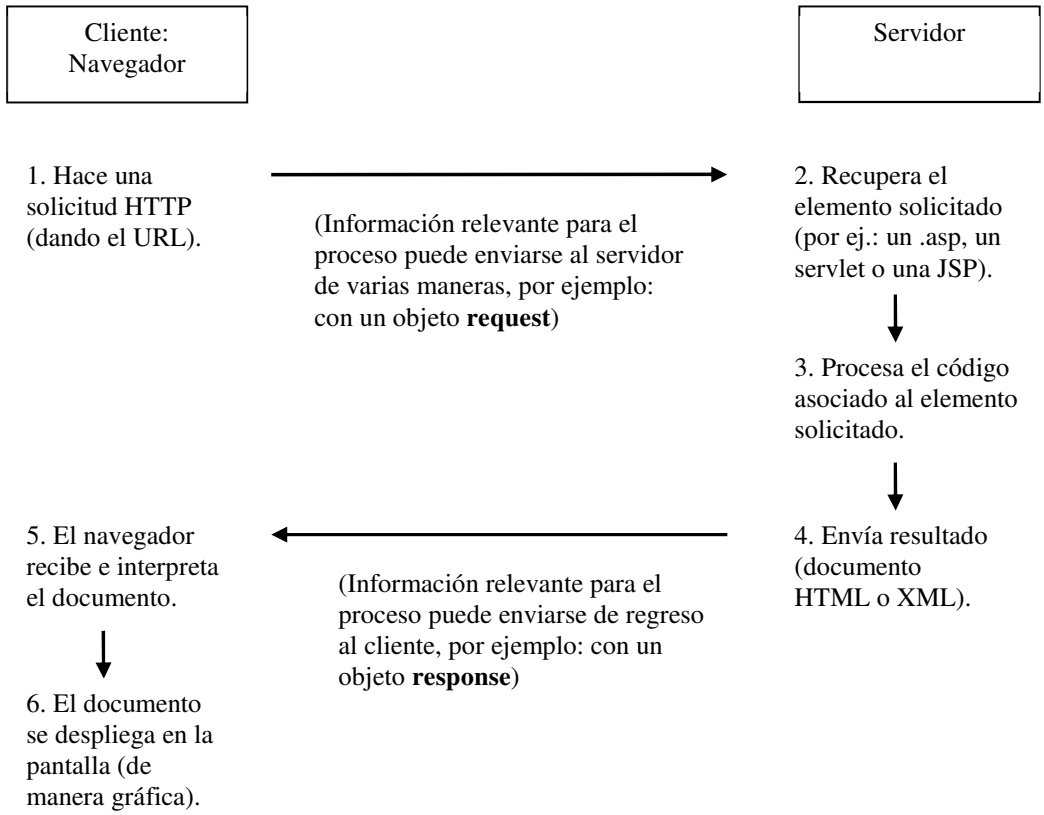
Las Web forms son similares a las formas de Windows, sólo que en un servidor; normalmente se les llama también páginas y en la terminología Microsoft se abrevian como ASP's (Active Server Pages). Los Web Services son componentes que proveen funcionalidad que puede usarse en Internet.

Microsoft proporciona tres tipos de servidores Web:
- **Visual Studio Developer:** es el servidor anterior al IIS Express usado para el desarrollo local de las aplicaciones Web.
- **IIS Express:** el servidor actual usado, fundamentalmente, para desarrollar **localmente** la aplicación Web.
- **Internet Information Server** (**IIS**): el cual se usa para dar servicio a una aplicación Web accedida masivamente por una gran cantidad de clientes. El IIS, normalmente, debería ubicarse en una máquina servidora que atiende a muchos clientes.

4.2.2 Atención de peticiones desde un navegador

Tanto en el ambiente de ASP.NET (Microsoft) como en el ambiente de Java, el proceso que se sigue para atender solicitudes (peticiones) hechas desde un navegador hacia un servidor[1] es el que se muestra, a grandes rasgos, en el siguiente esquema:

[1] Se usará el término: **servidor**, para hacer referencia tanto a un servidor de Internet, como a uno de aplicaciones, en el entendimiento de que pueden existir ambos o sólo alguno de los dos. Uno u otro son capaces de recibir y procesar las peticiones realizadas desde un navegador.



