Unidad Didáctica IX.- Tecnología Visual Studio.net

*El equipo de **SI Andalucía** sigue abordando nuevos proyectos. Actualmente tienen abierta una línea de investigación sobre las herramientas que hay disponibles en el mercado para el desarrollo de aplicaciones en entornos gráficos. Se han decidido a estudiar la alternativa de Microsoft que se conoce como plataforma .NET. Los encargados de esta tarea han sido **Carmen** y **Víctor**, que acaban de finalizar la aplicación web para **El PC Feliz**.*

*Su trabajo va a consistir en analizar alguna de las herramientas incluidas en la plataforma .NET. Piensan en **Visual Studio .NET** o alguna de las versiones gratuitas que se ofrecen de esta plataforma como **Visual Basic Express Edition**. Se deciden por esta última ya que no necesitan las características avanzadas de la primera por ahora.*

A lo largo de este módulo hemos estudiado una herramienta de generación de aplicaciones de escritorio y aplicaciones web como es Oracle JDeveloper, utilizando como lenguaje de programación Java y como gestor de base de datos Oracle 10g Express Edition. No queríamos dejar pasar la oportunidad de manejar alguna otra herramienta que hay en el mercado para desarrollo de entornos gráficos. De entre todas las disponibles, hemos optado por la alternativa de Microsoft, conocida como la **plataforma .NET**, por su amplia difusión y utilización.

La plataforma .NET puede considerarse una respuesta de Microsoft al creciente mercado de los negocios en entornos web, como competencia a la **plataforma Java** de **Sun Microsystems**. Su propuesta es ofrecer una manera rápida y económica de desarrollar aplicaciones.

Dentro de las tecnologías .NET nos encontramos con **Visual Studio .NET**, que ofrece un entorno de desarrollo de alto nivel para desarrollar aplicaciones que se ejecutan sobre el entorno llamado .NET Framework, y soporta varios lenguajes de programación tales como Visual C++, Visual C#, Visual J#, ASP.NET y Visual Basic .NET.

Visual Studio ha pasado por varias versiones, desde la versión 97 y la versión 6 hasta la 2005 y la 2008 actual. Con el objetivo de popularizar el uso de estas herramientas, a partir de la versión 2005 Microsoft ofrece gratuitamente las versiones *Express Edition*. Se trata de varias ediciones básicas separadas por lenguajes de programación o plataformas. Estas ediciones son iguales al entorno de desarrollo comercial, pero sin características avanzadas. Las ediciones disponibles son:

- **Visual Basic Express Edition:** dirigida a aprendices y recién llegados a la programación, así como para desarrolladores con experiencia previa en el uso del lenguaje Visual. El lenguaje de programación que utiliza es Visual Basic .NET.
- **Visual C# Express Edition:** utiliza el lenguaje **C sharp** para la programación de aplicaciones.
- **Visual C++ Express Edition:** la edición que ofrece más potencia y control, aunque como contrapartida puede ser la que requiera más tiempo dominar.
- **Visual J# Express Edition:** utiliza Java para .NET, enfocada a desarrolladores familiarizados con el lenguaje Java y para estudiantes que lo utilizan como base en su formación.
- **Visual Web Developer Express Edition**: para crear aplicaciones web programando en ASP.NET

Adicionalmente, Microsoft ha puesto gratuitamente a disposición de todo el mundo una versión reducida del gestor de base de datos MS SQL Server llamada **SQL Server Express Edition**, cuyas principales limitaciones son que no soporta bases de datos superiores a 4 GB de tamaño, únicamente utiliza un procesador y un GB de RAM.

Para conocer el funcionamiento de una de estas herramientas nos vamos a centrar en **Visual Basic Express Edition**, utilizando como base de datos **SQL Server Express Edition**. La versión que vamos a utilizar de estos programas es la 2005, ya que a la fecha de hoy (Marzo de 2008) la versión 2008 aún no se encuentra en español.

PARA SABER MÁS

La página principal de Microsoft para desarrolladores en Visual Basic .NET te da acceso a multitud de información sobre la programación en Visual Basic, como cursos, comunidades de desarrollo, referencias del lenguaje y aplicaciones de ejemplo:

Centro de Desarrollo de Microsoft Visual Basic .NET

En esta página puedes encontrar información sobre el lenguaje Visual Basic .NET:

Lenguaje Visual Basic

Enlace a la MSDN Library, que es una biblioteca con cantidad de información técnica de programación, incluidos código de ejemplo, documentación, artículos técnicos y guías de referencia:

MSDN Library en Español

Página de foros MSDN para desarrolladores de Microsoft, los foros están organizados en categorías por las distintas herramientas de desarrollo:

Foros MSDN

__Carmen__ y __Víctor__ van a instalar Visual Basic Express 2005 en sus respectivos puestos de trabajo. Al parecer la herramienta no necesita grandes requerimientos, lo cual siempre es una ventaja. El proceso de instalación se reduce a ir siguiendo los pasos de un asistente que les indica lo que ir haciendo en cada paso. Le surgen ciertas dudas sobre si es conveniente instalar la ayuda __MSDN Library__ o mejor acceden a ella desde Internet; al final se deciden por la segunda opción, ya que la documentación ocupa un espacio considerable en disco que pueden destinar a otros fines.

Antes de poder comenzar a escribir aplicaciones en el lenguaje Visual Basic .NET tenemos que instalar en nuestro ordenador las herramientas que van a permitirnos desarrollar aplicaciones en este entorno.

Dentro de estas herramientas nos encontramos con el **.NET FrameWork SDK**, paquete de desarrollo software que funciona en modo comando (MS-DOS) y en modo gráfico, y que es necesario para ejecutar las aplicaciones generadas por este entorno. Aunque podemos ejecutar aplicaciones sólo con .NET FrameWork SDK, utilizar **Visual Basic 2005 Express** facilita la creación de programas al tratarse de un entorno de desarrollo integrado que auna todas las herramientas del SDK: compiladores, editores, ayuda, etc.

Visual Basic Express está disponible para Windows y no requiere de grandes prestaciones en nuestro ordenador, con un equipo Pentium III a 600 Mhz y 192 MB de RAM, recomendables al menos 256 MB, puede funcionar de manera óptima. Cuanta más memoria RAM tengamos se mejorará el rendimiento, especialmente si ejecutamos varias aplicaciones simultáneamente o trabajamos con proyectos de gran tamaño.

Para descargarnos Visual Basic Express entraremos en la página de **Centro de Desarrollo de Microsoft** y desde ahí accederemos a la página de descarga de versiones de **Visual Studio Express Edition**:

Haz clic para ampliar la imagen.

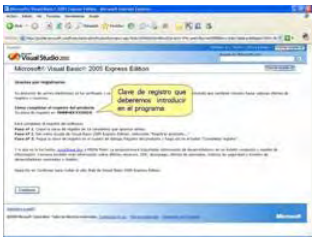
Elegiremos **Abrir/Ejecutar** y, tras copiarse los archivos necesarios al disco duro de nuestro ordenador, comenzará el proceso de descarga e instalación de la herramienta. Haz clic sobre la imagen para ampliarla.

Un asistente nos guiará en el proceso de instalación, durante el cual deberemos aceptar los términos del contrato de licencia. Deberemos decidir si instalamos **MSDN Express Library** y **SQL Server Express Edition** (MSDN es una librería que contiene información sobre todas las versiones de Visual Studio Express, no la vamos a instalar porque accederemos a ella a través de Internet), instalaremos SQL Server Express Edition y deberemos también seleccionar la ubicación del programa. A continuación el asistente comenzará a descargar los siguientes componentes:

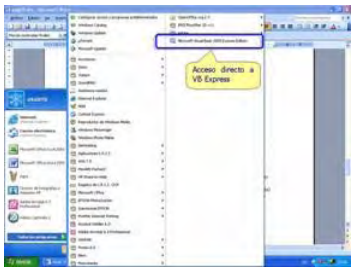
- Microsoft .NET Framework versión 2.0,
- el paquete de idioma español,
- SQL Server y
- Visual Basic 2005 Express.

