



Desarrollo de Aplicaciones Móviles II

ÍNDICE

Presentación	5
Red de contenidos	6

Unidad de aprendizaje 1 Diseño de Aplicaciones

SEMANA 1	: Introducción a la .NET Compact Framework y Windows Mobile	7
SEMANA 2	: Desarrollo de aplicaciones con Visual Studio 2008 – Manejo de entorno	31
SEMANA 3	: Uso de controles	39
SEMANA 4	: Construcción de interfaces	49
SEMANA 5	: Creación de controles	69
SEMANA 6	: Conectividad a redes	79

Unidad de aprendizaje 2 Acceso a datos

SEMANA 7	Semana de exámenes parciales	
SEMANA 8	Semana de exámenes parciales - laboratorio	
SEMANA 9	: Introducción al modelo de datos - SQL Server CE	91
SEMANA 10	: ADO.NET en Compact framework 1	99
SEMANA 11	: ADO.NET en Compact framework 2	115
SEMANA 12	: Remote data Access	147
SEMANA 13	: Sincronización de Merge-Replication con SQL Server de escritorio	163

Unidad de aprendizaje 3 Empaquetado y despliegue de aplicaciones móviles

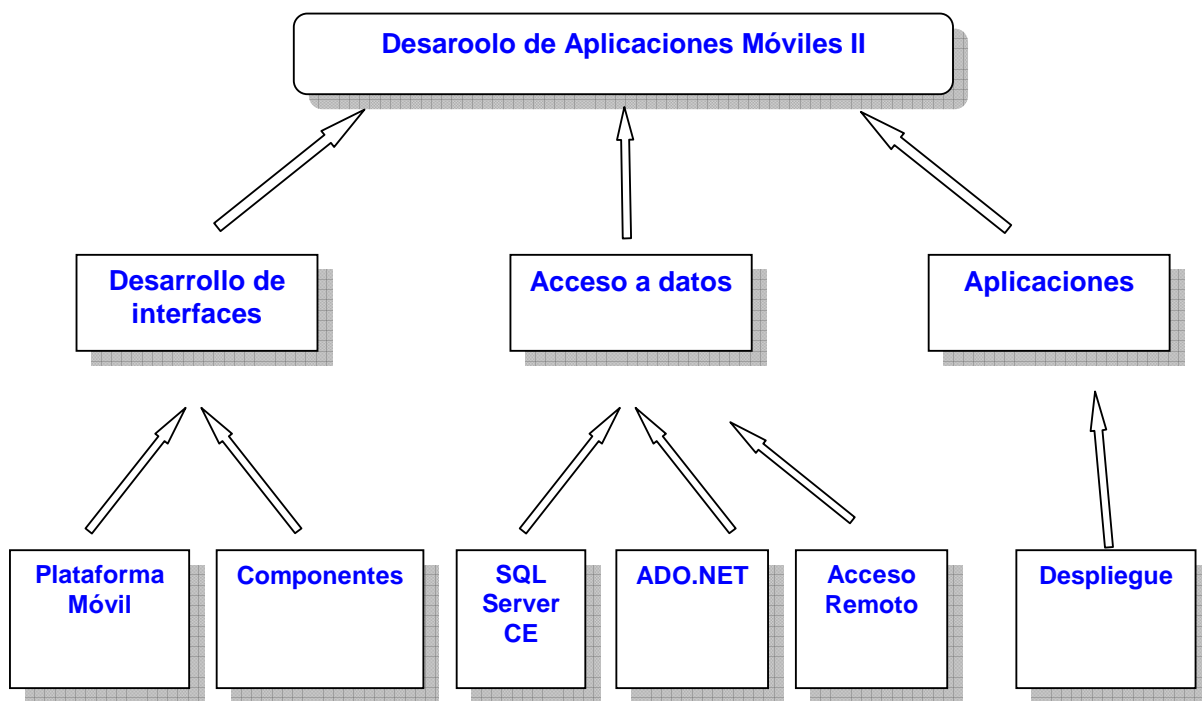
SEMANA 14	: Empaquetado y desplegado de aplicaciones	169
SEMANA 15	Semana de exámenes finales	

Presentación

El presente manual tiene como fin dar a conocer al estudiante, en la primera mitad del mismo, los elementos que comprende las API's para el desarrollo de aplicaciones móviles, que le permitirán generar aplicaciones más avanzadas dentro del desarrollo de aplicaciones móviles con el .NET Compact Framework, empezando por las nociones básicas acerca del .NET Compact Framework y la herramienta Visual Studio, siguiendo con el manejo de controles y su personalización; luego el desarrollo de interfaces de usuario ; y se finalizara con los conceptos de conectividad a redes.

Para la segunda mitad del curso, se abocará a la explicación del funcionamiento de ADO.NET, que permite el manejo de información de una base datos; se dará una explicación detallada del funcionamiento del Acceso Remoto a Datos (RDA), conceptos de la sincronización de datos con un servidor y si replicación; y se finalizará con la elaboración de un proyecto de instalación de una solución móvil.

RED DE CONTENIDOS



**UNIDAD DE
APRENDIZAJE****1****SEMANA****1**

Introducción al .NET Compact Framework y Windows Mobile

OBJETIVOS ESPECÍFICOS

- Conocer las características fundamentales del .NET Compact Framework y Windows Mobile

CONTENIDOS

- Compact Framework
- Arquitectura de Compact Framework.
- Manejo de la memoria de los dispositivos.
- Diferencias con .NET Framework
- Conceptos de hardware a tener en cuenta.
- Lo nuevo en .NET Compact Framework 3.5.

1. EL Compact Framework

.NET Compact Framework es un entorno independiente del hardware para la ejecución de programas en dispositivos de computación con limitaciones de recursos, entre los que se encuentran los asistentes de datos personales (PDA) como Pocket PC, teléfonos móviles, decodificadores de televisión, dispositivos de computación para automóviles y dispositivos incrustados de diseño personalizado, que están integrados en el sistema operativo Windows CE .NET

.NET Compact Framework es un subconjunto de la biblioteca de clases .NET Framework y también contiene clases diseñadas expresamente para él. Hereda la arquitectura .NET Framework completa de Common Language Runtime y la ejecución de código administrado.

.NET Compact Framework ofrece las siguientes funciones principales:

- Ejecuta programas independientes del hardware y el sistema operativo.
- Admite protocolos de red comunes y se conecta perfectamente con servicios XML Web.
- Proporciona a los desarrolladores un modelo para orientar sus aplicaciones y componentes ya sea a una amplia gama de dispositivos o a una categoría específica de éstos.
- Facilita el diseño y la optimización de los recursos de sistema limitados.
- Obtiene un rendimiento óptimo en la generación de código nativo cuando se utiliza compilación Just-In-Time (JIT).

.NET Compact Framework no admite las siguientes tecnologías:

- Funcionalidad de servidor
- ASP.NET
- Entorno remoto
- Emisión de la reflexión

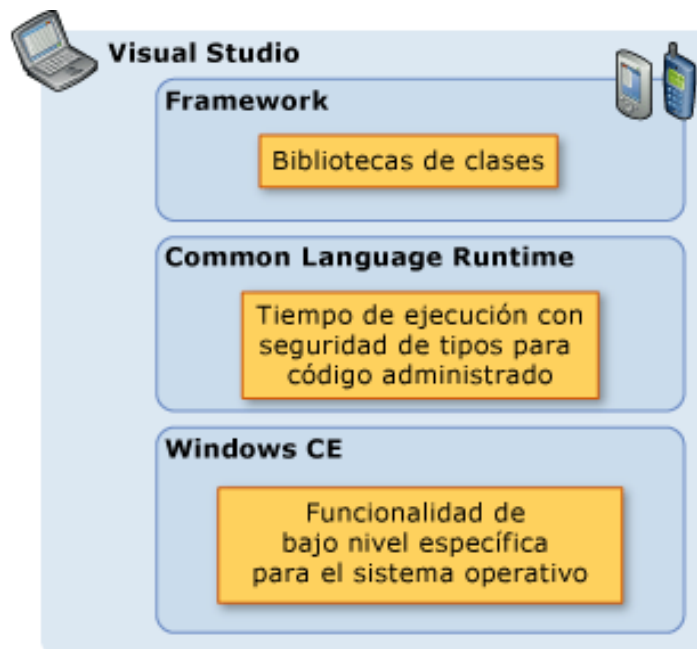
- Desarrollo en C++
- Desarrollo en J# y JSL

2. Arquitectura de .NET Compact Framework

.NET Compact Framework hereda la arquitectura .NET Framework completa de Common Language Runtime para ejecutar código administrado. Proporciona interoperabilidad con el sistema operativo Windows CE de un dispositivo para tener acceso a funciones nativas e integrar los componentes nativos favoritos en una aplicación.

Puede ejecutar aplicaciones nativas y administradas de manera simultánea. El host del dominio de aplicación, que también es una aplicación nativa, inicia una instancia del Common Language Runtime para ejecutar el código administrado.

En la ilustración siguiente se resume la arquitectura de la plataforma .NET Compact Framework.



2.1 Windows CE

.NET Compact Framework utiliza el sistema operativo Windows CE para la funcionalidad central y para diversas características específicas de dispositivos. Varios tipos y ensamblados, como los de los formularios Windows Forms, gráficos, dibujos y servicios Web, se han recompilado para que se ejecuten eficazmente en los dispositivos, en lugar de copiarse de .NET Framework completo.

Windows CE es el sistema operativo de Microsoft incrustado modular de tiempo real para dispositivos móviles de 32-bits inteligentes y conectados. Windows CE combina la compatibilidad y los ping a servicios de aplicación avanzados de Windows con soporte para múltiples arquitecturas de CPU y opciones incluidas de comunicación y redes para proporcionar una fundación abierta para crear una variedad de productos. Windows CE impulsa a los dispositivos electrónicos del cliente, terminales Web, dispositivos de acceso a Internet, controladores industriales especializados, computadoras de bolsillo, dispositivos de comunicación incrustados e incluso consolas de video juegos como fue en el caso de la Sega Dreamcast (1997 - 2001) con procesador SH4 de 128 Bits que ya con un sistema operativo propio, incluía compatibilidad con los kits para desarrollo de software de Windows CE.

Esta plataforma modular permite a los desarrolladores crear software para que la nueva generación de dispositivos móviles de 32-bits se integre con Windows e Internet.

Windows CE no es un subconjunto de Windows XP, o de Windows NT, sino que fue desarrollado a base de nuevas arquitecturas y una nueva plataforma de desarrollo. Aun así mantiene cierta conexión con sus hermanos. Windows CE tiene sus propias APIs para desarrollo, y necesita sus propios drivers para el hardware con el cual va a interactuar. Windows CE no es un sinónimo de Windows XP en forma pequeña, incrustada o modular.

Windows CE también ha permitido la creación de un sistema denominado AutoPC, que consiste en un PC empotrado en un automóvil

y que va ubicado donde normalmente va una radio. De esta manera permite controlar la radio, el reproductor de CD y revisar el correo electrónico. Windows CE también permite la creación de aplicaciones en tiempo real.

La última versión del Windows CE actualmente es Windows Mobile 6.1, Upgrade de Windows Mobile 6.0, sucesor de Windows Mobile 5.0, y sirve tanto para Pocket PC (PDA) como para SmartPhone.

Cabe destacar que este sistema operativo es el **único** producto de Microsoft que se distribuye junto con el código fuente (tal como Linux) y usa una licencia llamada Shared Source, así pues permite al usuario final modificar el código fuente sin notificar al propietario.

.NET Compact Framework ofrece la siguiente interoperabilidad con Windows CE:

- Compatibilidad con seguridad nativa.
- Integración completa con programas de instalación nativos.
- Interoperabilidad con código nativo mediante la interoperabilidad COM y la invocación de plataformas.

2.2 Common Language Runtime

También el Common Language Runtime (CLR) de .NET Compact Framework se ha vuelto a generar para permitir que los recursos restringidos se ejecuten en memoria limitada y lograr un uso eficaz de la energía.

Entre Windows CE y el Common Language Runtime existe una capa de adaptación de plataforma, que no aparece en la ilustración, para asignar las interfaces de servicios y dispositivos necesarias para CLR y Framework a los servicios e interfaces de Windows CE.

2.3 Framework

.NET Compact Framework es un subconjunto de .NET Framework pero también contiene características diseñadas en exclusiva. Ofrece prestaciones y facilidad de uso para acercar a los desarrolladores de aplicaciones nativas para dispositivos a .NET Framework, y para acercar a quienes desarrollan aplicaciones de escritorio a los dispositivos.

3. Administración de memoria de dispositivos

Una importante propiedad de .NET Compact Framework es su eficiente utilización de los recursos, en especial de la RAM volátil. No es necesario que los dispositivos tengan unidades de administración de memoria (MMU) de hardware ni memoria virtual del sistema operativo.

.NET Compact Framework hace un uso cuidadoso de la memoria libre del sistema. No se realiza un acceso a la RAM hasta que se ejecuta una aplicación. Además, .NET Compact Framework libera la RAM al finalizar los programas. No se requiere que el sistema operativo nativo tenga sus propias funciones de protección de la memoria. Siempre se produce una excepción cuando se obtiene acceso a memoria que no tiene propietario.

Si la memoria es escasa, .NET Compact Framework adopta una estrategia agresiva de liberación de las estructuras de datos internos que no son necesarias para el código que se está ejecutando. Por lo tanto, puede seguir ejecutándose el programa, incluso en situaciones de escasez de memoria. Si la aplicación requiere más memoria de la que tiene a su disposición, .NET Compact Framework la cierra limpiamente y libera todos los recursos subyacentes. El propio .NET Compact Framework no debería causar errores a causa de memoria insuficiente.

El host del dominio de aplicación inicia las aplicaciones de .NET Compact Framework y el Common Language Runtime. Las aplicaciones de .NET Compact Framework utilizan el espacio de código y el espacio de datos dinámicos y estáticos de la misma manera que las aplicaciones nativas. Mientras no se ejecutan aplicaciones de .NET Compact Framework, no se

ocupa más memoria RAM que la del host del dominio de aplicación, y se utiliza una pequeña cantidad de datos estáticos para el Common Language Runtime. Windows CE .NET crea un host del dominio de aplicación cuando se inicia una aplicación de .NET Compact Framework.