4.2.3 Eventos

Los eventos pueden ocurrir en el cliente o en el servidor, pero el código, en general, se ejecuta en el servidor. Cuando ocurre un evento en algún control de la página **no necesariamente** se envía de inmediato al servidor para que sea procesado. Por ejemplo, cuando ocurre el evento Click en un botón, se envía automáticamente al servidor, pero no sucede así con otros eventos u otros controles.

Cuando han ocurrido una serie de eventos que no se envían de inmediato al servidor, todos ellos son procesados (postbacked) cuando ocurre el primero que **sí** se envía inmediatamente al servidor. Así, se debe tener cuidado en la programación de los mismos para no tener efectos indeseados por el orden de ejecución de los eventos. Se puede poner la propiedad **AutoPostBack** de un control en **true**, para que los eventos del mismo se envíen inmediatamente al servidor.

4.2.4 Objetos predefinidos

Las páginas de una aplicación de Internet se dice, en general, que son **"sin estado" (stateless)** ya que no almacenan información acerca de su contenido después de hacer un viaje al servidor. Sin embargo, existe una excepción a esto, en Visual C#, ya que si la propiedad **EnableViewState** de un control está en **True**, entonces éste guarda su estado actual entre una invocación y otra de la página.

Aunado a lo anterior, Visual C# proporciona varios objetos que pueden usarse dentro de una aplicación para acceder a información de ésta. También, cuando se quiere transferir información entre páginas, se pueden emplear algunos de ellos para tal fin. Estos objetos son:

Session: existe uno por cada usuario de una aplicación y proporciona información acerca de la sesión de un usuario actual. Es un objeto que se mantiene en el servidor. Se puede usar para almacenar pares: **(clave, valor)**, para poder intercambiar información entre páginas o con el servidor. Cuando una aplicación tiene muchos usuarios, se puede degradar la respuesta del servidor.
Application: existe uno por aplicación. Existe en el servidor y es usado algunas veces para almacenar información acerca del programa, tal como la cantidad de veces que una página se ha solicitado. Es menos usado que el objeto anterior.
Request: contiene información acerca del usuario actual, datos dados por el mismo, y parámetros enviados con una solicitud al servidor.
Response: permite enviar datos de respuesta HTTP a un cliente y contiene información sobre esa respuesta. En particular, se puede usar el método **Redirect("NombrePágina.aspx")** para transferir el control a una página de una aplicación
Server: proporciona acceso a los métodos y propiedades del servidor. En particular, se puede usar el método **Transfer("NombrePágina.aspx")** para transferir el control a una página de una aplicación.

4.2.5 Controles

En este apartado se describen las características principales de controles de uso común en este tipo de aplicaciones (aunados a los básicos como Label, TextBox, etc.). Entre éstos se encuentran los siguientes:

- Control **ImageButton**. Sirve para desplegar imágenes en una ASP. En la propiedad ImageURL se especifica la dirección del archivo cuya imagen se mostrará con el control.
- Control **Table.** Hay dos posibilidades para este control: una se tiene insertándolo desde la caja de herramientas; otra, insertándolo con el menú Tabla, Insertar tabla.
  - **Alternativa 1** (**inserción desde la caja de herramientas**): en este caso el control sirve para desplegar una tabla simple en una ASP, conteniendo únicamente texto en cada una de sus celdas. Las propiedades importantes del control son:
    BorderStyle: sirve para especificar el tipo de contorno que tendrá la tabla.
    GridLines: dibuja separadores entra las celdas que conforman a la tabla.
    Rows: proporciona un asistente el cual permite definir la cantidad de filas que tendrá la tabla. Al abrir el asistente se define dicha cantidad y se especifica para cada fila, la cantidad de columnas que cada una tendrá (también por medio de otro asistente que se activa con la propiedad Cells de las filas). Para cada celda se puede especificar, con la propiedad Text, su contenido particular; la especificación se puede hacer tanto en tiempo de diseño, como en ejecución (con programación).
  - **Alternativa 2** (**inserción desde el menú Tabla, Insertar tabla**): en este caso el control sirve para desplegar una tabla más elaborada que la anterior. Al insertar la tabla se muestra un asistente en el cual se pueden especificar varias de las características de la tabla (cantidad de columnas, de filas, contenido de celdas, etc.). Después de cerrar el asistente, se puede volver a abrir otro que permite dar el estilo que tendrá la tabla al visualizarse (color, fuente, bordes, etc.). Este asistente se abre con la propiedad Style de la tabla.
    La diferencia más importante entre ambos tipos de tabla es que el segundo permite agregar otros controles Web en sus celdas (etiquetas, cajas de texto, botones, etc.), permitiendo hacer la misma programación con ellos que la que se hace sin estar dentro de tablas. La ventaja de tenerlos así es que al desplegarlos en el navegador, éste los