Tras la descarga procederá a la instalación. Haz clic sobre la imagen para ampliarla.

Seguidamente podremos optar por consultar Windows Update para obtener las actualizaciones más recientes y registrar el producto. Lo primero no lo vamos a hacer porque se supone actualizada la versión que estamos instalando, y el registro del producto puedes hacerlo ahora o después, en todo caso antes de que transcurran 30 días para poder utilizar todas las características del programa.



Si todo es correcto, deberemos tener un acceso a Microsoft Visual Basic 2005 Express Edition en la barra de tareas de Windows.



ZONA DE DESCARGA
Si visitas este enlace podrás acceder a la página de descarga de Visual Basic .NET.
Zona de descarga de Visual Basic 2005 Express Edition

Autoevaluación

La instalación de Visual Basic 2005 Express...

a) Requiere de una máquina con al menos 1GB de RAM
b) Está disponible para Windows y Linux
c) Requiere de Microsoft .NET Framework
d) Todas las respuestas anteriores son correctas

Comprobar

__Víctor__ se va a encargar de comenzar a trabajar con Visual Basic 2005. Lo primero que ha estado investigando es el entorno que ofrece nada más ejecutarla. No parece muy complicado, es una ventaja que se encuentre en español, y los menús y ventanas son muy intuitivos. Cree que no le va a costar hacerse con la herramienta en poco tiempo. Ahora lo primero que tiene que hacer es crear un proyecto sencillo y ver cómo se documenta, se depuran los errores y cómo se realiza el despliegue de la aplicación. Una vez que tenga claros estos conceptos los pondrá en común con __Carmen__.

Visual Basic Express es una herramienta para aprender a programar con el lenguaje **Visual Basic .NET**. Es gratuita y está pensada para uso no profesional, para programadores que no necesitan la versión completa de **Visual Studio .NET.** Haz clic sobre la siguiente imagen para ampliarla.

Cuando ejecutamos Visual Basic Express, nos aparece una ventana cuyos componentes principales puedes apreciar en la figura. Lo primero que aparece es la **página de inicio**, desde la cual podemos abrir proyectos, consultar las noticias de MSDN o acceder a ayuda y tutoriales. También tenemos el explorador de soluciones y el cuadro de herramientas que utilizaremos para crear nuestras aplicaciones.

Lo que vamos a hacer a continuación es crear un proyecto sencillo para ver sobre la marcha las características principales de este entorno. Verás qué rápidamente nos hacemos con la herramienta sin necesidad de conocimientos básicos ni del lenguaje ni del entorno en sí.

Creación de nuestro primer proyecto

Para crear un proyecto podemos hacerlo de las siguientes maneras:

- Menú Archivo / Nuevo Proyecto
- Ctrl + N
- Haciendo clic sobre el icono (Nuevo proyecto) de la Barra de herramientas.
- Haciendo clic sobre Crear: Proyecto de la sección "Proyectos recientes" de la página de inicio.

Haz clic sobre la siguiente imagen para ampliarla.

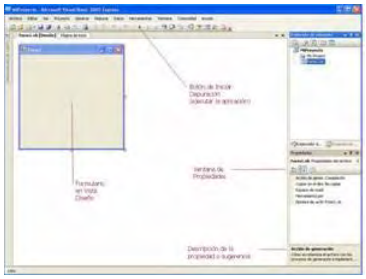
De cualquiera de las formas anteriores, accederemos al asistente de **Nuevo Proyecto**, donde podemos seleccionar una plantilla de aplicación para comenzar a desarrollar nuestro proyecto. Una plantilla de aplicación proporciona archivos de inicio y una estructura de proyecto, y contiene los objetos básicos del proyecto y la configuración del entorno que necesitamos para crear el tipo de aplicación que deseamos.

Las plantillas preinstaladas en Visual Express son:

- **Aplicación para Windows**. Se utiliza con el fin de crear aplicaciones para Windows que se ejecuten localmente en los equipos de los usuarios. Puede generar desde una simple herramienta de una única ventana como la calculadora de Windows hasta una aplicación completa con varias ventanas y funciones avanzadas.
- **Biblioteca de clases**. Se utiliza para crear clases reutilizables o componentes que se pueden compartir con otros proyectos.
- **Aplicación de consola**. Se utiliza para crear aplicaciones de línea de comandos, programas que se ejecutan desde un símbolo de sistema de Windows y no tienen ninguna interfaz visual.
- **Starter Kit de mi colección de películas**. Se utiliza para crear la aplicación de ejemplo pre-integrada Mi cinemateca, que podemos personalizar para satisfacer nuestras propias necesidades.
- **Starter Kit del protector de pantalla**. Se utiliza para crear la aplicación de ejemplo pre-integrada Protector de pantalla, que podemos personalizar.

Para nuestro primer proyecto vamos a elegir la plantilla Aplicación para Windows. Le cambiamos el nombre al proyecto y le llamamos por ejemplo **MiProyecto.**

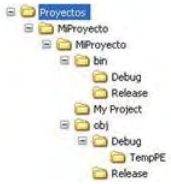
Haz clic sobre la siguiente imagen para poder verla con detalle, ampliada.



En la figura de podemos ver cómo se nos muestra el IDE de Visual Express cuando tenemos un proyecto abierto. Observamos que lo primero que ha hecho ha sido crear un nuevo formulario llamado **Form1.vb**. También se han creado la solución y el proyecto correspondiente a la aplicación. Una solución es un recipiente para proyectos y elementos de solución que pueden incluirse en una aplicación. Normalmente, una solución contiene uno o más proyectos relacionados. Un **proyecto** es un recipiente dentro de una solución que se utiliza para administrar, generar y depurar lógicamente los elementos de proyecto que constituyen nuestra aplicación. En esta unidad nosotros vamos a trabajar con un único proyecto dentro de una solución.

¿Qué debemos hacer a continuación? Pues antes de seguir es recomendable guardar el proyecto. Para ello, seleccionaremos **Menú Archivo / Guardar todo**, o haremos clic sobre el **icono** **(Guardar todo)**. Accederemos al diálogo Guardar proyecto.

En este diálogo podremos cambiar el nombre del proyecto, indicar dónde queremos guardarlo y darle nombre a la solución. Se nos creará la siguiente estructura de directorios:



Principales tipos de archivos que contiene el proyecto

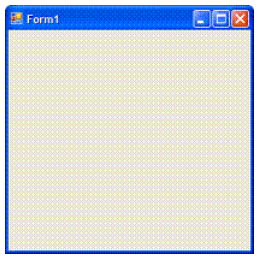
Una imagen vale más que mil palabras, así que presentártelo en forma de tabla es la forma más descriptiva.

Extensión	Nombre	Descripción
.sln	Solución Visual Studio	Organiza proyectos, elementos de proyectos y elementos de soluciones en una solución proporcionando al entorno referencias a sus ubicaciones en disco.
.vb	Proyecto Visual Basic	Representa los archivos de formularios, controles de usuario, clases y módulos que pertenecen a la solución de un solo proyecto.
.vbproj	Proyectos Visual Basic	Representa los archivos de formularios, controles de usuario, clases y módulos que pertenecen a la solución con múltiples proyectos. Esta extensión nos permite diferenciar entre archivos escritos en Visual Basic .NET y otros lenguajes compatibles con .NET. (Visual C# utiliza .csproj.)
.aspx .asmx .asax	Elementos de proyecto Web	Los elementos de proyecto Web incluyen archivos Web específicos como .aspx para Web Forms, .asmx para servicios Web XML, y .asax para clases globales de aplicaciones. Los proyectos Web también utilizan la extensión .vb para clases y módulos.

Ejecutamos el proyecto para verlo funcionar

Para ello, podremos hacerlo de varias formas:

- Menú Depurar / Iniciar Depuración.
- F5
- Icono de la barra de herramientas.



El aspecto de nuestro formulario en ejecución será como el de la ventana Form1. Como podemos comprobar, ya tiene lo básico de cualquier ventana (botón para cerrar, minimizar, maximizar, modificar su tamaño...)