Las aplicaciones de .NET Compact Framework están empaquetadas en archivos .exe y .dll, que pueden almacenarse en un sistema de archivos de sólo lectura o de lectura/escritura en la memoria flash (o en la ROM para sólo lectura). El cargador de clases de Common Language Runtime puede leer estos archivos en bloques direccionables sin crear una copia en la memoria y sin necesidad de una unidad de administración de la memoria para crear una vista del archivo asignada en memoria.

Se anima a los desarrolladores a probar sus aplicaciones en varios dispositivos, para comprender mejor las variaciones de rendimiento específicas de cada dispositivo.

3.1 Almacenamiento en la RAM

La memoria de acceso aleatorio (RAM) se utiliza para almacenar estructuras de datos dinámicas y código compilado JIT. .NET Compact Framework utiliza la RAM libre, hasta un límite especificado por el dispositivo, para almacenar en una memoria caché el código generado y las estructuras de datos, y después libera la memoria cuando es oportuno.

El Common Language Runtime utiliza una técnica de pitching de código para liberar bloques de código compilado JIT en tiempo de ejecución cuando no hay suficiente memoria. Esto permite ejecutar programas más grandes en sistemas con limitaciones de RAM con una pérdida de rendimiento mínima.

3.2 Almacenamiento en ROM

El código nativo que compone el Common Language Runtime puede residir en la memoria de sólo lectura (ROM) o en un sistema de archivos de RAM. .NET Compact Framework usa el espacio libre en la memoria ROM, Flash o el espacio del disco para permitir que las aplicaciones sigan ejecutándose, con menor rendimiento, en situaciones de escasez de memoria.

Los archivos que contienen instrucciones del lenguaje intermedio de Microsoft (MSIL) y metadatos para bibliotecas de clases se almacenan en un sistema de archivos en ROM o en RAM. Las bibliotecas de clases pueden descargarse en un sistema de archivos de lectura/escritura como parte del proceso de instalación de una aplicación descargable.

4. Diferencias con .NET Framework

4.1 .NET Framework

- **ASP.NET** – Se basó en el CLR. Puede ser usado sobre un servidor para desarrollar aplicaciones de Web. Los formularios Web creados en ASP.NET pueden ser usados para desarrollar interfaces de usuarios Web dinámicas fácilmente. No soportado en .NET Compact Framework.
- **Desktop controls** - 35 controles de escritorio disponibles y soportados.
- **File change notifications** – Soportado en .NET Framework pero no soportado en .NET Compact Framework debido a las diferencias en los sistemas operativos de dispositivo; hay limitaciones sobre los modelos de E/S.
- **Tooltips** – Este control de interfaz de usuario gráfico es soportado en el .NET Framework, pero estas API están ausentes en el .NET Compact Framework.

- **Checked List Box** - Este control de interfaz de usuario gráfico es soportado en el .NET Framework, pero estas API están ausentes en el .NET Compact Framework.
- **COM Objects** – No hay soporte para el desarrollo de objetos COM en el .NET Compact Framework
- **.NET Remoting** – No hay soporte para .NET Remoting en .NET Compact Framework. XML services son usadas como alternativa para esto.
- **System.Windows.Forms Namespace** – No hay soporte para drag and drop, printing, Microsoft ActiveX® support, y GDI+ en .NET Compact Framework.

4.2 .NET Compact Framework

- 28 de los 35 controles están disponibles en el .NET Compact Framework. Estos controles son mejorados para el tamaño y los requisitos de rendimiento del .NET Compact Framework. No todas las propiedades, eventos y métodos del .NET son admitidos en el .NET Compact Framework.
- **OLE DB Wrapper** – No soportada en .NET Compact Framework.
- **Datagrids** – No incorpora el soporte de edición como el .NET Framework. el .NET Compact Framework solo puede unir datagrids con tablas simples.
- **Infrared Data Association classes** - .NET Compact Framework nos brinda la facilidad de hacer conexiones infrarrojas y clases Web listening para servicios HTTP para dispositivos. Estos no están disponibles en el .NET Framework.
- **Infrared send/receive** – El .NET Compact Framework incluyen nuevas API's que le permiten a las aplicaciones enviar y recibir información sobre el puerto IR. Esta característica no esta presente en el .NET Framework.

- **Serialization** - Debido a las consideraciones de rendimiento, el .NET Compact Framework no soporta la serialización binaria usada por BinaryFormatter, o serialización SOAP usada por SoapFormatter. Hay, sin embargo, soporte para serializar objeto de datos al ser transmitidas usando SOAP con XML Web services.

5. Conceptos de hardware a tener en cuenta.

En el desarrollo de una aplicación Windows, Web o Middleware no tenemos en cuenta si la PC o servidor dónde se esté ejecutando tiene o no alimentación, espacio disponible o cobertura GPRS. Son conceptos que se dan por supuestos (o sencillamente no son necesarios como la cobertura GPRS) que estarán y no debemos (o no debiéramos) preocuparnos de ellos.

.NET Compact Framework está enfocado a dispositivos móviles, esto es, con determinadas características en la administración de memoria, procesos y recursos. Las aplicaciones que desarrollemos sobre ellos, aunque gracias al CLR de .NET CF 3.5 en ocasiones son transparentes, no debemos olvidar que el dispositivo móvil puede no tener suficiente batería como para, por ejemplo, efectuar una conexión GPRS/3G a un servicio Web, o bien no tiene el suficiente espacio y eficiencia como para almacenar un gran volumen de datos. Estos escenarios no nos los encontramos en aplicaciones Windows o Web.

Una premisa que se debe tener en cuenta es que todos los dispositivos son distintos entre sí. Evidentemente no los que son del mismo modelo y marca. Un PC clónico con Windows XP SP2 es distinto a uno “de marca” con el mismo sistema operativo pero esas diferencias (RAM, Caché, disco, procesador, controladores, etc.) no son temas a tener en cuenta antes del desarrollo.

Windows Mobile es instalado por parte del fabricante a cada uno de sus modelos siguiendo unas especificaciones técnicas. Microsoft, por su parte, provee todas las funcionalidades básicas a Windows Mobile sin embargo no todas esas funcionalidades están disponibles. Por ejemplo, un dispositivo de la

marca X incorpora un controlador de Bluetooth de Microsoft (Microsoft Bluetooth Stack).

Windows Mobile soporta este tipo de controlador por defecto así que, no habrá problema alguno. Otros fabricantes optan por utilizar Bluetooth de WIDCOMM. En ese caso el controlador Bluetooth que incorpora Windows Mobile por defecto es sustituido por el de WIDCOMM. Este controlador reside en ROM, así que forma parte del sistema operativo. Si decides hacer cualquier tipo de código de control de Bluetooth para un determinado dispositivo que utiliza Microsoft Bluetooth Stack no te servirá de nada si lo ejecutas en otro dispositivo con WIDCOMM. Lo mismo ocurre con el control de intensidad de la pantalla o teclado. Windows Mobile aporta la funcionalidad de modificarla mediante código sin embargo los fabricantes pueden optar por mantenerlo o sustituirlo por controlarlo por librerías nativas, por ejemplo.

6. Microsoft ActiveSync y la instalación de programas en Móviles

Si acabas de adquirir una PDA o Smartphone con SO Windows Mobile, quizá no sepas que igual que ocurre con un PC, puedes instalarle todo tipo de aplicaciones que amplíen su funcionalidad y utilidad.

La instalación de programas en Windows Mobile es muy sencilla, aunque también ligeramente diferente a lo que estamos acostumbrados en Windows. Te habrás dado cuenta de que cuando descargas una aplicación para tu PDA/Smartphone, aparte de archivos comprimidos en RAR o ZIP, te encuentras 2 tipos de ficheros: CAB y EXE.

- **Ficheros .CAB:** CAB es un formato de compresión utilizado por Microsoft para compilar archivos ejecutables en un espacio reducido. Para instalar este tipo de ficheros en Windows Mobile, deberemos copiarlos en la memoria del dispositivo y ejecutarlos desde ahí.



- **Ficheros .EXE** (o .MSI): es el formato que se utiliza para los ejecutables en Windows (PC), y no están soportados por Windows Mobile, por lo que si tratamos de instalarlos directamente desde el dispositivo nos será imposible hacerlo. Para instalar este tipo de archivos en la PDA/Smartphone, deberemos hacerlo a través de **ActiveSync** (o **Windows Mobile Device Center** si usas Windows Vista).

6.1 ¿Qué es Microsoft ActiveSync?

ActiveSync es el programa que utilizamos para sincronizar los datos de la PDA/Smartphone con nuestro ordenador. Desde el momento en el que adquieras un dispositivo Windows Mobile, ActiveSync se convertirá en una de las herramientas de uso común de tu PC.

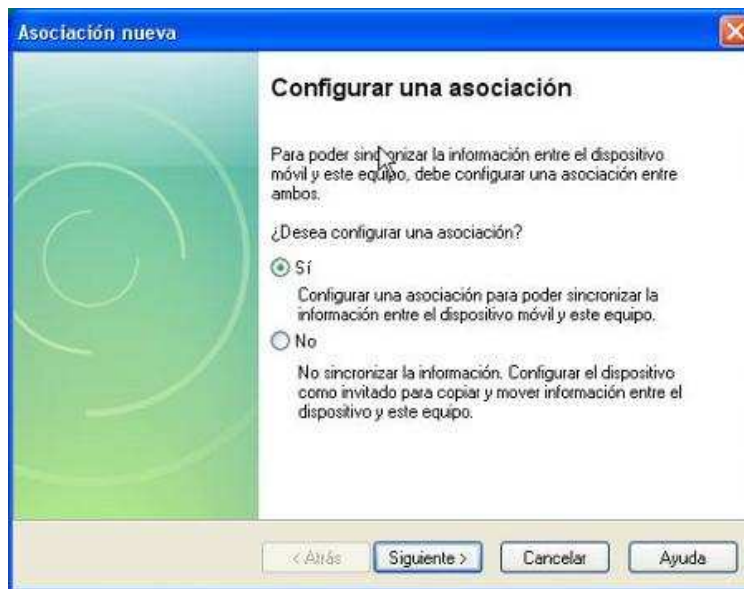
Además de sincronizar datos, contactos, favoritos y ficheros, ActiveSync se utiliza para instalar aplicaciones en el dispositivo desde el mismo ordenador.

6.1.1 Configuración

Por supuesto hay muchas maneras de configurar el ActiveSync, dependiendo de las necesidades y software instalado. Lo aquí dispuesto

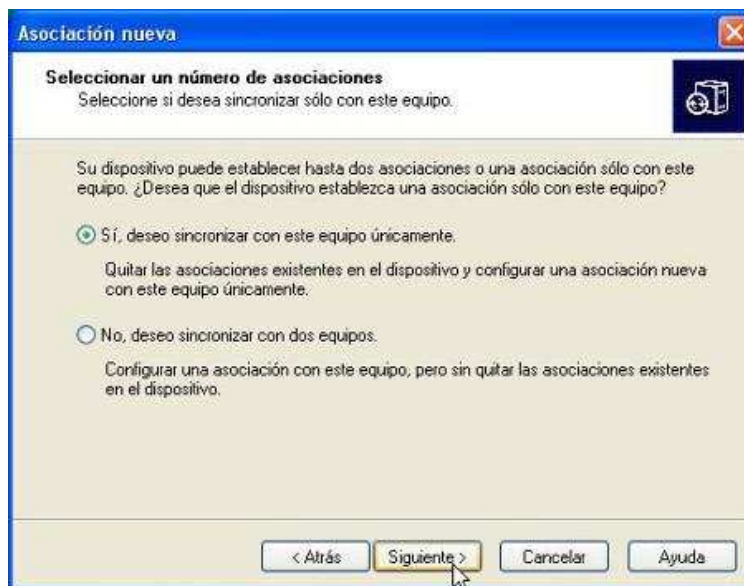
funcionará bien en la mayoría de dispositivos y optimiza la conexión para la copia de archivos entre el PC y el dispositivo.

La primera vez que se conecte un dispositivo el sistema operativo detectará un nuevo dispositivo y el *active sync* le mostrará la siguiente ventana:



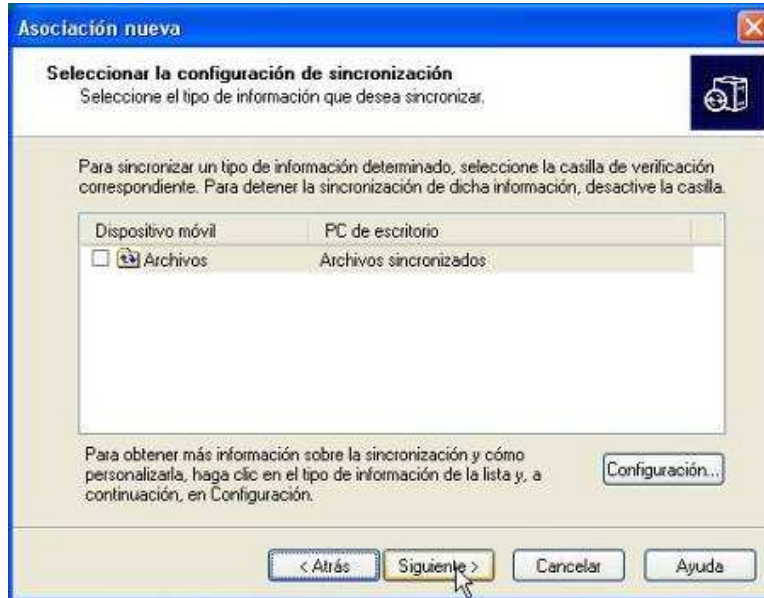
Seleccionamos *Sí* y pulsamos *Siguiente*

En las últimas versiones de Active Sync aparecerá lo siguiente:



Seleccionamos *Sí*, deseo sincronizar con este equipo únicamente y continuamos. (en versiones antiguas no saldrá este paso...)