"acomoda" mejor para su visualización, sin necesidad de tener que estar calculando la posición dónde deben ir para que se vean bien en la pantalla.

– Control **HyperLink**. Sirve para ir directamente a una página de la misma aplicación o de otra. En la propiedad NavigateUrl se especifica la dirección de la página a la cual se irá. La dirección puede ser *absoluta* (hay que indicar: http://, en qué máquina y directorio está la página, etc.) o *relativa* (la página debe localizarse en el mismo directorio que la aplicación actual). Cuando hay que hacer actividades adicionales antes de pasar a otra página, se debe usar un **LinkButton** para efectuarlas; este botón es como un botón de comando y permite especificar código en el procedimiento asociado a su evento Click. Después, para pasar a otra página, hay que hacerlo con alguno de los objetos del ambiente de Internet (ver 4.2.4).
– Control **GridView:** similar al DataGridView de las aplicaciones Windows. Hay que especificar en su propiedad DataSource la fuente de datos que se usará para llenarlo (por ejemplo, una tabla de un DataSet). Después, hay que llamar al método DataBind de la página para que se muestren los datos en el GridView.
Ejemplo:

```
GrdDetalle.DataSource = DsDet.Tables["Det"]; //Asigna tabla al Grid.
GrdDetalle.DataBind();
```

4.2.6 Desarrollo de una aplicación local de Internet

A continuación se describen los pasos básicos a seguir para crear una aplicación local de Internet[2]:

1) En el IDE de Visual C#: menú Archivo, Nuevo sitio Web…, en Plantillas: elegir el lenguaje (Visual C#), seleccionar Sitio web vacío de ASP.NET, Ubicación web: Sistema de archivos, seleccionar el directorio donde quedará el proyecto, escribir el nombre de la aplicación (*AplicaciónWebBD*), <Aceptar>.
2) Agregar la primera página al proyecto: en el Explorador de soluciones, colocarse en el proyecto, menú breve, Agregar, Formularios Web Forms, si es la primera página, dejar el nombre indicado para la misma: Default, <Agregar>. Se crea la página: Default.aspx
3) **Se puede dar un nombre significativo** a cualquier página del proyecto: seleccionar la página en el árbol del proyecto, dar el nuevo nombre en la propiedad Nombre de archivo (sin tocar la extensión .aspx). También se pueden especificar valores en algunas propiedades significativas de la página (p. ej.: bgColor**,** Title**,** etc.). **Así con todas las páginas** de la aplicación.
4) **Ejecutar:** si el IDE indica que no está habilitada la opción de depuración, seleccionar: Ejecutar sin depuración (o <ctrl><F5>).
5) **Crear la interfaz gráfica** de la página agregando los controles de Internet que se encuentran en el cuadro de herramientas. **Establecer valores en las propiedades significativas** de cada control de la interfaz (p. ej.: BackColor, BorderColor, Enabled, Font, Text, Visible, etc.). **Dar nombre significativo** a aquellos controles que se usarán en la programación (propiedad **(ID)**).
6) **Especificar el código de la aplicación**. En la ventana del explorador de la solución, expandir el nodo de la página a programar, doble clic al archivo asociado (extensión .aspx.cs) el cual contendrá el código asociado a la página, programar el código correspondiente. **Repetir esto** para todas las páginas de la aplicación.