Autoevaluación

La extensión .vb corresponde a...

a) Archivos de formularios, controles, clases y módulos
b) Elementos de proyecto web
c) Archivos de solución de Visual Studio .NET
d) Archivos de proyecto de Visual Studio .NET

Comprobar

Trabajando con controles Windows Form

Dentro del entorno de desarrollo de Visual Basic Express, nos podemos encontrar un completo conjunto de librerías y controles para utilizar en nuestras aplicaciones. Dependiendo de qué tipo de aplicación se trate, el entorno visualizará unos controles u otros.

A continuación vamos a añadir algunos controles a nuestro formulario. Los controles que podemos añadir a nuestro formulario se encuentran en el **Cuadro de Herramientas**, que por defecto, aparece plegado en el borde izquierdo del IDE. Para desplegarlo, sólo tendremos que pasar el puntero del ratón sobre su pestaña, o bien utilizar el botón para dejar fijo el cuadro. Para añadir controles simplemente tendremos que arrastrar sobre el control deseado hacia el formulario. Por ejemplo, podemos añadir un control Label y otro Button que se encuentran dentro de la categoría **Controles Comunes**.

Podemos querer modificar el aspecto de nuestro formulario u otras características de él (o de los controles que hayamos insertado). Para ello haremos uso de la **ventana de Propiedades** del IDE que nos permitirá cambiar el valor de los atributos o propiedades de cualquier componente de nuestro proyecto, como por ejemplo el nombre del objeto (Name) u otras propiedades de un formulario como Text (título de la ventana), ControlBox (si va a tener o no botones de maximizar, minimizar y cerrar) o BackColor (color de fondo).

En la imagen podemos observar los controles añadidos y cómo hemos modificado las propiedades del formulario (el título de la ventana) y de los controles (nombre y texto).

Al ejecutar el formulario observamos que el botón **Aceptar** no tiene asociada ninguna acción. Imaginemos que lo que queremos es que al hacer clic sobre él se muestre el mensaje de "Hola Mundo". ¿Te imaginas cómo hacerlo? Es muy sencillo:

- En la **Vista Diseño** del formulario, hacemos doble clic sobre el botón Aceptar
- Nos aparecerá la ventana de código con el evento asociado al botón
- Utilizamos el objeto **MessageBox** y el método **Show** de Visual Basic .NET para mostrar el mensaje, algo como: MessageBox.Show ("Hola Mundo")
- Si ejecutamos el proyecto comprobaremos que ahora al hacer clic sobre el botón Aceptar nos aparece el mensaje

También podemos configurar nuestro proyecto. Para ello podemos elegir la opción **Propiedades del menú Proyecto** o bien presionar con el botón secundario del ratón sobre el proyecto mostrado en el Explorador de soluciones.

Haz clic sobre la siguiente imagen para verla ampliada a pantalla completa. .

Al seleccionar las propiedades del proyecto, tendremos una nueva ventana desde la que podemos configurar algunas de las características de nuestro proyecto. Podemos apreciar que está dividida según el tipo de configuración que queremos realizar, en este caso concreto las opciones de generación o compilación del proyecto. Como vemos en la figura, existen multitud de opciones y apartados diferentes relacionados todos ellos con nuestra solución.

A continuación tienes una demostración de todos los pasos que hemos seguido para la creación de nuestro proyecto y la adición de algunos controles al formulario que lo compone:

Creación de nuestro primer proyecto

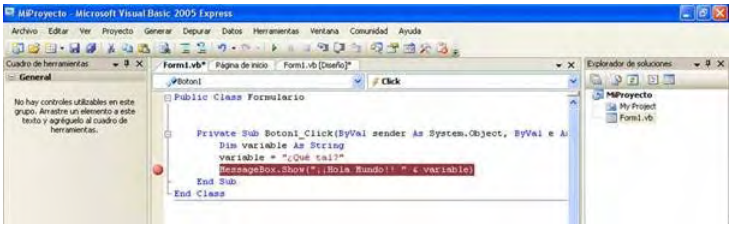
> ***PARA SABER MÁS***
> ***El siguiente enlace proporciona conceptos generales y procedimientos de formularios Windows Form:***
> ***Formularios Windows Form***

Depuración de aplicaciones

Una vez que hemos visto cómo crear un sencillo proyecto en Visual Basic Express, lo siguiente sería pasar al apartado de cómo depurar nuestra aplicación en caso de que presente errores. Por mucho cuidado que tengamos al diseñar un programa o escribir código, siempre pueden aparecer errores. El **IDE** de Visual Basic Express contiene varios comandos y ventanas especiales que ayudan a encontrarlos.

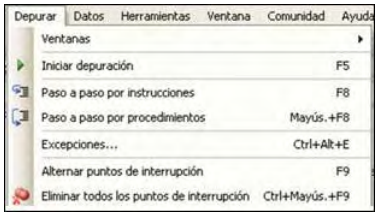
Una forma es colocar un punto de interrupción en el lugar que queramos inspeccionar. Los breakpoints o puntos de interrupción son paradas que permiten hacer una pausa en la ejecución mientras la aplicación está corriendo en modo

depuración, tomando el control de la ejecución del programa línea a línea. Los breakpoints en Visual Basic Express se colocan haciendo clic con el ratón a la izquierda de la sentencia a partir de la cual queramos generar la pausa en la aplicación, haciendo que la línea se quede sombreada.



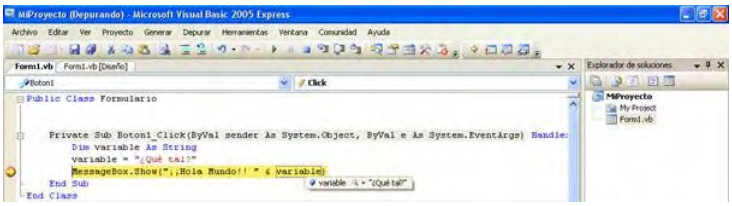
Una vez establecidos los puntos de interrupción en las secciones de código que creemos fallan, para ejecutar el programa en modo de depuración podemos:

- Pulsar **F5**
- Seleccionar el menú **Depurar / Iniciar depuración,** como solemos hacer para ejecutar la aplicación.



Cualquiera de estas acciones iniciará el programa dentro del contexto del depurador, deteniendo la ejecución en la primera línea de código ejecutable, destacada en color rojo. La línea marcada en rojo indica que está a punto de ejecutarse, para ejecutarla y pasar a la siguiente línea pulsaremos **F8** (paso a paso por instrucciones), y así sucesivamente hasta llegar a la última línea del programa, donde se finalizará el mismo, cerrándose el depurador. Si tenemos procedimientos en nuestro proyecto y queremos que el depurador se los salte en lugar de ejecutar el código que contienen, en vez de F8 pulsaremos **Mayus+F8** (paso a paso por procedimientos).

Otra funcionalidad del depurador es que podemos ver de forma inmediata el valor de una variable simplemente situando el cursor del ratón sobre ella, con lo que se mostrará una viñeta informativa de su valor.



Podemos también ver con detalle el valor que van adquiriendo las variables a lo largo de la ejecución, haciendo clic derecho sobre la variable y eligiendo "**Agregar Inspección**", nos aparecerá la ventana de inspección en la parte inferior de la pantalla, que irá mostrando el valor que va tomando la variable según vamos ejecutando el programa.

Si en cualquier momento queremos continuar la ejecución normal del programa sin seguir usando el depurador, elegiremos **Depurar / Eliminar todos los puntos de interrupción** y a continuación pulsaremos **Ctrl.+Mayus+F8** para que la aplicación continúe ejecutándose sin paradas.