En la siguiente pantalla, simplemente pulsar en *Siguiente*



Y finalizar

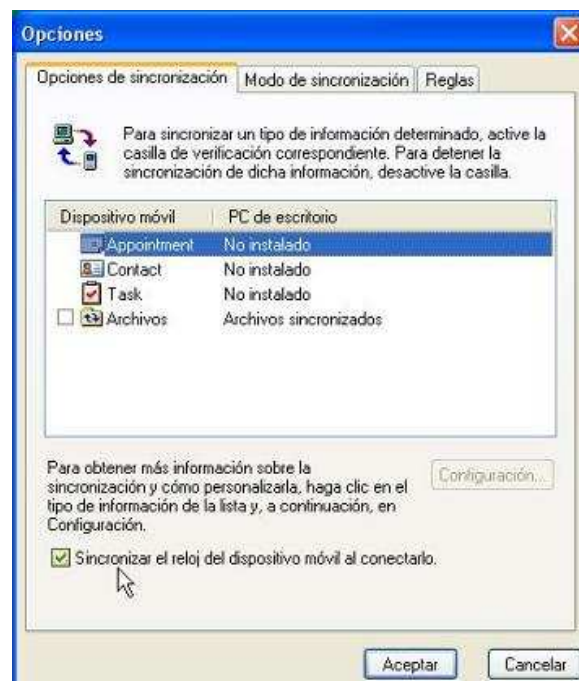


Ahora tenemos la ventana inicial de trabajo:

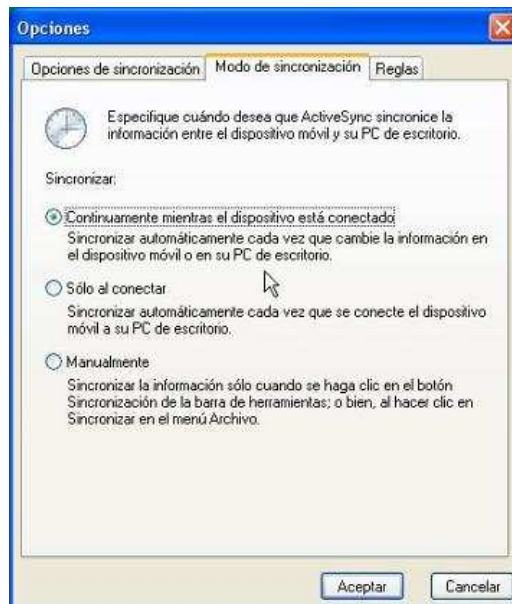


Ya está el equipo sincronizado (podremos copiar ficheros sin problemas).

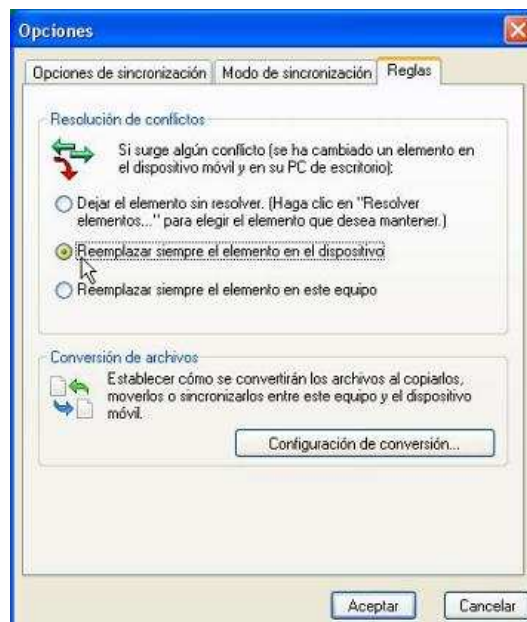
A continuación configuraremos la asociación para que la sincronización de los menos problemas posibles... Para ello pulsaremos en **opciones** y aparecerá lo siguiente:



Conectamos la casilla de *Sincronizar reloj* y pasamos a la siguiente pestaña (modo de sincronización)



Aquí comprobamos que esta seleccionado *Sincronizar continuamente* y pasamos a *Reglas*



Para finalizar aquí seleccionamos la 2ª opción (reemplazar siempre el elemento en el dispositivo) y pulsamos *Aceptar*

Con esto la comunicación entre los dispositivos quedará optimizada para la copia de archivos y el reloj del dispositivo se pondrá en hora cada vez que se conecten.

Una vez hayamos configurado el dispositivo, sólo tendremos que conectarlo y ejecutar desde el PC el fichero EXE que queramos instalar. ActiveSync se encargará de lanzar su herramienta para agregar o quitar programas y de instalarlo en el dispositivo.



Por último, salvo especificación del fabricante de la aplicación, es recomendable instalar los programas en la tarjeta de memoria, ya que no ocuparán espacio en la memoria principal y no lastrarán el rendimiento de tu dispositivo.

7. Diferencias entre Smartphone y Pocket PC

7.1 La Pantalla

Aparte del hecho que el Smartphone sirva para hacer llamadas telefónicas, hay algunas otras grandes diferencias entre los dos dispositivos. Vamos a comenzar con la pantalla. Un Pocket PC utiliza un típico estilo de pantalla PDA, en la que la pantalla abarca la mayoría de la parte frontal del dispositivo. Un Smartphone tradicional usa una pantalla que es sólo ligeramente superior a la pantalla encontrados en los celulares comunes.

Una pantalla más pequeña no significa imágenes pequeñas. También hay una diferencia significativa en el número de píxeles que los dos dispositivos soportan. Una típica pantalla de Smartphone tiene una resolución de 176 x 220, mientras que un Pocket PC tiene una resolución de 320 x 240. Esto significa que el Pocket PC puede mostrar mucha más información sobre la pantalla que un Smartphone.

7.2 Ingreso de Datos

Otra gran diferencia a tener en cuenta es la cantidad de entrada de dato. El ingreso de datos sobre un Pocket PC es típicamente realizada utilizando un lápiz especial. Aunque la entrada de datos varían de modelo a modelo, las Pocket PC suelen dar la opción de entrando datos pulsando las teclas en un teclado en pantalla o usando el lápiz para escribir sobre una parte de la pantalla.

Debido al escaso tamaño de la pantalla, los Smartphones no ofrecen un teclado en pantalla. En cambio, la entrada de datos sobre estos es realizada utilizando los botones del teléfono. Por ejemplo, si desea la letra C, tendría que presionar el botón "2" tres veces.

7.3 Hardware

Otras consideraciones para tener en cuenta son que la Pocket PC generalmente tienen más memoria y procesadores rápidos que un Smartphone debido a su tamaño y porque no tienen que utilizar

dispositivos de telefonía móvil. Por ejemplo, no es raro que un Pocket PC pueda tener 128 MB de RAM. En contraste, algunos Smartphones solo tienen 8 MB de RAM.

Las Pocket PC también pueden ampliar ciertas características. La mayoría Pocket PC le permitirá ganar adicional de almacenamiento mediante el uso de tarjetas de memoria flash compacto o tarjetas Secure Digital (SD). Smartphone soporta tarjetas SD, pero en este momento hay algunos problemas de compatibilidad que todavía tienen que resolverse.

7.4 Aplicaciones

Como se ha explicado anteriormente, los Smartphones tienen un hardware más modesto que las Pocket PC. Por lo tanto no debe ser ninguna sorpresa que no se pueda ejecutar en los Smartphones casi tantas aplicaciones como las que soportan las Pocket PC.

Por defecto, los Smartphones vienen con un conjunto básico de aplicaciones como bandeja de correo, Calendario, Internet Explorer, ActiveSync, MSN Messenger, Pocket MSN, tareas, Canto notas y Windows Media. Hay también una calculadora y un par de juegos.

Las Pocket PC incluyen el mismo conjunto básico de aplicaciones como los Smartphones, además de un mucho más aplicaciones. La más notable es una versión de Pocket Office de Microsoft.

Además de las diferencias en las aplicaciones incluidas en los dos dispositivos, también hay grandes diferencias en las aplicaciones hechas por terceros que están disponibles. Hay cientos, si no miles, de aplicaciones para Pocket PC que se puede descargar de Internet en contraste con las pocas aplicaciones que hay para Smartphone.

7.5 Conectividad

Tanto la Pocket PC y el Smartphone pueden sincronizar con la PC mediante el uso de un acoplamiento USB. La mayor diferencia es que el Smartphone sólo puede ser sincronizado con una sola cuenta Email, mientras que la Pocket PC puede por sincronizados con múltiples cuentas Email.

Ambos dispositivos también son capaces de conectarse a Internet. La conexión primaria para un Smartphone es un enlace vía línea telefónica, mientras que la conexión primaria para un Pocket PC es usualmente Wi-Fi. Cuando se trata de conectarse a Internet, la conexión vía línea telefónica es más lento y más caro en comparación con la conexión Wi-Fi, que generalmente están disponibles en cualquier lugar que usted puede obtener una señal.

8. Lo nuevo en .NET Compact Framework 3.5

.NET Compact Framework versión 3.5 amplía .NET Compact Framework con muchas características nuevas. Este tema proporciona información sobre las principales incorporaciones y modificaciones.

Para instalar .NET Compact Framework 3.5 en ROM en dispositivos con Windows Embedded CE, debe obtener la actualización mensual correcta de Platform Builder en el sitio web Windows Embedded CE Updates.

Windows Communication Foundation

.NET Compact Framework 3.5 admite Windows Communication Foundation (WCF), que es el modelo de programación unificado de Microsoft para generar las aplicaciones orientadas a servicios. Los clientes que están ejecutando .NET Compact Framework pueden conectarse a los servicios web de WCF que ya existan en el escritorio. Además, se ha agregado compatibilidad para un nuevo transporte de WCF, el transporte de correo Microsoft Exchange Server, tanto para aplicaciones .NET Compact Framework como para aplicaciones de escritorio.

LINQ

Language-Integrated Query (LINQ) agrega funciones de consulta de uso general a .NET Compact Framework que se aplican a diferentes orígenes de

información, como bases de datos relacionales, datos XML y objetos en memoria.

Formularios Windows Forms

TabPage, Panel, Splitter, PictureBox: Ahora, los usuarios pueden agregar gráficos a estos controles

Control: Ahora, se admiten fuentes ClearType y puede modificar la propiedad BackColor de los controles de sólo lectura.

ComboBox: Ya no se admiten las propiedades SelectionStart y SelectionLength.

SoundPlayer

.NET Compact Framework 3.5 admite SoundPlayer, que permite reproducir varios sonidos. Un dispositivo puede mezclar estos sonidos si el hardware admite esta posibilidad.

Compresión

.NET Compact Framework 3.5 incorpora compatibilidad para las siguientes clases del espacio de nombres System.IO.Compression:

- CompressionMode
- DeflateStream
- GZipStream

Generador de perfiles de CLR de .NET Compact Framework

.NET Compact Framework 3.5 admite el generador de perfiles de CLR, que sólo estaba disponible con la versión completa de .NET Framework. El generador de perfiles de CLR permite ver el montón administrado de un proceso e investigar el comportamiento del recolector de elementos no

utilizados. El generador de perfiles de CLR y su documentación asociada están incluidos en las herramientas avanzadas de .NET Compact Framework.

Herramienta de configuración

.NET Compact Framework 3.5 admite la herramienta de configuración, que proporciona información sobre la versión del motor en tiempo de ejecución y funciones administrativas que permiten, por ejemplo, especificar en qué versión de .NET Compact Framework se ejecutará una aplicación. La herramienta de configuración y su documentación asociada están incluidas en las herramientas avanzadas de .NET Compact Framework.

Depuración

Las mejoras realizadas en la depuración de .NET Compact Framework 3.5 son las siguientes:

- Ahora se admiten las evaluaciones de funciones anidadas.
- Ahora, las excepciones no controladas realizan la interrupción en el lugar donde ocurrió la excepción, en lugar del lugar donde se llamó al método Run.

Registro

Se han realizado las mejoras siguientes en las características de registro:

- Ahora, los registros de interoperabilidad incluyen información sobre los objetos cuyas referencias se van a calcular y que están contenidos en estructuras o en tipos de referencia.
- El registro de finalizador incluye información sobre el orden y la temporización del finalizador.
- Los archivos de registro ya no se bloquean mientras la aplicación se está ejecutando. Por consiguiente, puede leer los registros en tiempo de ejecución.

- Las trazas de la pila incluyen la firma de método completa para distinguir las sobrecargas de los métodos.


Id. de plataforma


.NET Compact Framework 3.5 proporciona información nueva sobre el tipo de plataforma, concretamente si una plataforma es Pocket PC o Smartphone.


Herramientas del motor en tiempo de ejecución

Ahora, la biblioteca de herramientas del motor en tiempo de ejecución proporciona compatibilidad para ejecutar con el emulador las herramientas de diagnóstico del SDK de .NET Compact Framework, como Monitor de rendimiento remoto. Supervisión remota del rendimiento y su documentación asociada están incluidos en las herramientas avanzadas de .NET Compact Framework.

Para recordar

 .NET Compact Framework es el entorno en el que se ejecutan las aplicaciones administradas en los dispositivos. Proporciona acceso a las funciones subyacentes del dispositivo. Además, las aplicaciones y los componentes pueden interactuar en el dispositivo y a través de Internet.

 .NET Compact Framework es un subconjunto del entorno completo de .NET Framework. Implementa aproximadamente un 30 por ciento de la biblioteca de clases completa de .NET Framework y contiene asimismo las características y clases específicas del desarrollo móvil e incrustado.

 Si desea saber más acerca de estos temas, puede consultar las siguientes páginas.

- [http://msdn.microsoft.com/es-es/library/f44bbwa1\(VS.80\).aspx](http://msdn.microsoft.com/es-es/library/f44bbwa1(VS.80).aspx)
- http://articles.techrepublic.com.com/5100-22_11-5647464.html
- [http://msdn.microsoft.com/es-es/library/024xhhht\(VS.80\).aspx](http://msdn.microsoft.com/es-es/library/024xhhht(VS.80).aspx)

**UNIDAD DE
APRENDIZAJE****1****SEMANA****2**

Desarrollo de aplicaciones con Visual Studio 2008 – Manejo de entorno

OBJETIVOS ESPECÍFICOS

- Conocer la herramienta Visual Studio 2008 para el desarrollo de dispositivos móviles.

CONTENIDOS

- Uso de Visual Studio.

1. Uso de Visual Studio

Visual Studio 2008 incluye las herramientas y marcos de trabajo necesarios con el fin de desarrollar aplicaciones para Pocket PC, Smartphone y otras plataformas basadas en .NET de Windows CE. Si no cuenta con un dispositivo inteligente, es posible crear y probar las aplicaciones para dispositivos inteligentes utilizando la tecnología de la emulación sin dejar el entorno de desarrollo integrado (IDE) de Visual Studio.

Visual Studio 2008 admite los lenguajes de Visual Basic .NET, Visual C# y Visual C++ para el desarrollo de aplicaciones de dispositivos inteligentes.

Visual Studio admite dos enfoques con el fin de desarrollar aplicaciones para dispositivos.