[2] Si es la primera aplicación Web que se crea, en el IDE de Visual C# ir a: menú HERRAMIENTAS, Opciones…, Proyectos y soluciones, Ubicación de proyectos. Ahí especificar la carpeta general donde quedarán estos proyectos, por ejemplo: C:\BD\WebSitesCS.

4.2.7 Controles de validación

Son controles que se pueden usar para aplicar ciertas reglas de validación a los datos que un usuario introduce en una página de internet. Por ejemplo, se puede comprobar si un dato se dio en una caja de texto o no, si el dato sigue un formato determinado o si se ajusta a un rango de valores, entre otras comprobaciones.

A los diversos controles gráficos de una página, principalmente a las cajas de texto, se les pueden asociar varios controles de validación para determinar si cumplen con los criterios establecidos para los datos. Se pueden crear grupos de validación en una página para validar selectivamente a los controles gráficos de cada grupo creado.

La validación normalmente se hace en el servidor, aunque también existe la posibilidad de hacerla en el cliente, en cuyo caso el navegador debe tener habilitada esta opción (puede no ser trivial el hacer esta habilitación).

La siguiente tabla resume estos controles:

Control de validación	Propósito	Propiedades a definir
RequiredFieldValidator	Se deben dar datos en el control asociado.	ControlToValidate ErrorMessage
CompareValidator	Compara el valor en el control gráfico con otro control gráfico o con una constante. En Operator se elige el operador de comparación. Se puede establecer la propiedad Type para verificar que la entrada se pueda convertir a ese tipo de datos.	ControlToValidate ControlToCompare o ValueToCompare Operator Type ErrorMessage
RangeValidator	Valida que la entrada esté en un rango de valores.	ControlToValidate MinimumValue MaximumValue Type ErrorMessage
RegularExpressionValidator	Valida contra una expresión regular, tal como una cantidad requerida de dígitos, o un valor con formato tal como un RFC.	ControlToValidate ValidationExpression (existe un lenguaje asociado para construir la expresión regular[3]) ErrorMessage
ValidationSummary	Despliega un resumen de todos los mensajes de los otros controles de validación.	DisplayMode

En los validadores se puede colocar una misma cadena (por ejemplo, AllValidators) en la propiedad ValidationGroup para agruparlos y continuar con el procesamiento de la aplicación, cuando todos los validadores cumplen exitosamente con sus comprobaciones. De esta manera, también se pueden crear diferentes grupos de validación dentro de una misma página.

Cuando ocurre un error de validación, en lugar de mostrar el error al lado del control en que ocurrió, se puede usar un ValidationSummary, al final de la página, para ahí desplegarlo. En su propiedad ValidationGroup se coloca la cadena del grupo de validadores cuyos errores se

[3] Por ejemplo, para validar un RFC habría que dar la expresión: \w{4}\d{6}(\w|\d){3}
Para ver la Ayuda de expresiones regulares: colocarse en el control de validación, <F1>, Expresiones regulares de .NET Framework, Elementos del lenguaje de expresiones regulares.

desplegarán con este control (por ejemplo, AllValidators). En este caso, en la propiedad Text del validador asociado al control gráfico se coloca un asterisco para que éste se despliegue cuando ocurre el error.

Cada validador tiene una propiedad IsValid que se coloca en true en caso de que no haya error en el control asociado. Igual, se puede usar el objeto Page, junto con IsValid, para verificar que en todos los controles de las páginas no hay error alguno. Por ejemplo:

```
if (RequiredFieldValidator1.IsValid) {
        'Ejecutar algún procesamiento.
}

if (Page.IsValid) {
        'Ejecutar algún procesamiento.
}
```

Las instrucciones anteriores deberían escribirse en el botón que envía al servidor de Internet la petición del procesamiento de la página en que aparecen los controles de validación. Adicionalmente, debe establecerse el valor **true** en su propiedad **CausesValidation** y la misma cadena del grupo de controles que se va a validar (por ejemplo, AllValidators) en la propiedad ValidationGroup.

En caso de que al dar clic al botón no se lleve a cabo la validación, hay que verificar que en el código html del botón esté la instrucción: causesvalidation="true".

Si la validación se va a disparar con otro control gráfico, por ejemplo un DDL, habría que hacer lo mismo que en el caso del botón.