A continuación tienes una demostración de cómo depurar nuestra aplicación:

Depuración de aplicaciones

PARA SABER MÁS

Enlace a la información en MSDN Library sobre el manejo de las herramientas de Visual Studio 2005 para la generación, depuración y prueba de aplicaciones:

Generar, Depurar, Probar

Generación de documentación

Otra fase importante en el desarrollo de una aplicación es la generación de documentación sobre la misma. La mejor manera de generar documentación de calidad, es hacerlo mientras programamos. Visual Basic 2005

Express nos permite hacerlo a través de los **comentarios XML**. Para crear un comentario XML tenemos que seguir los siguientes pasos:

- Colocamos el punto de inserción de código en la línea encima de la declaración de la función de la que queremos generar documentación
- Escribimos tres caracteres de comilla simple '''.
- Visual Basic generará una plantilla de comentario XML que corresponde a la declaración de la función
- Rellenamos la plantilla como si fuera un formulario con la documentación de la función.

Por ejemplo, imagina que queremos insertar un comentario XML en nuestro proyecto HolaMundo. El comentario lo vamos a introducir para el único método que tenemos, **Boton1_Click**. La característica de IntelliSense de Visual Basic Express nos proporcionará la plantilla que está compuesta por unas etiquetas entre los signos de menor y mayor y nosotros sólo tendríamos que rellenarla con un texto descriptivo. Algo como lo siguiente:

```
Public Class Formulario

    ''' <summary>

    ''' Controlador de eventos para el evento Click del control Boton1

    ''' Se ejecuta al hacer clic sobre el botón

    ''' Dispone de dos parámetros que permiten controlar correctamente el evento

    ''' </summary>

    ''' <param name="sender">Este parámetro proporciona una referencia

    ''' al objeto que provocó el evento</param>

    ''' <param name="e">Pasa un objeto específico del evento

    ''' que se está controlando</param>

    ''' <remarks>Con los parámetros podemos obtener información

    ''' tal como la posición del ratón o los datos que se están

    ''' transfiriendo en los eventos de arrastrar y colocar</remarks>

    Private Sub Boton1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Boton1.Click

        Dim variable As String

        variable = "¿Qué tal?"

        MessageBox.Show("¡¡Hola Mundo!! " & variable)

    End Sub

End Class
```

Entre las etiquetas XML que podemos utilizar se encuentran las siguientes:

- **<param>.** Se usa para describir los parámetros de un método. Cuando se utiliza, el compilador comprueba si el parámetro existe y si todos los parámetros están descritos en la documentación. Si la comprobación no tiene éxito, el compilador emite una advertencia.
- **<summary>.** Describe un tipo o miembro.
- **<remarks>.** Agrega información sobre un tipo, complementando la información especificada en la etiqueta <summary>.
- **<exception>.** Se aplica a una definición de método y especifica las excepciones que se pueden producir.
- **<returns>.** Describe el valor devuelto por un método.
- **<see>.** Para referenciar a otra clase o método relacionado con el que estamos documentando.

Cuando compilamos nuestro proyecto, se procesan los comentarios de documentación del código y se genera con ellos un **archivo XML**. Este archivo se encuentra en el mismo directorio que el **archivo .exe** de salida o el **archivo .dll** del proyecto. Al tratarse de un archivo XML, se puede transformar con facilidad en otros formatos de resultados según sea necesario.

Consulta en la siguiente demostración cómo generar la documentación para nuestro proyecto "Hola Mundo":

Generar documentación

Puedes acceder al código fuente de MiProyecto a través del siguiente enlace:

Fuentes de MiProyecto

> ***PARA SABER MÁS***
>
> ***En el siguiente enlace tienes información sobre cómo crear automáticamente documentación XML para los proyectos Visual Basic:***
>
> ***Documentar el código con XML***

Autoevaluación

La característica IntelliSense ...

a) Es una característica del editor XML de Visual Basic Express
b) Facilita una plantilla para introducir comentarios XML en el código
c) Introduce etiquetas en el código, como param, summary o remarks

d) Todas las respuestas anteriores son correctas

Comprobar

Despliegue de aplicaciones

En la última fase de desarrollo de nuestra aplicación vamos a ver cómo hacer un despliegue de la misma, para que esté disponible a otros usuarios. Cuando compilamos un programa, el sistema .NET lo compila a un lenguaje intermedio. Al programa resultante en ese lenguaje intermedio se le llama **ensamblado**. Este programa luego es interpretado por el **.NET Framework** de la máquina en la cual lanzamos nuestra aplicación.

A la hora de realizar el despliegue de nuestra aplicación en **Visual Basic Express**, una de las maneras es hacer una copia de los archivos y directorios de nuestro proyecto al equipo cliente donde queremos que se ejecute. Para instalar nuestra aplicación en un sistema cliente, bastará por lo tanto, con realizar una acción similar al **XCOPY** de DOS, es decir, copiaremos en el ordenador cliente, los ficheros o ensamblados necesarios (ejecutables, recursos, dlls, etc.) de la estructura de nuestro proyecto.

Debemos tener en cuenta, que en otros lenguajes, como Visual Basic 6, podíamos hacer esto igualmente, pero debíamos además registrar las DLL con un comando especial para que no hubiera problemas. Aún así, algunas veces nos encontrábamos con algunos contratiempos. Sin embargo, con Visual Basic 2005, esta forma de trabajar ha desaparecido y ahora el despliegue de una aplicación es mucho más sencillo.

Otra opción es utilizar la implementación ClickOnce. Este sistema es un método de distribución de aplicaciones de Microsoft, que permite publicar aplicaciones en un servidor Web o en recurso compartido de archivos de red o en un equipo cliente, para simplificar la instalación. Una vez publicada la aplicación, los usuarios finales pueden descargar e instalar la aplicación haciendo clic en un icono de una página Web o en una carpeta. Además, la aplicación se agrega al menú Inicio de Windows y al grupo Agregar o quitar programas del Panel de control.

Para ver un ejemplo de funcionamiento de esta técnica vamos a publicar la sencilla aplicación que hemos desarrollado en apartados anteriores. Para ello, seguimos los siguientes pasos:

1. Seleccionamos en el menú **Generar / Publicar**.
2. Aparecerá el asistente para publicación, y en la página **¿Dónde desea publicar la aplicación?**, indicamos dónde queremos publicar la aplicación.
3. En la página **Instalación de la aplicación**, seleccionamos la ubicación donde los usuarios instalarán la aplicación: sitio Web o dirección URL, ruta de un recurso compartido de archivos o CD-ROM/DVD-ROM.
4. A continuación podremos decidir si la aplicación va a estar disponible aunque no estemos conectados a Internet, en cuyo caso se creará un acceso directo en el Menú Inicio de Windows.

La máquina donde se instale la aplicación debe disponer de **.NET Framework** para poder interpretar correctamente el **ensamblado**, y cualquier otro programa necesario para ejecutar nuestra aplicación, como por ejemplo el gestor de base de datos para acceder a los datos de la aplicación. En caso contrario, la ejecución de nuestra aplicación, podría no funcionar o provocar algún tipo de excepción o error.

Puedes ver los pasos que hemos seguido para la publicación en la siguiente demostración:

Publicar aplicaciones con ClickOnce

PARA SABER MÁS
Si quieres profundizar sobre la implementación ClickOnce tienes a tu disposición el siguiente enlace de la MSDN Library:
Implementación ClickOnce

__Víctor__ se ha reunido con __Carmen__ para poner en común los avances que ha hecho con Visual Basic Express. Tras intercambiar impresiones, ambos han coincidido en que se trata de un entorno muy adecuado para el aprendizaje y desarrollo de aplicaciones rápidas. También han podido detectar ciertas opciones que no están disponibles en la versión Express y sí en la versión comercial de __Visual Studio .NET__, como por ejemplo la generación de proyectos de instalación. Ahora es el momento de seguir avanzando y ver cómo trata esta plataforma el acceso a datos.

Para entender cómo se realiza el acceso a datos en Visual Basic 2005 Express debemos introducir algunos conceptos teóricos sobre su funcionamiento. Pero no te preocupes que van a ser unas pocas definiciones y enseguida entramos en acción.