- Es posible desarrollar aplicaciones Web móviles que se ejecuten en un servidor Web y que se representan en formatos diferentes atendiendo a una amplia variedad de dispositivos móviles equipados en el explorador.
- Es posible desarrollar aplicaciones cliente complejas basadas en Windows Mobile y Windows CE que se ejecutan en el propio dispositivo. Este último enfoque es lo que se conoce como desarrollo de aplicaciones para *Smart Devices*.

1.1 Compatibilidad entre versiones

En los proyectos administrados de Visual Studio 2008, todas las plataformas tienen como destino la versión 3.5 de .NET Compact Framework a menos que se indique de otra manera. Por ejemplo, en el cuadro de diálogo **Nuevo proyecto**, las plantillas de dispositivos

inteligentes aparecen marcadas con "(1.0)" si su versión de destino es la 1.0 de .NET Compact Framework.

1.2 Instrucciones de diseño

El diseño de una aplicación para dispositivo determina con qué facilidad, rapidez y eficiencia puede un usuario llevar a cabo sus tareas. Al optimizar una aplicación para sacar partido de las capacidades de diferentes dispositivos, se ofrecerá al usuario una experiencia óptima, al crear una aplicación más utilizable, coherente, receptiva y accesible. Para obtener instrucciones detalladas sobre diseño relacionadas con características de interfaz específicas, vea el kit de desarrollo de software (SDK) de un determinado dispositivo.

1.3 Emulador de dispositivos

El emulador de dispositivos está específicamente diseñado para proyectos de dispositivos de Visual Studio 2008. Ejecuta aplicaciones compiladas para el conjunto de instrucciones ARM y se ejecuta como un proceso en modo usuario. Ahora Visual Studio proporciona transporte de Acceso directo a memoria (DMA) para comunicarse con el emulador. Superando al transporte tradicional TCP/IP, el transporte de DMA es más rápido, no basado en conectividad de red u otros factores externos, a la vez que proporciona una conexión y desconexión determinística.

Visual Studio 2008 incluye imágenes del emulador para Pocket PC 2003 SE, Pocket PC 2003 SE Square, Pocket PC 2003 SE Square VGA, Pocket PC 2003 VGA, Smartphone 2003 SE y Smartphone 2003 SE QVGA.

Haga clic en **Ayuda** en la barra de menú del emulador para ver la colección de temas de la Ayuda relacionados con el emulador.

Para abrir el emulador, haga clic en **Herramientas**, en **Conectar con dispositivo**, seleccione el emulador que desee abrir y, a continuación, haga clic en **Conectar**.

1.4 Dispositivo y escritorio

Se utiliza el mismo entorno de Visual Studio que se utiliza para desarrollar aplicaciones de escritorio, pero surgen algunas diferencias palpables al establecer como destino determinados dispositivos. Por ejemplo:

- El entorno de Visual Studio proporciona herramientas adicionales para conectarse y realizar depuraciones en un dispositivo.
- Aparte de seleccionar un tipo de proyecto y una plantilla cuando se crea un proyecto, debe seleccionarse un dispositivo en el que se ejecutará y depurará la aplicación. El dispositivo puede ser un dispositivo físico conectado al equipo de desarrollo, un dispositivo conectado en red o un emulador de dispositivos que se ejecute en el equipo de desarrollo.
- El número de clases y de miembros de éstas difieren de los disponibles cuando se desarrollan aplicaciones para escritorio. En proyectos administrados que utilizan .NET Compact Framework, hay pocas clases que se encuentren disponibles para dispositivos y el complemento de las clases normalmente difiere entre plataformas. Sucede lo mismo con los proyectos nativos, en lo que sólo se encuentra disponible un subconjunto de API de Windows, clases MFC o componentes ATL. Es posible determinar el tipo de clases disponibles con tan sólo ver la documentación, utilizar IntelliSense o utilizar el Examinador de objetos de Visual Studio mientras que un determinado proyecto se encuentre activo.
- Como ocurre con las aplicaciones de escritorio, puede obtener acceso al código nativo mediante la invocación de la plataforma. .NET Compact Framework ofrece una compatibilidad limitada con la interoperabilidad COM. No acepta la creación de objetos COM en el código administrado ni la interoperabilidad con los controles ActiveX.

- Algunos elementos del lenguaje pueden diferir, por ejemplo, no se aceptan todas las palabras clave de Visual Basic que se utilizan en la programación para escritorio.
- Algunos miniprogramas de código proporcionados en la documentación de Visual Studio para proyectos de escritorio pueden generar errores de ejecución en proyectos de dispositivos.

Hay ciertas consideraciones del diseño, como el factor de la forma del dispositivo, el gasto de energía, limitaciones de la memoria y otros detalles, que no plantean dificultades en el desarrollo para escritorio.

2. Capacidades de dispositivos y herramientas de desarrollo necesarias

Visual Studio admite el desarrollo de aplicaciones para dispositivos que ejecuten varias versiones de Windows Mobile y Windows CE. Sin embargo, Visual Studio no admite el desarrollo de aplicaciones en dispositivos heredados. Esta situación puede llevar a confusión en cuanto a lo que se necesita a modo de herramientas de desarrollo, la versión .NET Compact Framework y el sistema operativo de Windows CE subyacente.

2.1 Gráficos para la comparación de herramientas

Las siguientes tablas proporcionan una instantánea de las variaciones relacionadas con el hardware de dispositivos inteligentes, las características de hardware y las herramientas de desarrollo. Estos listados pueden cambiar con el tiempo.

2.1.1 Información general sobre las funciones IDE

Esta tabla proporciona información general sobre las funciones de los diferentes IDE. Las abreviaturas de encabezado de columna son las siguientes:

- eVT3C = eMbedded Visual C++ 3.0
- eVT3V = eMbedded Visual Basic 3.0
- eVC4 = eMbedded Visual C++ 4.0 y Service Pack 4.0
- VS2003 = Visual Studio .NET 2003
- VS2005 = Visual Studio 2005
- VS2008 = Visual Studio 2008

		eVT3C	eVT3V	eVC4	VS2003	VS2005	VS2008
Tipo de código	Código nativo	X		X		X	X
	Código interpretado		X				
	Código administrado				X	X	X
	Código del servidor (web)				X	X	X
SDK de dispositivos	Pocket PC 2000 y Pocket PC 2002	X	X		X		
	Smartphone 2002	X					
	Windows Mobile 2003			X	X	X	X
	Windows Mobile 2003 (segunda edición)			X	X	X	X
SDK de dispositivos	Windows Mobile 5.0					X	X
	Windows Mobile 6.0					X	X

2.1.2 Compatibilidad de sistemas operativos y herramientas de .NET Compact Framework


Esta tabla proporciona información general sobre qué versiones de herramientas y qué versiones de software de Windows Mobile admiten las versiones 1.0, 2.0 y 3.5 de .NET Compact Framework.


		Versión 1.0	Versión 2.0	Versión 3.5
Herramienta	Visual Studio .NET 2003	X		
	Visual Studio 2005	X	X	
	Visual Studio 2008		X	X
Versión de software de Windows Mobile	Windows Mobile 6.0		En ROM (2.0 SP2)	Usuario que se puede instalar
	Windows Mobile 5.0	En ROM (1.0 SP3)	Usuario que se puede instalar	Usuario que se puede instalar
	Windows Mobile 2003 (segunda edición)	En ROM (1.0 SP2)	Usuario que se puede instalar (sólo Pocket PC)	Usuario que se puede instalar
	Windows Mobile 2003	En ROM (1.0 SP2)	Usuario que se puede instalar (sólo Pocket PC)	Usuario que se puede instalar


- Compruebe con el fabricante del dispositivo la actualización relacionada con un determinado dispositivo en una versión posterior de Windows CE o Windows Mobile. Microsoft no proporciona actualizaciones para determinados dispositivos destinados a usuarios finales.
- Las ediciones de Visual Studio Express no incluyen compatibilidad con proyectos de Smart Device.

- Ya no se admiten las herramientas de eMbedded Visual Basic. El tiempo de ejecución de eMbedded Visual Basic ya no se encuentra en la memoria ROM de dispositivos.

Para recordar

 Es posible desarrollar aplicaciones Web móviles que se ejecuten en un servidor Web y que se representan en formatos diferentes atendiendo a una amplia variedad de dispositivos móviles equipados en el explorador.

 Si desea saber más acerca de estos temas, puede consultar las siguientes páginas.

 [http://msdn.microsoft.com/es-es/library/77wek3f7\(VS.80\).aspx](http://msdn.microsoft.com/es-es/library/77wek3f7(VS.80).aspx)

**UNIDAD DE
APRENDIZAJE****1****SEMANA****3**

Uso de controles

OBJETIVOS ESPECÍFICOS

- Conocer los controles disponibles del .NET Compact Framework

CONTENIDOS

- Controles del .NET Compact Framework

1. Controles de .NET Compact Framework

La tabla siguiente enumera los controles, componentes y tipos que proporciona .NET Compact Framework para desarrollar aplicaciones de Windows Forms, así como las notas de implementación pertinentes en cada caso.

Puede cargar y crear instancias de datos y controles en el constructor del formulario, pero colocar el control y configurar otras propiedades se realiza mejor en la función de carga.

.NET Compact Framework 3.5 aumenta la compatibilidad con la propiedad BackColor en la mayoría de los controles comunes.

Control o componente	Pocket PC	Smartphone	Notas
Button	Sí	No	Para obtener información sobre cómo crear controles de botón.
CheckBox	Sí	Sí	Para dar al usuario una opción del tipo verdadero/falso o sí/no
Clipboard	Sí	Sí	Se admite en la versión 2.0.
ComboBox	Sí	Sí	El valor predeterminado para DropDownStyle es DropDownList.
ContextMenu	Sí	No	Un menú contextual creado desde un formulario secundario permanece en la pantalla cuando se destruye dicho formulario.
Control	Sí	Sí	Puede usar este control como base

			para controles personalizados.
Control o componente	Pocket PC	Smartphone	Notas
Cursor y Cursors	Sí	Sí	.NET Compact Framework sólo admite la propiedad Current para Cursor , y para los cursores WaitCursor y Default.
DataGrid	Sí	Sí	La clase DataGrid para .NET Compact Framework está en un ensamblado independiente. Debe agregar una referencia a System.Windows.Forms.DataGrid.dll en el proyecto para utilizarlo.
DateTimePicker	Sí	Sí	Windows Mobile 5.0 para Smartphone necesario.
DocumentList	Sí	No	Muestra y administra documentos. Este control sólo está disponible en .NET Compact Framework.
DomainUpDown	Sí	Sí	La propiedad Height cambia el tamaño del control, a diferencia de lo que ocurre en .NET Framework completo. El evento SelectedItemChanged solamente ocurre cuando se hace clic en las flechas arriba o abajo, no cuando se escribe texto en el control.
Form	Sí	Sí	En Pocket PC, la propiedad FormBorderStyle solamente admite

			los valores None y FixedSingle.
Control o componente	Pocket PC	Smartphone	Notas
HardwareButton	Sí	No	Proporciona funcionalidad de reemplazo para los botones del hardware de Pocket PC.
Help	Sí	No	Muestra los archivos HTML de Ayuda utilizados en la ayuda de Pocket PC.
HScrollBar	Sí	Sí	
ImageList	Sí	Sí	
InputPanel	Sí	No	Manipula el panel de entrada de software (SIP) en un Pocket PC. Este control sólo está disponible en .NET Compact Framework.
InputMethodCollection	Sí	No	Proporciona acceso a todo el software del método de entrada instalado en un Pocket PC. Este control sólo está disponible en .NET Compact Framework.
InputMode	No	Sí	Este control sólo está disponible en .NET Compact Framework.
LinkLabel	Sí	No	Compatibilidad limitada de un hipervínculo básico al que se aplica automáticamente el formato de subrayado y de color azul.
Label	Sí	Sí	

Control o componente	Pocket PC	Smartphone	Notas
ListBox y ListControl	Sí	Sí	Si la lista no contiene elementos, no se puede establecer la propiedad Text. En .NET Framework completo, el valor se conserva pero no se tiene en cuenta. .NET Compact Framework no permite seleccionar varios elementos.
ListView	Sí	Sí	.NET Compact Framework no permite seleccionar el método Sort. .NET Compact Framework no permite seleccionar varios elementos.
LogFont	Sí	Sí	Define una estructura de fuente lógica (LogFont) para los efectos de texto, como el texto en ángulo. Esta clase sólo está disponible en .NET Compact Framework.
MainMenu	Sí	Sí	No se puede agregar un elemento de menú a la ubicación ordinal de un separador de menús. .NET Compact Framework produce una excepción y .NET Framework completo hace caso omiso de esta acción. No se puede mostrar el símbolo de Y comercial (&) en el texto de un

			elemento de menú.
Control o componente	Pocket PC	Smartphone	Notas
MessageBox	Sí	Sí	Windows Mobile para Smartphone sólo admite cuadros de mensaje de 1 ó 2 botones.
MessageWindow	Sí	Sí	Proporciona la capacidad de generar y recibir mensajes de Windows. Esta clase sólo está disponible en .NET Compact Framework.
MonthCalendar	Sí	Sí	
Notification	Sí	No	Muestra y responde a las notificaciones del usuario.
MobileDevice	Sí	Sí	Esta clase proporciona el evento Hibernate, que ofrece la oportunidad de liberar recursos almacenados en memoria caché. Esta clase sólo está disponible en .NET Compact Framework.
NumericUpDown	Sí	Sí	La propiedad Height cambia el tamaño del control, a diferencia de lo que ocurre en .NET Framework completo. A diferencia del control de .NET Framework completo, este control no ejecuta una validación de la entrada. El evento ValueChanged solamente ocurre cuando se hace

			clic en las flechas arriba o abajo.
Control o componente	Pocket PC	Smartphone	Notas
OpenFileDialog	Sí	No	El directorio inicial se restringe a la carpeta Mis documentos y sus subcarpetas. El sistema operativo de Pocket PC impone esta restricción para ayudar a los usuarios a organizar sus archivos en los directorios estándar.
Panel	Sí	Sí	
PictureBox	Sí	Sí	
ProgressBar	Sí	Sí	
RadioButton	Sí	No	
SaveFileDialog	Sí	No	
ScreenOrientation	Sí	Sí	Le permite cambiar el valor de orientación de la pantalla a 90, 180 ó 270 sin restablecer el dispositivo. Esta clase requiere el software Windows Mobile versión 5.0 para Pocket PC y Smartphone. Esta clase sólo está disponible en .NET Compact Framework.
Splitter	Sí	No	
StatusBar	Sí	No	Siempre se acopla a la parte inferior del formulario. No se le puede

			cambiar el tamaño.
Control o componente	Pocket PC	Smartphone	Notas
TabControl	Sí	No	El tamaño del control TabControl se ajusta inicialmente de modo que ocupe el área de cliente completa del formulario. Desacople el control para cambiar su tamaño.
TabPage	Sí	No	
TextBox	Sí	Sí	<p>Los controles TextBox de una sola línea, especificados mediante la propiedad Multiline, sólo admiten la alineación a la izquierda. Los controles de cuadro de texto Multiline pueden alinearse a la izquierda, a la derecha o en el centro.</p> <p>Si el valor de Multiline es false, .NET Compact Framework ajusta el tamaño del control según la especificación de Height, pero sólo se puede utilizar la primera línea de TextBox. La versión completa de .NET Framework mantiene el alto en una línea.</p> <p>La propiedad PasswordChar siempre es un asterisco (*), con independencia del valor que se especifique.</p>


Timer	Sí	Sí	
Control o componente	Pocket PC	Smartphone	Notas
ToolBar	Sí	No	<p>Form solamente admite un ToolBar. Si se intenta agregar un ToolBar adicional, se produce una excepción <code>NotSupportedException</code>.</p> <p>No se puede establecer el índice de un botón ToolBar antes de establecer la propiedad <code>ImageList</code> de ToolBar.</p>
TrackBar	Sí	No	
TreeView	Sí	Sí	.NET Compact Framework no admite el evento <code>Click</code> para TreeView . Como alternativa se puede utilizar el evento <code>AfterSelect</code> .
VScrollBar	Sí	Sí	No se puede establecer el valor máximo porque, al igual que el control NumericUpDown , el valor máximo alcanzable es la primera línea vacía sobre el control de posición. Más específicamente, desde las propiedades del editor, equivale a <code>Maximum</code> menos <code>LargeChange</code> más 1.
WebBrowser	Sí	Sí	


Autoevaluación

- ¿Cuáles controles son exclusivos de .NET Compact Framework?

Para recordar

- Puede cargar y crear instancias de datos y controles en el constructor del formulario, pero colocar el control y configurar otras propiedades se realiza mejor en la función de carga.

 Si desea saber más acerca de estos temas, puede consultar las siguientes páginas.

 [http://msdn.microsoft.com/es-es/library/hf2k718k\(VS.80\).aspx](http://msdn.microsoft.com/es-es/library/hf2k718k(VS.80).aspx)

**UNIDAD DE
APRENDIZAJE****1****SEMANA****4**

Construcción de interfaces

OBJETIVOS ESPECÍFICOS

- Desarrollar interfaces amigables para el usuario

CONTENIDOS

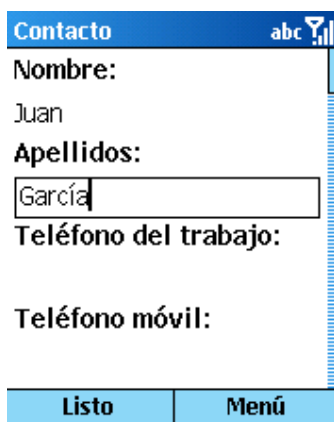
- Conceptos básicos de las interfaces de usuario inteligentes

ACTIVIDAD

- Desarrollar la interfaz de una aplicación móvil usando algunos de los controles mostrados en el capítulo anterior.

1. Conceptos básicos de las interfaces de usuario inteligentes

Smartphone no tiene ningún puntero o lápiz, por lo que se debe utilizar un control de dirección para desplazarse por los formularios. Generalmente los desplazamientos son verticales (arriba y abajo) entre los campos y horizontales (derecha e izquierda) dentro de un campo. Una vez que se ha descontado el título de la ventana y la barra de menús, la pantalla queda limitada a 176x180 píxeles. Las aplicaciones deben estar diseñadas para trabajar con dicha resolución, pero también deben poder escalarse horizontalmente y desplazarse verticalmente para que sean útiles también en el futuro. Si desea obtener más información, consulte la sección Colocación de los controles. El ancho no deja mucho espacio para los campos de entrada, por lo que, encima de los controles de entrada, se colocan etiquetas identificativas de los campos. La convención estándar es que los controles estén en negrita y el campo de entrada en fuente normal. Los controles de entrada no deben tener bordes; y esto se debe a que Smartphone dibujará de forma automática un borde alrededor del control que tenga el enfoque (consulte la figura 1). Si todos los cuadros de texto tuvieran bordes, habría que buscar el cursor.



La imagen muestra una interfaz de usuario de un teléfono inteligente con el título "Contacto". Hay una barra superior azul con el texto "abc" y un icono de señal. El formulario contiene las siguientes etiquetas y campos:

- Nombre:** Juan
- Apellidos:** García (este campo tiene un borde azul que indica que está enfocado)
- Teléfono del trabajo:**
- Teléfono móvil:**

En la parte inferior hay dos botones: "Listo" y "Menú".

Figura 1. Los campos de entrada tienen dibujado un borde para indicar el enfoque.

El diseño del formulario puede desplazarse verticalmente si hay demasiados controles que incluir en el formulario; no obstante, siempre deberá ajustarse de forma horizontal. Además, se ha puesto a dieta a algunos de los controles más grandes, como puede ser la edición de múltiples líneas o el cuadro de lista, para asegurar que puedan encajar en la interfaz de usuario. No obstante, conservan todas sus características de tamaño; si se selecciona el botón de acción, se expanden a pantalla completa.
















Nuevo	Notas
Tiene lugar: Una vez Estado: Ocupado <input type="checkbox"/> Privado Notas: Esto es un campo de edición	Esto es un campo de edición de múltiples líneas. Muy útil para este tipo de notas.
Listo Cancelar	Listo Cancelar

Figura 2. Controles a dieta, el cuadro de edición de múltiples líneas expansible

Las aplicaciones para equipos de escritorio tradicionales y Pocket PC utilizan botones para permitir a los usuarios realizar acciones. Las aplicaciones para Smartphone no deberían nunca utilizar botones, ya que dicha utilización requeriría mover el enfoque a los botones y no promovería un uso o desplazamiento más rápidos. El único lugar donde se pueden ver botones es en un explorador Web. En lugar de utilizar botones para desencadenar acciones, el teléfono dispone de dos botones físicos situados debajo de la pantalla, llamados teclas de software, que se asignan a una barra de menús que contiene hasta dos elementos de menú de nivel superior.

Los controles

Una vez presentados los conceptos básicos, se describirán los 14 controles de interfaz de usuario disponibles en .NET Compact Framework y se explicara cómo se asignan a un proyecto de Smartphone.

Control	Uso por parte de Smartphone
 Label	Se puede utilizar para identificar un campo (el formato debe ser negrita) o para mostrar texto.
 TextBox	Campo de entrada utilizado para capturar datos alfanuméricos. Puede tener múltiples líneas; consulte la figura 2.
 MainMenu	Se utiliza para ejecutar acciones en el formulario.
 CheckBox	Campo de entrada para entradas con dos o tres estados.
 PictureBox	Se utiliza para mostrar mapas de bits o como superficie de dibujo personalizada.
 Panel	Se utiliza para agrupar controles.
 DataGrid	Este control no se admite en Smartphone. Consulte la sección DataGrid que encontrará más abajo.
 ComboBox	Campo de entrada para seleccionar un elemento. Se ha comprimido en una única línea para Smartphone; si se selecciona la tecla de acción, se expandirá a pantalla completa.
 ListView	Debe ser a pantalla completa.
 TreeView	Debe ser a pantalla completa.
 HScrollBar  VScrollBar	Los formularios sólo deben implementar el desplazamiento vertical. Las barras de desplazamiento horizontales se deben utilizar para formularios que no sean de entrada, como la presentación de mapas de bits de gran tamaño.
 Timer	No es un control visual, se utiliza para activar eventos a intervalos específicos.
 ProgressBar	Se utiliza para informar a los usuarios de tareas de ejecución larga.
 ImageList	No es un control visual, se utiliza para almacenar imágenes usadas en la aplicación.

Hemos visto un subconjunto de los controles disponibles para un proyecto de Pocket PC o Windows CE. Algunos de los controles que faltan, como TabControl y ToolBar, no se incluyen ya que no tendría sentido desplazarse por dichos controles utilizando el control de dirección.

La etiqueta

Normalmente las etiquetas están acopladas a un campo de entrada, proporcionando, de esta manera, una identidad para dichos campos de entrada. La figura 4 muestra un nombre de etiqueta que identifica a la entrada del cuadro de texto. Estas etiquetas de identificación se colocan en el lado izquierdo del formulario y la fuente predeterminada es Nina negrita de 11 puntos. La etiqueta y el cuadro de texto que se muestran a continuación se han colocado con un valor Left (izquierda) de 3, para asegurar que ambos queden alineados a la izquierda, cerca del borde.

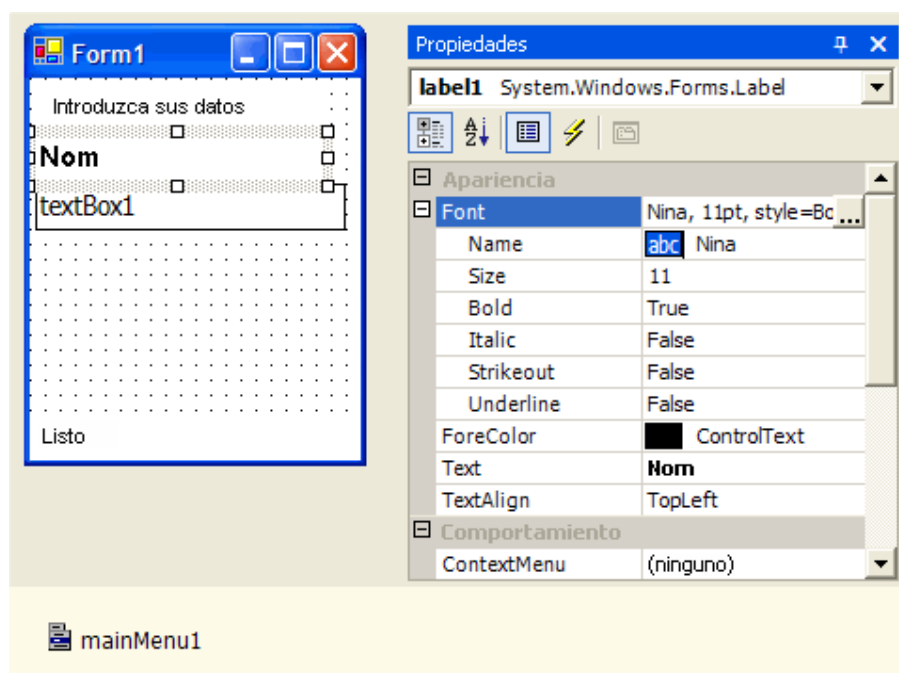


Figura 4. El control de etiqueta

En la parte superior del formulario se incluye otra etiqueta que no está acoplada a ningún campo de entrada. En este caso, he cambiado la fuente a Nina normal de 10 puntos.

El cuadro de texto

El control de cuadro de texto es útil para la entrada de datos de texto. Es importante recordar que los usuarios no van a escribir una novela con el teclado, así que es recomendable capturar sólo lo necesario. Si se configura la propiedad de múltiples líneas, Multiline, como verdadera, se puede expandir el control a pantalla completa para aumentar la capacidad de entrada de datos (consulte la figura 2). El sistema operativo proporciona esta funcionalidad junto con las teclas de software Listo/Cancelar. Además, también se puede definir cuál será el modo de entrada predeterminado para el control, como T9, multipunteo o numérico. Consulte la sección Modos de entrada más adelante en este mismo artículo.

Barras de menús y teclas de software

La certificación del logotipo diseñado por Windows Mobile estipula que la tecla de software izquierda sea siempre una acción predeterminada común (y no un menú emergente). El propósito subyacente es promover que se pueda utilizar fácilmente el teléfono con una sola mano. La tecla de software derecha puede tener elementos de submenús o sólo un elemento de menú de nivel superior. El sistema operativo agrega de forma automática aceleradores numéricos a cada elemento de los menús emergentes. Consulte la figura 5. El número máximo de aceleradores es 10 (del 1 al 0).



Figura 5. Barras de menús

Si se tienen más de 10 elementos de menú bajo la tecla de software derecha, el sistema operativo obligará al usuario a desplazarse arriba y abajo por el

menú emergente, por lo que es más recomendable utilizar menús anidados. Esto a su vez, puede incrementar la complejidad del desplazamiento por los menús, por lo que sigue siendo mejor dividir la funcionalidad entre varios formularios.

Activado, desactivado o puede que activado

En este tema no hay ninguna sorpresa en comparación con un equipo de escritorio o un Pocket PC. El estado de las casillas de verificación puede ser activado, desactivado o indeterminado.

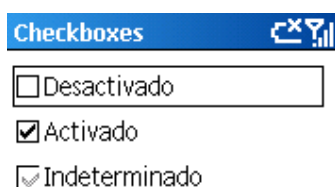


Figura 6. Estados de las casillas de verificación

Como los controles CheckBox contienen texto para identificarlos, no es necesario utilizar la etiqueta identificadora en negrita encima de la casilla de verificación.

Mapas de bits

Estos elementos no tienen ninguna diferencia visual con respecto a los del Pocket PC. Se pueden utilizar mapas de bits para mostrar datos de imágenes, por ejemplo, como los que podría recibir de un servicio Web o se pueden utilizar como una superficie de dibujo personalizada

Panel

El control Panel permite agrupar los controles. Para ver un ejemplo de utilización de un panel, consulte la sección Formularios de desplazamiento.

DataGrid

No busque el control DataGrid en el SDK de Smartphone 2003, porque no lo encontrará. Ello se debe a que no aparece en la lista de los controles administrados compatibles. Una de las diferencias entre las plataformas Pocket PC y Smartphone es la memoria RAM. Un dispositivo Pocket PC normal actual dispone de 64 MB, de los cuales cerca de 32 MB están disponibles de forma predeterminada para aplicaciones. Mi Smartphone tiene 16 MB y la pila de radio ocupa una parte de ellos. Un control DataGrid de gran tamaño es un verdadero lastre para la memoria RAM, en parte porque deberá estar respaldado por un gran origen de datos. Con un conjunto de trabajo significativamente más pequeño, el rendimiento se verá seriamente afectado, y es por eso por lo que no se admite este control.