El acceso a datos en Visual Basic .NET se hace a través del conjunto de clases denominado **ADO.NET**. Este conjunto de clases nos ofrece un alto nivel de abstracción, ocultando los detalles de bajo nivel de la implementación de la base de datos. Existen varias versiones de ADO .NET, Visual Basic 2005 utiliza la versión 2.0, la cual añade algunas características nuevas adicionales a las anteriores. ADO.NET tiene dos partes importantes:

- El espacio de nombres **System.Data,** que constituye los objetos y clases globales de ADO.NET.
- Los **Proveedores de Acceso a Datos**, que están compuestos por los objetos que permiten el acceso a datos a una determinada fuente de datos desde ADO.NET y que utilizan así mismo, las clases del nombre de espacio **System.Data**. Haz clic sobre la imagen siguiente para ampliarla.

El **Proveedor de acceso a datos** es el que se encarga de la conexión con la base de datos y la ejecución de las instrucciones SQL (acceso a datos). La capa superior que corresponde con el nombre de espacio **System.Data** se encarga de la tarea de recuperar esos datos para tratarlos y manipularlos. El .NET Framework incluye numerosos proveedores de datos .NET, como por ejemplo el proveedor de datos de .NET para SQL Server o el proveedor de datos de .NET Oracle Client.

En la tabla que te mostramos puedes ver representadas las clases que se utilizan en cada espacio de nombres para el acceso a las fuentes de datos.

	Clase	Descripción
System.Data	DataView	Clase que nos permite representar los datos de la clase DataTable, permitiéndonos editar, ordenar y filtrar, buscar y navegar por un conjunto de datos determinado. Este objeto nos permite crear múltiples vistas de nuestros datos, además de permitirnos presentar los datos.
	DataSet	Contiene una representación en memoria de los datos que han sido volcados desde un proveedor de datos. Es una colección de DataTables.
	DataTable	Contiene una representación en memoria de un objeto tabla de la base de datos.
Proveedor de Acceso a Datos	Connection	Establece y gestiona una conexión con una fuente de datos
	Command	Ejecuta un comando en una fuente de datos. Este objeto es el que representa una determinada sentencia SQL o un Stored Procedure (procedimiento almacenado de la BD). Aunque no es obligatorio su uso, en caso de necesitarlo, lo utilizaremos conjuntamente con el objeto DataAdapter que es el encargado de ejecutar la instrucción indicada.
	DataReader	Proporciona un flujo de datos "conectado" desde una fuente de datos. Este objeto es utilizado en una sola dirección de datos, y tiene un acceso a datos muy rápido.
	DataAdapter	Llena un objeto DataSet y actualiza la fuente de datos con los cambios efectuados. Cuando deseamos establecer una comunicación entre una fuente de datos y un DataSet, utilizamos como intermediario a un objeto DataAdapter.

La mayoría de los componentes del proveedor de datos .NET están diseñados explícitamente para la manipulación de entornos desconectados. En estos casos se hace necesario mantener un **DataSet** en memoria con los datos desconectados, que se actualizan con posterioridad en la base de datos. Para aplicaciones de larga ejecución, es a menudo el mejor método. Sin embargo, los desarrolladores de aplicaciones Web generalmente realizan operaciones cortas y sencillas con cada petición, como visualizar datos. Para estas breves operaciones, generalmente no es eficiente mantener un objeto **DataSet**. En tales casos, se utiliza un **DataReader**.

Todos estos tipos de objetos los generará automáticamente el entorno de Visual Basic Express, así que no debes preocuparte de aprenderlos aunque sí hemos creído conveniente introducir una definición de los mismos para poder identificarlos con claridad cuando trabajes con acceso a datos.

Autoevaluación

¿Qué se entiende por ADO .NET?

a) Es una forma de conectarse a bases de datos SQL Server
b) Es el método empleado para el despliegue de aplicaciones en Visual Studio
c) Es el conjunto de clases para la conexión a bases de datos de distintos fabricantes
d) Todas las respuestas anteriores son correctas

Comprobar

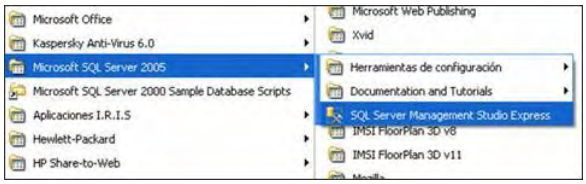
Preparar la base de datos

Las ediciones Express de Visual Studio como **Visual Basic Express** permiten la conexión con bases de datos **Oracle** a nivel de código, o sea, programando y con bases de datos **SQL Server** a través del entorno gráfico. Nosotros vamos a utilizar el entorno gráfico, para no perdernos en aspectos de programación que pudieran alejarnos del objetivo de conocer los aspectos generales de funcionamiento del IDE. El sistema gestor de base de datos que vamos a utilizar por tanto es **SQL Server Express**.

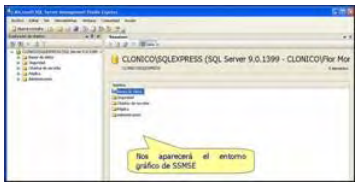
Antes de crear nuestro proyecto debemos preparar la base de datos que vamos a utilizar, lo cual consiste básicamente en cargar los scripts de **Gestión de Taller**. Esto no es necesario si ya disponemos del archivo de la base de datos en SQL Express (archivo con extensión .mdf), pero si no tenemos ese archivo debemos crearlo con los pasos que te vamos a indicar en este apartado.

Lo primero que debemos hacer es instalar **Microsoft SQL Server Management Studio Express (SSMSE),** que es una herramienta gráfica de administración gratuita para SQL Server 2005 Express Edition. La página donde puedes descargarte este programa la encontrarás en un link posterior. No debes tener problemas ya que la instalación es rápida y siguiendo los pasos que indica el asistente.

Una vez instalado **SSMSE**, se habrá creado un acceso directo en el menú Inicio de Windows, dentro del grupo de programas Microsoft SQL Server 2005.



Al abrir SSMSE lo primero que nos pregunta es a qué servidor deseamos conectarnos, normalmente será al **nombreequipo\SQLEXPRESS** y nos lo proporcionará por defecto, así que sólo tenemos que darle al botón conectar y accederemos a la siguiente pantalla (Haz clic sobre ella para ampliarla):



A continuación los pasos que debemos seguir para crear la base de datos son los siguientes:

1. Crear la base de datos haciendo clic con el botón derecho sobre **Bases de Datos** / **Nueva Base de Datos**. Le daremos el nombre de **GestionTaller**.
2. Abrir el script **CrearBD.sql** haciendo clic en el botón 
3. Ejecutamos el script **CrearBD.sql** haciendo clic en el botón 

4. Seguimos los pasos 2 y 3 para ejecutar el script **InsertaDatos.sql**
5. Si lo deseamos podemos crear un diagrama de la base de datos haciendo clic derecho sobre **Diagramas de bases de datos**, donde podremos visualizar gráficamente todas las tablas y las relaciones entre ellas.

Con estos sencillos pasos ya tendremos nuestra base de datos creada en **SQL Server Express**.

Aquí tienes los scripts que debes cargar para crear las tablas e insertar los datos en nuestra base de datos:

Scripts de Gestión Taller

Y la demostración de cómo hemos cargado los scripts en Microsoft SQL Management Studio Express:

Cargar Scripts con SSMSE

> ***ZONA DE DESCARGA***
> ***Si visitas este enlace podrás acceder a la página de descarga de Microsoft SQL Server Management Studio Express (SSMSE).***
> ***Microsoft SQL Server Management Studio Express***

Tecnología Visual Studio .NET

Acceder a los datos. Dataset tipado.

Para acceder a los datos desde **Visual Basic Express** debemos realizar la conexión con el **archivo .mdf** generado en el apartado anterior, que no es otro que el archivo de la base de datos que se ha creado al cargar los scripts en **SQL Server Express**.

En primer lugar lo que debemos hacer es localizar el archivo .mdf. Si lo hemos creado con **SSMSE**, lo podremos encontrar en el directorio donde esté instalado el programa, normalmente en la carpeta MSSQL.1, dentro de ella en MSSQL y dentro de ella en Data. Por ejemplo, si lo hemos instalado en el disco duro C: una ruta válida para nuestro archivo .mdf es la siguiente: **C:\Archivos de programa\Microsoft SQL Server\MSSQL.1\MSSQL\Data**.