ComboBox

En los equipos de escritorio y Pocket PC, el control ComboBox proporciona una lista desplegable de elementos. Este control se ha reducido en Smartphone para que ocupe el mismo espacio que un cuadro de texto. También se conoce como control Spinner (de número), consulte la figura 7.

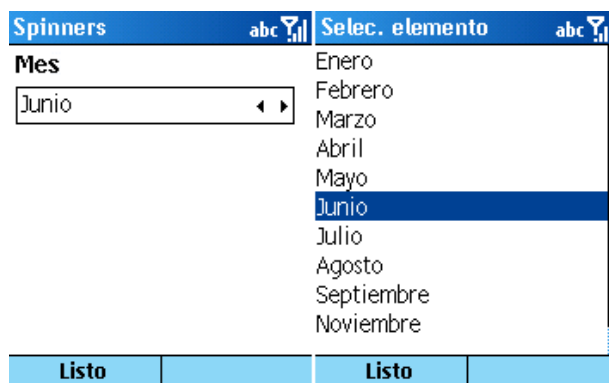


Figura 7. Control de número

Cuando el control recibe el enfoque, el usuario selecciona izquierda y derecha para desplazarse por los elementos disponibles y, al seleccionar la tecla de acción, el control aumenta de tamaño hasta desvelar un cuadro de lista a pantalla completa, con el elemento actual preseleccionado. Si los elementos no cupieran, el sistema operativo proporcionaría de forma automática barras de desplazamiento.

ListView

El control ListView proporciona la misma funcionalidad que la vista de archivos y directorios del Explorador de Windows, consulte las figuras 8 y 9. Normalmente se utiliza junto con un control ImageList.

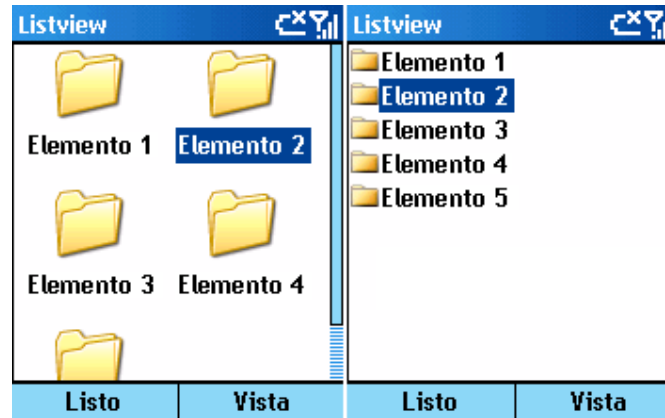


Figura 8. ListView: vistas de iconos grandes y de lista

Si echa un vistazo a todas las aplicaciones internas de Smartphone, no encontrará ninguna interfaz de usuario que utilice estos estilos de ListView. En su lugar, las interfaces utilizan un control ListView personalizado, también conocido como control ListView dibujado por el propietario. Para obtener más información, consulte la sección Controles ListView dibujados por el propietario. De forma adicional, se puede asignar un control CheckBox junto a cada elemento del control ListView. Para ver la interfaz de usuario resultante, consulte la sección Controles no incluidos, Selector de varios elementos.

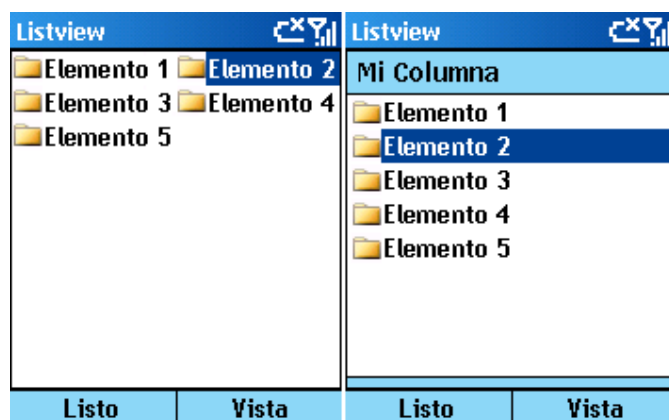


Figura 9. ListView: vistas de iconos pequeños y de detalles

TreeView

De forma predeterminada, el control TreeView ocupa todo el tamaño del área de cliente. Se puede utilizar con un control ImageList y, para cada nodo, puede proporcionar una imagen que indique si está o no seleccionado. Consulte la figura 10.

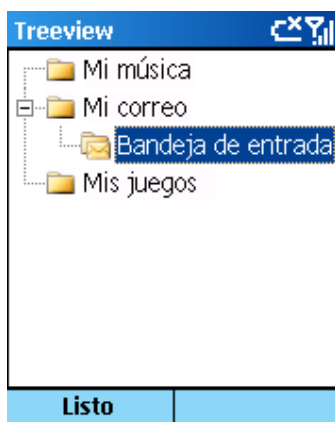


Figura 10. Control TreeView

ScrollBars

Estos controles se implementan con el mismo estilo que en Pocket PC. No proporcionan un desplazamiento automático; en vez de ello, se pueden establecer mediante programación las posiciones mínima, máxima y actual de la barra de desplazamiento. Aunque se puede generar un evento si cambia la posición, corresponde a la aplicación volver a dibujar el contenido del

formulario en la nueva posición. Para ver un ejemplo, consulte la sección Formularios de desplazamiento.

Timer

No es un control visual, sino que se debe utilizar desde un formulario de ventanas. Se utiliza para generar un evento a intervalos definidos.

ProgressBar

El control ProgressBar puede ser útil para mantener al usuario informado acerca del estado de una tarea de larga duración. Consulte la figura 11. La implementación de este control es igual que en Pocket PC.

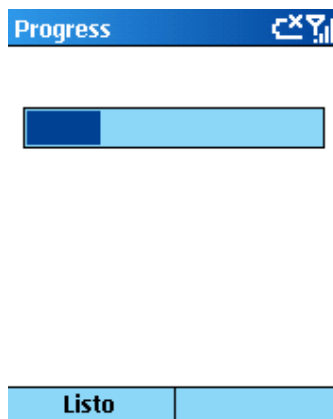


Figura 11. Barra de progreso

ImageList

Este control contiene una lista de imágenes que se pueden utilizar en toda la aplicación. Los controles TreeView y ListView están diseñados para trabajar directamente con este control. Se pueden agregar imágenes a través del editor en tiempo de diseño; éste incrusta de forma automática la imagen como un recurso, por lo que no es necesario distribuirla con la aplicación. Consulte la figura 12.

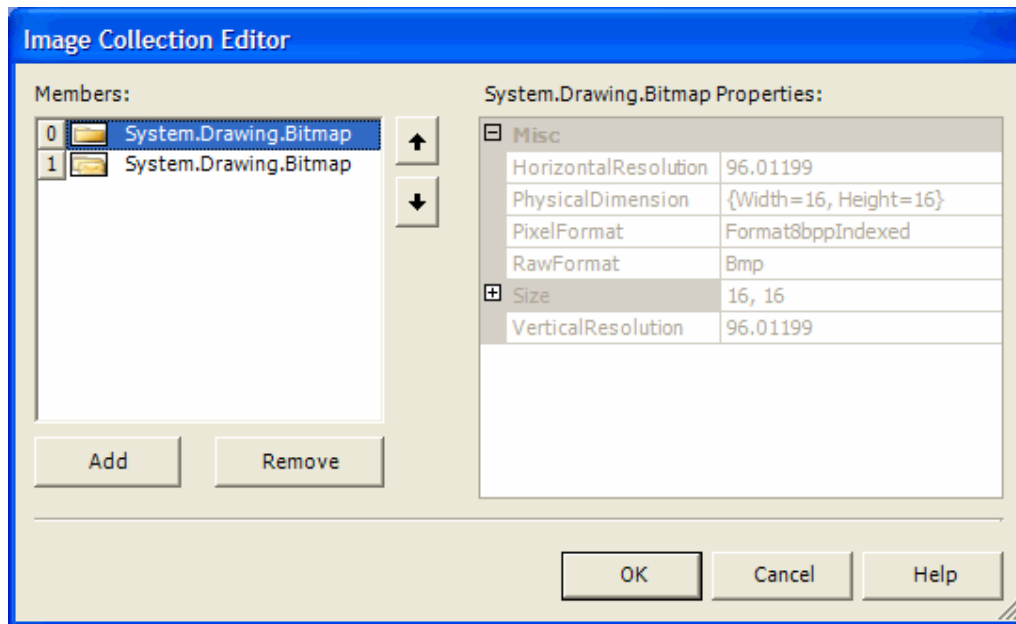


Figura 12. Adición de imágenes en tiempo de diseño

También se pueden agregar imágenes mediante programación, por ejemplo, si las imágenes se cargan de forma dinámica desde un almacén de datos.

Actividad

Desarrollar la interfaz de una aplicación móvil usando algunos de los controles mostrados en el capítulo anterior

Para esta actividad crear un nuevo proyecto en el Visual Studio llamado Ejminterface, crear los formularios llamados Form_Ingreso, Form_Principal, Form_Producto y Form_Proveedor.

Formulario de Ingreso



```
Public Class Form_Ingreso

    Private Sub Llenar_Usuarios()
        'Metodo para llenar el ComboBox de Usuarios

        ComboBox_User.Items.Add("jnolasco")

        ComboBox_User.Items.Add("avega")

        ComboBox_User.Items.Add("rvillanueva")

        ComboBox_User.SelectedIndex = 0

    End Sub

    Private Sub Button_Ingresar_Click(ByVal sender As System.Object, _
        ByVal e As System.EventArgs) Handles Button_Ingresar.Click

        'Validacion de Campos Vacíos
        If ComboBox_User.Text = "" Or TextBox_Pass.Text = "" Then

            MessageBox.Show("Debe llenar todos los campos", _
                "Advertencia", MessageBoxButtons.OK, _
                MessageBoxIcon.Exclamation, _
                MessageBoxDefaultButton.Button1)

        Else
            'Verificacion de Password
            If TextBox_Pass.Text = "123" Then
```

```
        Dim frm2 As New Form_Principal

        TextBox_Pass.Text = ""

        ComboBox_User.Text = ""

        Form_Principal.Show()

    Else

        MessageBox.Show("Password invalido", "Advertencia", _
        MessageBoxButtons.OK, _
        MessageBoxIcon.Exclamation, _
        MessageBoxDefaultButton.Button1)

    End If

End If

End Sub

Private Sub Button_Salir_Click(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles Button_Salir.Click

    If (MessageBox.Show("¿Desea salir?", "Advertencia", _
    MessageBoxButtons.YesNo, MessageBoxIcon.Exclamation, _
    MessageBoxDefaultButton.Button1) = _
    Windows.Forms.DialogResult.Yes) Then

        Application.Exit()

    End If

End Sub

Private Sub Form_Ingreso_Load(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles MyBase.Load

    Llenar_Usuarios()

End Sub

End Class
```

Formulario Principal



```
Public Class Form_Principal

    Private Sub Button_Productos_Click(ByVal sender As System.Object _
    , ByVal e As System.EventArgs) Handles Button_Producto.Click

        'Llama al formulario productos
        Form_Productos.Show()

    End Sub

    Private Sub Button_proveedor_Click(ByVal sender As System.Object _
    , ByVal e As System.EventArgs) Handles Button_Proveedor.Click

        'Llama al formulario proveedores
        Form_Proveedores.Show()

    End Sub

    Private Sub Button_Salir_Click(ByVal sender As System.Object _
    , ByVal e As System.EventArgs) Handles Button_Salir.Click

        If (MessageBox.Show("¿Deseas salir", "Advertencia", _
        MessageBoxButtons.YesNo, MessageBoxIcon.Exclamation, _
        MessageBoxDefaultButton.Button1) = _
        Windows.Forms.DialogResult.Yes) Then

            Me.Close()

        End If

    End Sub

End Class
```

Formulario Productos



Nota: Agregar al TreeView los nodos principales entrada, plato de fondo, postre bebida en la propiedad Nodes

```
Public Class Form_Productos
    'Variable nodo de tipo TreeNode
    Dim nodo As New TreeNode

    Private Sub Form_Productos_Load(ByVal sender As System.Object _
        , ByVal e As System.EventArgs) Handles MyBase.Load

        Llenar_Tipos()

    End Sub

    Private Sub Button_Cancelar_Click(ByVal sender As System.Object _
        , ByVal e As System.EventArgs) Handles Button_Cancelar.Click

        TextBox_Descrip.Text = ""

        ComboBox_Tipo.SelectedIndex = 0

    End Sub

    Private Sub Button_Registrar_Click(ByVal sender As System.Object _
        , ByVal e As System.EventArgs) Handles Button_Registrar.Click

        If TextBox_Descrip.Text = "" Then
```



```

        MessageBox.Show("Debe llenar todos los campos", "Aviso", _
        MessageBoxButtons.OK, MessageBoxIcon.Exclamation, _
        MessageBoxDefaultButton.Button1)

Else

    'De acuerdo al indice seleccionado del combobox
    'se manda la variable al metodo Llenar_Nodos()
    If ComboBox_Tipo.SelectedIndex = 0 Then

        Llenar_Nodos(0)

    ElseIf ComboBox_Tipo.SelectedIndex = 1 Then

        Llenar_Nodos(1)

    ElseIf ComboBox_Tipo.SelectedIndex = 2 Then

        Llenar_Nodos(2)

    ElseIf ComboBox_Tipo.SelectedIndex = 3 Then

        Llenar_Nodos(3)

    End If

End If

End Sub

Private Sub Button_Regresar_Click(ByVal sender As System.Object _
, ByVal e As System.EventArgs) Handles Button_Regresar.Click

    If (MessageBox.Show("¿Desea regresar?", "Advertencia", _
    MessageBoxButtons.YesNo, MessageBoxIcon.Exclamation, _
    MessageBoxDefaultButton.Button1) = _
    Windows.Forms.DialogResult.Yes) Then

        Me.Hide()

    End If

End Sub

Private Sub Llenar_Nodos(ByVal x As Integer)
    'Metodo para llenar los subnodos del TreeView de Productos
    'donde x sera el numero de nodo del TreeView

    'Se cierran los nodos del TreeView
    TreeView_Prod.CollapseAll()

    'Se iguala la variable TreeNode al Nodo principal del TreeView
    'establecido por la variable x
    nodo = TreeView_Prod.Nodes(x)

    'Se aumenta un subnodo al nodo principal
    nodo.Nodes.Add(TextBox_Descrip.Text)

    MessageBox.Show("Producto registrado", "Aviso", _
    MessageBoxButtons.OK, MessageBoxIcon.Exclamation, _
    MessageBoxDefaultButton.Button1)