También puedes optar por coger el archivo .mdf que se te proporciona al final de este apartado. El archivo .mdf tiene también asociado otro archivo de configuración con extensión **.ldf**.

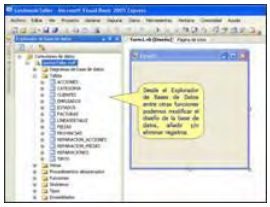
Una vez localizado nuestro archivo de la base de datos podemos realizar la conexión. Los pasos a seguir son los siguientes:

- Creamos un nuevo proyecto llamado **GestiondeTaller** mediante el comando **Archivo / Nuevo Proyecto**.
- Agregamos una conexión a nuestra base de datos haciendo clic en el vínculo **Agregar nuevo origen de datos,** en el panel Orígenes de Datos.
- Seleccionamos Base de Datos como origen de nuestra fuente de datos y le damos a Siguiente.
- En el cuadro **Elija conexión de datos** hacemos clic en el botón **Nueva conexión** y en el nombre del archivo de la base de datos elegimos el archivo **GestionTaller.mdf**, seleccionándolo desde la carpeta donde se encuentre.
- Realizamos la Prueba de la conexión para comprobar que tenemos acceso a la base de datos y le damos a Siguiente.
- Guardamos la **Cadena de Conexión** para poder modificarla posteriormente si fuera necesario. La cadena de conexión es una propiedad del objeto Connection que es utilizado para conectarnos con la base de datos. De esta propiedad depende que nos conectemos correctamente a la fuente de datos.
- Haremos clic en Siguiente. Visual Basic 2005 nos mostrará el siguiente mensaje **"¿Desea copiar el archivo al proyecto y modificar la conexión?"**. Haremos clic en **NO,** ya que deseamos mantener el archivo de base de datos .mdf fuera de nuestro proyecto.
- En el cuadro **Elija los objetos de la base de datos**, seleccionamos las tablas que vamos a necesitar para el formulario que posteriormente elaboraremos.
- Hacemos clic en **Finalizar** y ya tendremos nuestra conexión creada.

Al agregar la conexión con la fuente de datos dentro del **Explorador de soluciones** se ha creado un nuevo archivo llamado **GestionTallerDataSet.xsd**. Este archivo corresponde al **DataSet Tipado** de la conexión que hemos realizado. ¿Y qué es un DataSet Tipado? Pues es un DataSet que posee un esquema de datos, a diferencia del DataSet no tipado, que no necesita de ese esquema. El esquema es un documento XML con extensión .xsd. También observamos que se ha creado un formulario vacío **Form1.vb** al que posteriormente le añadiremos controles.

¿Qué ventajas nos aportan los DataSet tipados? Muchas, ya que a través de los DataSets tipados, podemos realizar acciones de manera rápida y sencilla y con menos código, como actualización de datos, modificación, inserción o eliminación de datos. Posteriormente veremos cómo utilizar el Dataset tipado para generar nuestro formulario. Haz clic sobre la siguiente imagen para ampliarla.

Al agregar la conexión a la fuente de datos, otro elemento que se nos ha añadido al proyecto ha sido una nueva conexión en el **Explorador de Bases de Datos**. Haz clic sobre la siguiente imagen para ampliarla.



Esta nueva conexión nos facilita el trabajo con la base de datos directamente, de forma que desde el **Explorador de Bases de Datos**, haciendo clic con el botón derecho sobre los objetos que contiene, podemos:

- Modificar el diseño de una tabla mediante el comando **Abrir definición de tabla.**
- Ver los datos de una tabla mediante el comando **Mostrar datos de tabla.**
- Introducir nuevos datos en las tablas mediante el comando **Mostrar datos de tabla.**

Aprovecharemos entonces para añadir algunos registros más a las tablas de nuestra base de datos para que se muestren en el formulario que posteriormente crearemos.

¿Qué te ha parecido el proceso de acceso a datos? No ha sido muy difícil ¿no?

Para finalizar aquí tienes el archivo .mdf de la base de datos que has debido generar al cargar los scripts en el apartado anterior:

BD Gestión de Taller

Y en esta demostración puedes ver todos los pasos que hemos seguido para crear la conexión a la base de datos Gestión de Taller:

Acceso a Datos con ADO .NET

> ***PARA SABER MÁS***
>
> ***En el siguiente enlace puedes encontrar vínculos a páginas que tratan sobre el acceso a datos de aplicaciones, tutoriales y toda la información relacionada con el acceso a datos en Visual Basic .NET:***
>
> ***Obtener acceso a datos en aplicaciones Visual Basic***

Creación de un formulario maestro-detalle

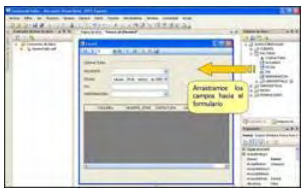
Una vez que tenemos creada la conexión con la base de datos, podemos crear formularios que accedan a esos datos de una manera muy fácil y sin apenas introducir código. Vamos a comprobar cómo de fácil resulta esta tarea creando un formulario maestro-detalle en nuestro proyecto **Gestión de Taller**.

En el apartado anterior, al crear el proyecto, hemos podido comprobar que se añadía un formulario vacío **Form1.vb**. Antes de añadir los campos al formulario, debemos configurar

cómo queremos que aparezcan. Para ello, en el panel de **Orígenes de Datos** tenemos a la derecha de cada campo una la lista desplegable mediante la cual podemos seleccionar cómo queremos que ese control se añada al formulario: si queremos que se añada como una etiqueta (**Label**), o como una lista desplegable (**combobox**), y cuando se trata de un conjunto de controles podemos elegir si queremos que se vean en formato tabular (**DataGridView**) o bien con campos separados (**Detalles**). La configuración que vamos a elegir para nuestro formulario te la mostramos en la tabla. A la derecha tienes una columna de observaciones donde encontrarás mayor aclaración sobre cada configuración elegida.

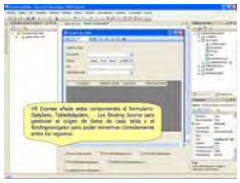
Objeto	Tipo de control	Observaciones
Facturas	Detalles	Queremos que cada uno de los campos de Facturas se añadan uno a uno (formato de columnas). Sólo se mostrará un registro de Facturas por pantalla.
CodFactura	Label	CodFactura va a ser un campo que no va a poder modificarse, por eso lo añadimos como Label.
Idcliente	Combobox	Será una lista desplegable de clientes.
Fecha	DateTimerPicker	Permitirá seleccionar la fecha de un calendario.
IVA	TextBox	Un simple cuadro de texto.
IdReparacion	ComboBox	Una lista desplegable con las reparaciones existentes.
LineasDetalle	DataGridView	Mostrará en formato tabular para una factura dada, cada una de las líneas de detalle que le correspondan. Se visualizarán varias líneas de detalle en una sola pantalla.

Una vez que hemos configurado cómo queremos que aparezcan los campos en nuestro formulario, tan sólo tenemos que arrastrar dichos campos desde el panel **Orígenes de Datos** hacia el formulario.



Los campos arrastrados se agregarán al formulario según la configuración elegida en el panel de Orígenes de Datos. A continuación vamos a modificar algunos aspectos de estos controles. En las líneas de detalle vamos a eliminar el campo **CODFACTURA** porque está repetido con el mismo campo que aparece en la cabecera de la factura. Esto lo hacemos editando las columnas del **DataGridView** que hemos creado para **Líneas de Detalle**, haciendo clic en la especie de triángulo que aparece en la esquina superior derecha. (Haz clic sobre las siguientes imágenes para ampliarlas).

Durante todo este proceso **Visual Basic Express** ha creado por nosotros los componentes que necesita para acceder a los datos. Estos componentes los puedes encontrar en la parte inferior de la pantalla cuando estamos diseñando el formulario como por ejemplo los **DataSets** y los **TableAdapter** para rellenar los datos. También vemos que crea dos componentes nuevos como son los **BindingSource** para gestionar el origen de datos de las tablas y el **BindingNavigator** para poder movernos cómodamente entre los registros.



Antes de continuar, observa que hemos hecho un par de cambios al formulario que han sido ponerle un nombre más adecuado, como por ejemplo **"Gestión de Taller",** modificando la propiedad **Text,** y al campo CodFactura le hemos aplicado un borde en 3D mediante la propiedad **BorderStyle**.

A continuación tienes una demostración de todos los pasos seguidos para crear el formulario maestro-detalle de Facturas:

Creación de un formulario maestro-detalle

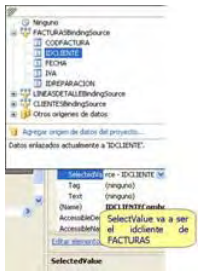
Creación de listas desplegables y campos calculados

Una vez confeccionado el formulario maestro-detalle hemos de realizar ciertos ajustes de formato y configuración en la interfaz gráfica. Observa que en lugar de datos significativos del cliente o de la reparación, actualmente se nos muestran campos numéricos que no nos dicen nada y que denotan una interfaz poco amigable hacia el usuario final. Lo ideal sería mostrar el nombre del cliente y el diagnóstico de la reparación, por ejemplo, en lugar

de códigos numéricos, lo cual daría mucha más información al usuario final para poder elegir entre los distintos valores. Para ello debemos crear una lista desplegable de Clientes y otra de Reparaciones.

Para crear la lista desplegable de Clientes seleccionamos el control en el formulario y establecemos las siguientes propiedades del control:

- **DataSource:** elegimos **CLIENTES**, que se encuentra dentro del DataSet que hemos creado en pasos anteriores **GestionTallerDataSet**
- **DisplayMember:** indicamos que de todos los campos de clientes, el que queremos que se muestre en la lista desplegable es el campo **Nombre**.
- **ValueMember:** seleccionamos cuál es el campo que va a guardar la lista desplegable, en este caso **idcliente**.
- **DataBinding/SelectValue:** introducimos el campo **idcliente** de **FACTURAS**. Este campo se vinculará con el **idcliente** introducido en **ValueMember**. Haz clic sobre cada una de las siguientes imágenes para ampliarlas.



Para crear la lista desplegable de Reparaciones vamos a hacer una modificación en el DataSet que va a consistir en crear un nuevo campo compuesto del **diagnóstico** de la reparación y del **idcliente** de esa reparación. Estos cambios sólo afectan al Dataset y no se ven reflejados en la base de datos. Hacer esto es muy sencillo.

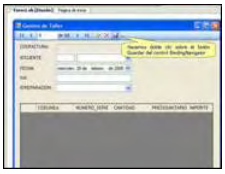
- Nos vamos al **Explorador de Soluciones** y
- hacemos doble clic sobre **GestionTallerDataSet**.

Nos aparecerán todos los **DataComponent**, el que queremos modificar es **Reparaciones**.

- Hacemos clic derecho sobre él y elegimos el comando **Agregar / Columna.**
- A la nueva columna le llamamos **REPARACION**, y lo único que tenemos que hacer es en la ventana Propiedades, en la propiedad **Expresión**, introducir lo siguiente:
  - **DIAGNOSTICO + ' (cliente ' + IDCLIENTE + ')'**.
- Esto creará una nueva columna dentro del DataComponent que mostrará el diagnóstico de la reparación más el **idcliente** al que se le ha realizado.

Ahora ya podemos seguir los mismos pasos que en el caso anterior para crear la lista desplegable de Reparaciones:

- **DataSource:** elegimos **REPARACIONES**, que se encuentra dentro del DataSet que hemos creado en pasos anteriores
- **DisplayMember:** campo que va a mostrar la lista desplegable: columna **Reparacion**,.
- **ValueMember:** campo que va a guardar la lista desplegable: **idreparacion**.
- **DataBinding/SelectValue:** introducimos el campo **idreparacion** de **REPARACIONES**. Este campo se vinculará con el **idreparacion** introducido en **ValueMember**.



De la misma forma que hemos hecho para agregar la nueva columna en REPARACIONES, podemos hacer para agregar una nueva columna en el DataComponent **LINEASDETALLE**, que se llame Importe y que como expresión tenga **CANTIDAD * PRECIOUNITARIO**, y como tipo de datos (propiedad **DataType**) debemos indicarle que sea **System.Single** que representa un número de punto flotante de precisión simple.

Finalmente, lo único que nos queda es agregar unas líneas de código para que el formulario guarde los cambios y actualice la base de datos para que la información perdure en el tiempo. Visual Basic Express genera la mayoría del código por nosotros, tan sólo tenemos que añadir unas líneas para que se guarden los cambios realizados en el control **DataGridView** de la tabla **LINEASDETALLE**. Para ello hacemos doble clic sobre el botón Guardar del control **BindingNavigator** y accederemos al código que responde al evento de hacer clic sobre dicho botón, (haz clic sobre la imagen de la derecha para ampliarla y ver el botón al que nos referimos) donde añadiremos las siguientes líneas:

```
Me.LINEASDETALLEBindingSource.EndEdit() 'Finalizamos la edición del control

        Me.LINEASDETALLETableAdapter.Update(Me.GestionTallerDataSet.LINEASDETALLE) 'Actualizamos el

        TableAdapter asociado con LINEASDETALLE
```

A continuación tienes una demostración de todos los pasos seguidos para crear las listas desplegables de clientes y reparaciones:

Creación de listas desplegables

Para crear el campo calculado Importe en el DataComponent LINEASDETALLE y posteriormente agregarlo en el DataGrid del formulario hemos seguido los pasos que te indicamos en la siguiente demostración. Al final de esta demostración también tienes el código que es necesario añadir para que los cambios hechos en el DataGrid se actualicen en la base de datos:

Creación de campos calculados

Autoevaluación

¿Cómo se crea un formulario maestro-detalle en VB Express?

a) Hay que crear el formulario, arrastrar los campos e introducir el código correspondiente para navegar por los registros, ya que el IDE no lo hace automáticamente
b) Tan sólo debemos crear el formulario y arrastrar los campos porque todo lo demás lo hace el IDE automáticamente
c) Crear el formulario, arrastrar los campos desde el panel de Orígenes de datos e introducir el código para guardar los datos de la sección de detalle
d) Todas las respuestas anteriores son correctas

Comprobar

*Una vez creado el enlace de la aplicación con los datos, **Carmen** le comenta a **Víctor** que lo que quedaría sería dotar a la aplicación de una interfaz gráfica que le diera un aspecto más uniforme al conjunto. **Carmen** ha leído que la creación de formularios hijos que se encuadren dentro de un formulario principal es algo bastante sencillo en Visual Basic Express, de hecho ella misma lo ha podido comprobar y le da unas breves pinceladas a **Víctor.***

Para finalizar con la creación de nuestra aplicación, nos faltan unos sencillos pasos para conformar lo que va a ser el interfaz gráfica, que se reducen a crear una pantalla de bienvenida y un formulario principal o formulario MDI que contendrá el resto de formularios o ventanas de dicha aplicación.

Lo primero que vamos a hacer es crear la pantalla de bienvenida y seguidamente continuaremos con la creación del formulario principal, el cual enlazará con el formulario maestro-detalle creado en los apartados anteriores.

Pantalla de bienvenida

Visual Basic Express Edition proporciona una forma fácil y rápida de crear una pantalla de bienvenida, tan sólo tenemos que crear el formulario y posteriormente configurar el proyecto indicándole que ese formulario va a ser la pantalla de bienvenida de nuestra aplicación. Esos simples pasos nos van a permitir que al ejecutar nuestra aplicación antes de que aparezca el formulario principal, se muestre una pantalla con la información que queramos, como puede ser el nombre de la aplicación, el número de revisión e información sobre el copyright.