```

```

Tab_Producto.SelectedIndex = 1

'Se expande el nodo principal seleccionado del TreeView
nodo.Expand()

TextBox_Descrip.Text = ""

ComboBox_Tipo.Text = ""

End Sub

Private Sub Llenar_Tipos()
'Metodo para llenar el ComboBox de Tipos de productos

ComboBox_Tipo.Items.Add("Entrada")

ComboBox_Tipo.Items.Add("Plato de Fondo")

ComboBox_Tipo.Items.Add("Postre")

ComboBox_Tipo.Items.Add("Bebidas")

ComboBox_Tipo.SelectedIndex = 0

End Sub
End Class

```

Formulario Proveedores



Nota: Agregar al ListView las columnas descripción y Ruc en la propiedad Columns.

```
Public Class Form_Proveedores

    Private Sub Button_Registrar_Click(ByVal sender As System.Object _
    , ByVal e As System.EventArgs) Handles Button_Registrar.Click

        'Se validan los campos vacios
        If TextBox_Descrip.Text = "" Or TextBox_Ruc.Text = "" Then

            MessageBox.Show("Debe llenar todos los campos", "Aviso", _
            MessageBoxButtons.OK, MessageBoxIcon.Exclamation, _
            MessageBoxDefaultButton.Button1)

        Else

            'Variable de tipo ListViewItem
            Dim ListView_Item As ListViewItem

            'Se crea el Item Principal para el ListView
            ListView_Item = New ListViewItem(TextBox_Descrip.Text)

            'Se añade el subitem al Item Principal del ListView
            ListView_Item.SubItems.Add(TextBox_Ruc.Text)

            'Se añade el Item principal al ListView
            Listview_Prov.Items.Add(ListView_Item)

            MessageBox.Show("Proveedor registrado", "Aviso", _
            MessageBoxButtons.OK, MessageBoxIcon.Exclamation, _
            MessageBoxDefaultButton.Button1)

            Tab_Prov.SelectedIndex = 1

            Borrar_Campos()

        End If

    End Sub

    Private Sub Button_Cancelar_Click(ByVal sender As System.Object _
    , ByVal e As System.EventArgs) Handles Button_Cancelar.Click

        Borrar_Campos()

    End Sub

    Private Sub Borrar_Campos()

        TextBox_Descrip.Text = ""

        TextBox_Ruc.Text = ""

    End Sub

End Class
```

```

Private Sub Button_Regresar_Click(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles Button_Regresar.Click

    If (MessageBox.Show("¿Deseas regresar?", "Advertencia", _
        MessageBoxButtons.YesNo, MessageBoxIcon.Exclamation, _
        MessageBoxDefaultButton.Button1) = _
        Windows.Forms.DialogResult.Yes) Then

        Me.Hide()

    End If

End Sub


End Class


```


Autoevaluación

- ¿El Control Datagrid se encuentra disponible para un proyecto Smartphone?

Para recordar

 El diseño del formulario puede desplazarse verticalmente si hay demasiados controles que incluir en el formulario; no obstante, siempre deberá ajustarse de forma horizontal.

 Si desea saber más acerca de estos temas, puede consultar las siguientes páginas.

 <http://www.microsoft.com/spanish/msdn/articulos/archivo/060204/voices/grfCraftingSmartphone.aspx>

**UNIDAD DE
APRENDIZAJE**

1

SEMANA

5

Creación de controles

OBJETIVOS ESPECÍFICOS

- Desarrollar controles personalizados con .NET Compact Framework.

CONTENIDOS

- Desarrollo de controles personalizados.
- Agregar controles personalizados al cuadro de herramientas
- Técnicas y consideraciones sobre controles personalizados

ACTIVIDAD

- Desarrollar un control personalizado utilizando un Label y un TextBox.

1. Desarrollo de controles personalizados

.NET Compact Framework proporciona controles de formularios Windows Forms que pueden satisfacer las necesidades de la mayoría de los proyectos para dispositivos. Estos controles están diseñados para utilizar recursos limitados de la manera más eficiente posible y, por lo tanto, no son compatibles con todos los métodos, propiedades y eventos heredados. Para satisfacer las funcionalidades restringidas, puede derivar sus propios controles personalizados de los controles comunes. Un control personalizado puede crearse definiendo un tipo público que herede de la clase **Control** o de un **UserControl** existente en su ensamblado.

La personalización más sencilla de un control consiste en reemplazar un método de un control común. Por ejemplo, puede reemplazar el método **OnKeyPress** por un control **TextBox** para tener un código que sólo acepte la entrada de caracteres numéricos.

Puede derivar de controles comunes para:

- Reemplazar métodos, propiedades y eventos con los suyos propios en controles comunes.
- Definir métodos, propiedades y eventos adicionales para un control.
- Generar un control compuesto, como una colección de controles **TextBox** y **Button**.
- Definir la respuesta de un control a las acciones del usuario, como un **TextBox** que sólo acepta datos numéricos.

.NET Compact Framework por el momento no ofrece la capacidad de agregar un control personalizado que se encuentre fuera del proyecto para acceso en tiempo de diseño, salvo que el control fuese creado en el mismo proyecto.

2. Agregar controles personalizados al cuadro de herramientas

Cuando crea un proyecto para un control personalizado en Microsoft Visual Studio 2008, el control se agrega automáticamente al **Cuadro de herramientas** al compilar la aplicación. Puede crear un control personalizado utilizando uno de los tipos de proyecto siguientes:

- Biblioteca de controles
- Biblioteca de clases
- Biblioteca de clases (1.0)

Para agregar un control personalizado al Cuadro de herramientas, haga clic en **Elegir elementos del cuadro de herramientas** en el menú Herramientas. Puede buscar a continuación el ensamblado de control.

3. Técnicas y consideraciones sobre controles personalizados

Tenga en cuenta lo siguiente al crear controles personalizados:

- .NET Compact Framework no admite los valores heredados de un control primario, como es posible con algunos controles en la versión completa de .NET Framework. Para evitar este problema, puede utilizar el método **OnParentChanged** para determinar si se cambian los controles primarios y determinar las acciones adecuadas. El ejemplo de código siguiente muestra el cambio de color de fondo cuando cambia el primario:

```
Protected Overrides Sub OnParentChanged(ByVal e As EventArgs)
    MyBase.OnParentChanged(e)
    Me.BackColor = Parent.BackColor
End Sub
```

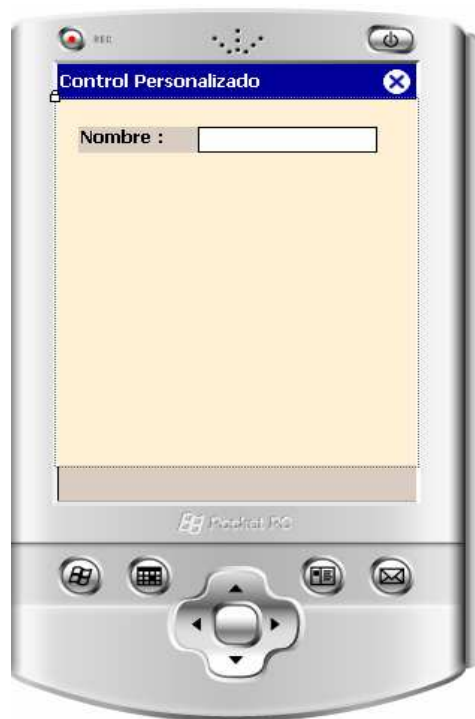
- .NET Compact Framework no admite la herencia de la fuente de un control primario.
- Para detectar la ubicación de un control personalizado donde se ha mostrado un menú contextual mediante una acción de "arrastrar y tocar", controle el evento **Popup** y, a continuación, consulte las coordenadas del mouse actuales mediante la propiedad **MousePosition**.
- Para saltar de un control personalizado al control anterior, utilice el código siguiente del controlador de eventos **KeyDown** cuando se detecte una clave **Up**.

```
Me.Parent.Controls( _  
    Me.Parent.GetChildIndex(customcontrol) - 1).Focus
```


Actividad

Desarrollar un control personalizado utilizando un Label y un TextBox, el cual tendrá las propiedades:

- Elegir si se desea que el TextBox solo acepte números.
- En el caso que se desee que el TextBox acepte letras, establecer que se muestren en mayúsculas y minúsculas.
- Dicho control personalizado ahorrara tiempo al desarrollador al jalar ambos controles (Label y TextBox) a la misma vez y de forma alineada.



```
Imports System
Imports System.Collections
Imports System.ComponentModel
Imports System.Data
Imports System.Drawing
Imports System.Text

Public Class UCLabelText

    Private _isNumeric As Boolean
    Private mCaseType As CaseType = CaseType.Normal

    Private Sub LabelText_Resize _
        (ByVal sender As Object, ByVal e As System.EventArgs)

        Me.Height = Me.Panel_TextBox.Height
    End Sub

    '
    ' propiedades públicas
    '

    'Propiedad Para Cambiar el Texto del Label
    Public Property LabelCaption() As String

        Get
            Return Me.Label.Text
        End Get

        Set(ByVal value As String)

            Me.Label.Text = value
        End Set
    End Property

    'Propiedad Para Cambiar el Texto del TextBox
    Public Property TextCaption() As String

        Get
            Return Me.TextBox.Text()
        End Get

        Set(ByVal value As String)

            Me.TextBox.Text = value
        End Set
    End Property
End Class
```

```

'Propiedad Para Cambiar Ancho del Panel_Label
'La cual afectara el tamaño del label

Public Property LabelWidth() As Integer
    Get
        Return Panel_Label.Width

    End Get
    Set(ByVal value As Integer)
        ' el ancho de la etiqueta será como máximo
        ' 3/4 partes del ancho total del control
        Panel_Label.Width = CInt(Math.Min(value, Me.Width * 0.75))

    End Set
End Property

'Propiedad Para Cambiar Ancho del Panel_TextBox
'La cual afectara el tamaño del TextBox y del Label

Public Property TextWith() As Integer
    Get
        Return Panel_TextBox.Width

    End Get
    Set(ByVal value As Integer)
        Panel_TextBox.Width = value
        Me.Panel_Label.Width = Me.Width - value

    End Set
End Property

' Propiedad para el alineamiento del texto del label
' TopLeft,TopCenter y TopRight.
Public Property LabelTextAlign() As ContentAlignment
    Get
        Return Me.Label.TextAlign

    End Get

    Set(ByVal value As ContentAlignment)

        Me.Label.TextAlign = value

    End Set
End Property

' Array con los dígitos aceptados por el textbox
' Solo cuando la propiedad isNumeric sea True

Private digitos As Char() = {"0", "1", "2", "3", "4", _
    "5", "6", "7", "8", "9", ".", ",", "-", "+"}

' Esta función permite controlar si el caracter es de los
admitidos
Protected Function CaracterCorrecto(ByVal c As Char) As Boolean

```

```
        If Asc(c) = 8 Then
            Return True
        Else
            ' devolverá true si el caracter está en el array
            Return (Array.IndexOf(digitos, c) <> -1)
        End If
    End Function

    ' Propiedad para definir si tu TextBox aceptara solo numeros.
    Public Property isNumeric() As Boolean
        Get
            Return _isNumeric
        End Get
        Set(ByVal value As Boolean)
            _isNumeric = value
        End Set
    End Property

    Private Sub txtbox_KeyPress(ByVal sender As System.Object, _
        ByVal e As System.Windows.Forms.KeyPressEventArgs) Handles
        TextBox.KeyPress

        If isNumeric Then
            If Not CharacterCorrecto(e.KeyChar) Then
                e.Handled = True
            End If
        Else
            UpdateTextCase()
        End If
    End Sub

    Public Enum CaseType
        Normal
        Upper
        Lower
    End Enum

    'Propiedad para elejir si el texto va ser MAYUSCULA,miniscula o
    normal.
    Public Property TextCase() As CaseType
        Get
            Return mCaseType
        End Get
    End Property
```

```
        Set(ByVal value As CaseType)
            mCaseType = value
        End Set
    End Property

    'Metodo que cammbia el texto digitado en mayusculas o en
    minusculas

    Private Sub UpdateTextCase()

        'Dim sControlText As String = Me.TextBox.Text

        'Se captura la posicion del cursor del TextBox

        Dim cursorPosition As Integer = TextBox.SelectionStart()

        Select Case (Me.TextCase)

            Case CaseType.Lower

                'Cambio del texto a minusculas

                Me.TextBox.Text = TextBox.Text.ToLower()

            Case CaseType.Normal

                'no se realiza ni un cambio

            Case CaseType.Upper

                'Cambio del texto a mayusculas

                Me.TextBox.Text = TextBox.Text.ToUpper()
            Case Else

        End Select

        'Se regresa la posicion del cursor del TextBox.

        TextBox.SelectionStart = cursorPosition

    End Sub

    Private Sub TextBox_TextChanged(ByVal sender As System.Object, _
        ByVal e As System.EventArgs) Handles TextBox.TextChanged


        UpdateTextCase()
    End Sub


End Class
```


Autoevaluación

- ¿Qué tipos de proyectos se pueden usar para crear controles personalizados?

Para recordar

 La personalización más sencilla de un control consiste en reemplazar un método de un control común. Por ejemplo, puede reemplazar el método OnKeyPress por un control TextBox para tener un código que sólo acepte la entrada de caracteres numéricos.

 Si desea saber más acerca de estos temas, puede consultar las siguientes páginas.

 [http://msdn.microsoft.com/es-es/library/4yf3whkx\(VS.80\).aspx](http://msdn.microsoft.com/es-es/library/4yf3whkx(VS.80).aspx)

**UNIDAD DE
APRENDIZAJE****1****SEMANA****6**

Conectividad a redes

OBJETIVOS ESPECÍFICOS

- Conocer los medios de comunicación adecuados para un dispositivo móvil

CONTENIDOS

- Programación para redes en .NET Compact Framework.
- Solicitudes HTTP.
- Comunicaciones móviles seguras.
- Consideraciones acerca de la longitud del contenido,
- Programación de Sockets.