Hacer esto es muy sencillo, tan sólo tenemos que seguir los pasos siguientes:

- Desde el Explorador de Base de Datos, hacemos clic derecho sobre el nombre del proyecto y elegimos el comando **Agregar / Nuevo Elemento ...**
- De entre las que nos propone Visual Basic Express, utilizamos la plantilla **"Pantalla de Bienvenida"**
- Hacemos clic sobre **Agregar** y ya tendremos creado nuestro formulario de bienvenida

En el Explorador de Soluciones aparecerá el nuevo formulario creado. Ahora sólo tenemos que decirle a la aplicación que utilice ese formulario como pantalla de bienvenida, en las opciones de configuración del proyecto, dentro del menú **Proyecto / Propiedades de ... / Pantalla de Bienvenida.**

El último paso en este apartado va a ser cambiar el nombre del formulario **Form1.vb** por **Facturas.vb**, esto lo puedes hacer simplemente haciendo clic con el botón derecho sobre el formulario en el **Explorador de Soluciones** y eligiendo el comando **Cambiar nombre** del menú contextual. Cuando cambias el nombre al archivo, Visual Basic Express también cambia el nombre de la clase que lo define.

Si ejecutas la aplicación comprobarás que puedes visualizar esa pantalla de bienvenida que acabamos de crear, para posteriormente mostrar el formulario Facturas. En la siguiente demostración tienes todos los pasos que hemos realizado:

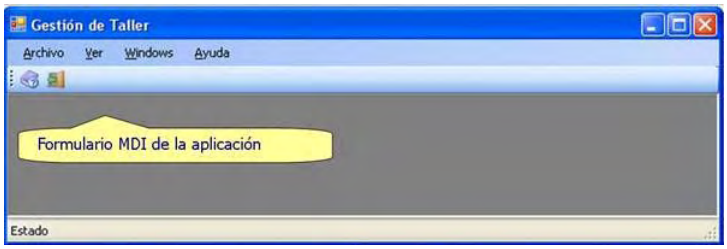
Pantalla de Bienvenida

Formularios MDI

Una aplicación de tipo **MDI (Multiple Document Interface: Interfaz de Documento Múltiple)** se compone de un formulario principal que actúa como contenedor de otros formularios (ventanas) abiertos durante la ejecución del programa. Normalmente, el formulario principal o formulario MDI tiene los controles estrictamente necesarios, como un

menú, una barra de herramientas y una barra de estado, por ejemplo.

Las aplicaciones que están compuestas por un solo formulario, o por varios pero que se abren cada uno de manera independiente reciben el nombre de aplicaciones SDI **(Single Document Interface: Interfaz de Documento Sencillo)**



Visual Basic Express dispone de una plantilla para formularios MDI, por lo que la tarea de crear un formulario de este tipo se reduce a los siguientes pasos:

- Desde el Explorador de Base de Datos, hacemos clic derecho sobre el nombre del proyecto y elegimos el comando **Agregar / Nuevo Elemento ...**
- De entre las que nos propone Visual Basic Express, utilizamos la plantilla **"Formulario MDI"**
- Hacemos clic sobre **Agregar** y ya tendremos nuestro formulario principal creado

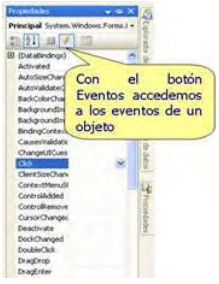
Vemos que nos aparece un formulario con todas las opciones típicas de la ventana principal de una aplicación. De entre todas ellas sólo vamos a utilizar unas pocas y crear otras, las demás las borramos. El formulario lo vamos a dejar con una barra de menú, una barra de herramientas y la barra de estado.

Las opciones que va a tener cada menú son las que aparecen en la imagen. Como puedes observar dentro del menú **Archivo** tenemos un elemento de menú para acceder a **Facturación** y dentro de **Ayuda** otro elemento para acceder a la información **Acerca de...** de la aplicación. Los menús **Windows** y **Ver**, así como su código asociado, se han creado automáticamente al generar el formulario MDI. También tenemos los botones de la barra de herramientas. Cada uno de estos elementos tiene un código en Visual Basic .NET. Se trata de unas cuantas líneas de código y lo único que hay que hacer es doble clic sobre el elemento en la ventana de diseño del formulario, dando paso al código del formulario y, concretamente, a un método que se creará vacío con los argumentos o parámetros necesarios para responder al evento predeterminado (**hacer clic**). Introducir código para responder a otro tipo de evento es muy fácil, tan sólo elegimos el evento apropiado en la ventana de propiedades del objeto en cuestión. Haciendo clic sobre el botón podremos acceder a todos los eventos disponibles para el formulario o control seleccionado.

En la siguiente tabla tienes detallado el código de cada control creado por nosotros, siempre para el evento predeterminado de hacer clic sobre el objeto.

Elemento	Código
Botón Salir	Global.System.Windows.Forms.Application.Exit()
Botón Ayuda	Me.AboutToolStripMenuItem_Click(sender, e)
Menú Acerca de...	Dim ChildForm As New Acercade ChildForm.MdiParent = Me m_ChildFormNumber += 1 ChildForm.Text = "Acerca de... " & m_ChildFormNumber ChildForm.Show()
Menú Facturación	Dim ChildForm As New Facturas ChildForm.MdiParent = Me ChildForm.Text = "Facturación" ChildForm.Show()

Si te das cuenta, todo el código empleado está basado en el que ya había generado Visual Basic Express automáticamente para el resto de elementos de menú y controles, cambiando los valores que nos interesan. Para el botón **Salir** llamamos al método **Exit** de la clase **Application**. En el caso del botón **Ayuda** lo que hacemos es ejecutar el método que responde al evento hacer clic del menú **Acerca de**, cuyo código aparece en la fila siguiente y lo único que hace es crear un formulario hijo que tenga como padre el formulario principal, le cambia la propiedad **Text** y lo muestra, igual para el menú **Facturación**. En definitiva, no hemos tenido que introducir código de nuestra cosecha, esto nos lleva a descubrir que con unas nociones básicas podemos crear una aplicación Windows de forma rápida y sencilla.



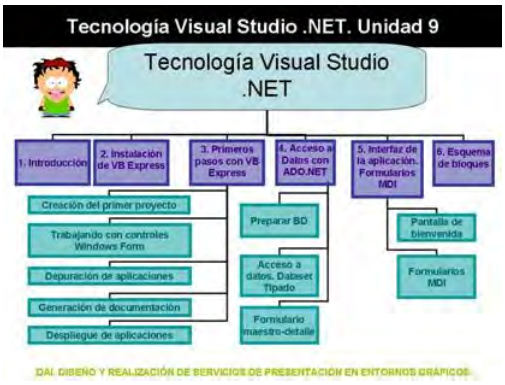
Los pasos seguidos para crear el formulario MDI los tienes en la siguiente demostración:

Formulario MDI

A continuación te facilitamos los archivos de la aplicación publicados con ClicOnce, recuerda que sólo tienes que ejecutar el archivo **setup.exe** para instalar la aplicación. El código fuente de la aplicación también lo tienes disponible en el siguiente enlace. La aplicación accede a la base de datos **C:/net/BD/GestionTaller.mdf** (facilitada en un recurso anterior):

Aplicación Gestión de Taller

Aplicación Gestión de Taller (fuentes)


Tecnología Visual Studio .NET. Unidad 9
Tecnología Visual Studio .NET
1. Introducción
2. Instalación de VB Express
3. Primeros pasos con VB Express
4. Acceso a Datos con ADO.NET
5. Interfaz de la aplicación. Formularios MDI
6. Esquema de bloques
Creación del primer proyecto
Trabajando con controles Windows Form
Depuración de aplicaciones
Generación de documentación
Despliegue de aplicaciones
Preparar BD
Acceso a datos. Dataset Tipado
Formulario maestro-detalle
Pantalla de bienvenida
Formularios MDI
DAI. DISEÑO Y REALIZACIÓN DE SERVICIOS DE PRESENTACIÓN EN ENTORNOS GRÁFICOS