ACTIVIDAD

- Desarrollar una aplicación móvil que pueda conectarse a una pagina web y realizar una solicitud http.

1. Programación para redes en .NET Compact Framework

.NET Compact Framework proporciona compatibilidad integrada con servicios Web XML, así como con las siguientes funcionalidades del protocolo:

- Controles basados en HTTP.
- Autenticación de NTLM.
- Contenidos XML codificados en SOAP. Esta compatibilidad incluye la transferencia de conjuntos de datos ADO.NET.
- Métodos de peticiones Web y respuestas Web que pueden enviar mensajes HTTP SOAP y recibir en respuesta mensajes SOAP.
- Bibliotecas y métodos SOAP que puedan serializar y deserializar llamadas a métodos y objetos arbitrarios en mensajes XML SOAP y desde éstos.

2. Solicitudes HTTP

Los elementos siguientes se refieren a enviar y recibir solicitudes HTTP:

- Cuando utilice el emulador, no use localhost como nombre de servidor. Especifique el nombre de equipo o la dirección IP del equipo de desarrollo que aloja el servicio Web.
- El emulador, como los dispositivos, tiene su propia dirección IP. El uso de localhost indica al emulador que se utilice a sí mismo para conectarse con el servicio Web, en lugar del servicio Web alojado por el equipo de desarrollo u otro equipo de escritorio.

Por ejemplo, en lugar de:

`http://localhost/myWebService/Service1.asmx`

Especifique lo siguiente:

`http://ComputerName/myWebService/Service1.asmx`

- Cuando realice una solicitud HTTP mediante **HttpRequest**, el dispositivo iniciará una conexión de red nueva, si no estaba establecida antes. Por lo tanto, al realizar una petición de HTTP solamente para averiguar si está disponible una conexión, puede hacer que el dispositivo intente iniciar una conexión, por ejemplo, una conexión GPRS.
- .NET Compact Framework no almacena información del proxy en la propiedad **System.Net.GlobalProxySelection.Select**, pero utiliza esta propiedad para las conexiones HTTP si especifica un valor en el código.
- Es probable que, para las conexiones con Internet, deba especificar su propia configuración de proxy local. El código siguiente muestra la configuración del proxy para el puerto 80:

```
System.Net.GlobalProxySelection.Select=new_  
WebProxy("http://myproxy:80");
```

- Si establece **AllowWriteStreamBuffering** en **false**, los datos no se almacenarán en el búfer y no serán compatibles con las solicitudes de autenticación ni las redirecciones del servidor Web.
- Para asegurarse de que las operaciones sean satisfactorias, especifique rutas de acceso absolutas.

Tenga en cuenta el comportamiento de Windows CE siguiente para resolver las especificaciones relativas al archivo:

file://myfile se resuelve como \\myfile.

file:///myfile se resuelve como \myfile en el directorio raíz.

- Es un problema conocido que el método **System.Net.Dns.GetHostName** de .NET Framework produce una excepción cuando hay más de 50 protocolos de red instalados en el equipo actual.

Para evitar este problema, desinstale los protocolos de red que realmente no se necesitan. Una manera de hacerlo es utilizar el Administrador de dispositivos de Windows para quitar los adaptadores de red que no se

utilizan. Otra manera es desinstalar aplicaciones que tienen protocolos instalados.

3. Comunicaciones móviles seguras

Hay dos métodos principales para obtener comunicaciones móviles seguras:

- ***Autenticación HTTP***

.NET Compact Framework admite autenticación básica e implícita. Estos mecanismos de autenticación son sencillos y se conoce bastante bien su seguridad y compensaciones, como por ejemplo que el servicio Web se limita a un enlace de HTTP.

La versión 3.5 de .NET Compact Framework admite servidores que ejecutan NTLM o Kerberos (Autenticación Integrada de Windows), lo que no requerirá ningún cambio de código de la autenticación básica o implícita actual.

- ***Encabezados de seguridad personalizados***

Actualmente, .NET Compact Framework no admite Seguridad de Servicios Web (WS-Security) ni Mejoras del Servicio Web (WSE).

Además, ya sea que autentique utilizando HTTP o un encabezado personalizado, puede utilizar SSL para aumentar la seguridad. La autenticación básica pasa el nombre y la contraseña en texto no cifrado, por lo que no es particularmente seguro a menos que se ejecute desde dentro de SSL. Sin embargo, cuando se utiliza junto con SSL, es bastante seguro, con el único problema significativo de que se revelen accidentalmente las credenciales al servidor de destino.

4. Consideraciones acerca de la longitud del contenido

Si envía una petición Web HTTP de contenidos enviados como secuencias de datos mediante el protocolo POST, deberá especificar una longitud para el contenido. Si **SendChunked** es **false** y **Method** = POST, especifique un valor para **ContentLength**.

A diferencia de .NET Framework completo, .NET Compact Framework no almacena previamente los datos en el búfer por consideraciones de limitaciones de memoria. Para garantizar el almacenamiento en búfer, establezca **SendChunked** en **false**.

5. Programación de sockets

Los elementos siguientes se refieren a la programación de sockets.

- No todas las opciones de socket son compatibles con todos los sistemas operativos del dispositivo.

El diseño de .NET Compact Framework le permite ser trasladado a cualquier número de sistemas operativos, cada uno con sus propios niveles de funcionalidad. Por consiguiente, .NET Compact Framework no limita artificialmente la disponibilidad de opciones de socket en función del nivel de compatibilidad determinado de un sistema operativo.

- Los sockets causan problemas conocidos en los Pocket PC que ejecutan Windows CE 3.0.

Si cierra un socket que contiene datos no enviados de una llamada **Send** anterior, los datos se perderán o se dañarán.

Si acepta un socket y después cierra el socket enlazado antes de cerrar el aceptado, no podrá enlazar con el puerto hasta que finalice el tiempo de espera de aproximadamente 4,5 minutos.

- En las aplicaciones de .NET Compact Framework, se admiten las siguientes opciones, pero no funcionan si no se modifica la pila de TCP/IP y en la actualidad se reservan para su uso futuro: **AcceptConnection**, **ReceiveLowWater**, **ReceiveTimeout**, **SendLowWater**, **SendTimeout** y **Type**.

Actividad

Desarrollar una aplicación móvil que pueda conectarse a una pagina web y realizar una solicitud http.

Para esto crearemos un website con el siguiente código en su index:

Website

```
Partial Class _Default

    Inherits System.Web.UI.Page

    Private _user, _password As String

    'Se llenan los datos internas para realizar validacion

    Private usuarios As String() = _
        {"rvillanueva", "jnolasco", "avega"}

    Private passwords As String() = _
        {"123", "456", "789"}

    Private saldos As String() = _
        {"675.50", "1050.10", "800.25"}

    Private mensaje As String

    Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Load

        'Se crean los parametros para realizar validacion

        _user = Request.QueryString("varUsuario")

        _password = Request.QueryString("varPassword")

        Dim CantidadUser As Integer

        CantidadUser = usuarios.Length

        Dim i As Integer

        Dim z As Integer
```

```
Dim fila As Integer

Dim usuario As Boolean

usuario = False

fila = 0

For i = 0 To CantidadUser - 1

    'Se verifica si el usuario y el password mandado
    'son correctos

    If _user = usuarios(i) Then

        fila = i
        usuario = True

        Exit For

    Else
        mensaje = "Usuario Incorrecto"
    End If

Next

If usuario Then

    For z = 0 To CantidadUser - 1

        If _password = passwords(fila) Then

            mensaje = saldos(fila)

            Exit For

        Else
            mensaje = "Contraseña Incorrecta"
        End If

    Next

End If

Response.Write(mensaje)

End Sub

End Class
```

Luego de esto se publica la pagina web en el Internet Information Server

Aplicación Móvil



Formulario de Consulta

```
Public Class Frm_Consultar

    Private _Request As System.Net.HttpWebRequest = Nothing

    Private _Response As System.Net.HttpWebResponse = Nothing

    Private result As String

    Private sr As IO.StreamReader

    Private Sub Button_Consultar_Click( _
        ByVal sender As System.Object, ByVal _
        e As System.EventArgs) Handles Button_Consultar.Click

        'Validacion si los Texbox se encuentran vacias

        If TextBox_User.Text = "" Or _
            TextBox_Password.Text = "" Then

            MsgBox("Falta digitar Datos", _
                MsgBoxStyle.Exclamation, "Digitar. . .")

        Else
```

```
        TextBox_Saldo.Text = GetRespuestaWebPage()

    End If

End Sub

Public Function GetRespuestaWebPage() As String

    Try

        _Request = Net.WebRequest.Create( _
            "http://192.168.1.80/WebSiteEjemplo/" & _
            "Index.aspx?varUsuario=" + TextBox_User.Text + _
            "&" + "varPassword=" + TextBox_Password.Text)

        'Obtiene o establece el tiempo que transcurre
        'Hasta que se agota el tiempo de espera de la solicitud.

        _Request.Timeout = 100000

        'Obtiene o establece la versión de HTTP que
        'se va a utilizar para la solicitud

        _Request.ProtocolVersion = Net.HttpVersion.Version11

        ' Se indica si se va a realizar o no una
        'conexión persistente a los recursos de Internet.

        _Request.KeepAlive = False

        'Se solicita una respuesta.

        _Response = _Request.GetResponse()

        'Verificamos si hubo respuesta

        If (_Response Is Nothing) Then

            result = "Error en la conexion"

        Else

            'Verificamos si el estado de la respuesta es OK
            If _Response.StatusCode = _
                System.Net.HttpStatusCode.OK Then

                'Obtiene el cuerpo de la respuesta
                'del servidor en una instancia de Stream

                sr = New IO.StreamReader _
                    (_Response.GetResponseStream())

                'Obtengo todo el contenido que tiene el Stream

                result = sr.ReadToEnd

            End If

        End If

    Catch ex As Exception

        result = "Error al obtener la respuesta"

    End Try

End Function
```



```
        End If

    End If

    Catch ex As Net.WebException

        result = ex.Message.ToString

    Finally

        'Cerramos y limpiamos las variables

        If Not (_Response Is Nothing) Then

            _Response.Close()
            _Response = Nothing

        End If

        If Not (_Request Is Nothing) Then

            _Request.Abort()
            _Request = Nothing

        End If

        sr.Close()

    End Try

    Return result


End Function


End Class
```


Autoevaluación

- ¿Las opciones del Socket trabajan adecuadamente en todos los sistemas operativos móviles?

Para recordar

 A diferencia de .NET Framework completo, .NET Compact Framework no almacena previamente los datos en el búfer por consideraciones de limitaciones de memoria. Para garantizar el almacenamiento en búfer, establezca SendChunked en false.

 Si desea saber más acerca de estos temas, puede consultar las siguientes páginas.

 <http://msdn.microsoft.com/es-es/library/1afx2b0f.aspx>

 <http://msdn.microsoft.com/es-es/library/ms172494.aspx>

**UNIDAD DE
APRENDIZAJE****2****SEMANA****9**

Introducción al modelo de datos - SQL Server CE

OBJETIVOS ESPECÍFICOS

- Conocer el modelo de datos SQL Server CE.

CONTENIDOS

- SQL Server CE.
- Instalar componentes de SQL Server CE.
- Información general y escenarios de SQL Server Compact.
- Características nuevas de SQL Server Compact.
- Integración con SQL Server

1. SQL Server CE

SQL Server Compact permite crear bases de datos compactas que se pueden implementar en equipos de escritorio y dispositivos inteligentes. SQL Server Compact comparte un modelo de programación común con otras versiones de SQL Server para desarrollar tanto aplicaciones administradas como nativas. SQL Server Compact proporciona funcionalidad de bases de datos relacionales: un almacén de datos sólido, un procesador de consultas de optimización y una conectividad confiable y escalable.

Visual Studio proporciona compatibilidad para el desarrollo con SQL Server Compact. SQL Server proporciona compatibilidad para la administración de bases de datos. Para tener acceso a las bases de datos de SQL Server Compact almacenadas en un dispositivo inteligente o en un equipo de escritorio, puede usar SQL Server Management Studio en SQL Server. Puede crear aplicaciones administradas utilizando Microsoft Visual Basic o Visual C#, o puede utilizar Visual C++ para crear aplicaciones nativas.

2. Instalar componentes de SQL Server Compact

Los componentes de SQL Server Compact están disponibles en los archivos de Microsoft Windows Installer (.msi) siguientes:

- Herramientas de diseño de SQL Server Compact (SSCEVSTools-ENU.msi)
- SQL Server Compact Runtime (SSCERuntime-ENU.msi)
- SQL Server Compact para dispositivos (SSCEDeviceRuntime-ENU.msi)
- Herramientas de consultas de SQL Server Compact (SSCESqlWbTools-ENU.msi)

- Herramientas de servidor de SQL Server Compact (SSCEServerTools-ENU.msi)
- Libros en pantalla de SQL Server Compact (SSCEBOL-ENU.msi)

3. Información general y escenarios de SQL Server Compact

SQL Server Compact proporciona las características siguientes cuando se usa como origen de datos local para las aplicaciones.

- SQL Server Compact está basado en archivos, lo que significa que la cadena de conexión es una ruta de acceso al archivo de base de datos (.sdf).
- SQL Server Compact no se ejecuta como un servicio. Ésta es una de las principales diferencias entre usar SQL Server Compact y usar SQL Server o SQL Server Express.
- SQL Server Compact admite un máximo de 256 conexiones. También es posible abrir conexiones en procesos diferentes.
- SQL Server Compact admite archivos de base de datos de hasta 4 GB.

SQL Server Compact es una base de datos basada en archivos que consta de varias DLL que ocupan 1,4 MB, aproximadamente. En la lista siguiente se proporcionan algunas situaciones en las que puede ser conveniente usar SQL Server Compact en las aplicaciones:

- En aplicaciones diseñadas para equipos de escritorio y dispositivos móviles.
- En aplicaciones que se vayan a usar ocasionalmente en escenarios conectados.
- Cuando necesite una base de datos que sea redistribuible de forma gratuita.
- Cuando los requisitos de tamaño y memoria de las aplicaciones deban reducirse.
- Cuando desee que el código de acceso a los datos se ejecute en un proceso.

4. Características nuevas de SQL Server Compact

En las secciones siguientes se describen las nuevas características de SQL Server Compact 3.5 y SQL Server Compact 3.5 SP1.

4.1 SQL Server Compact 3.5

SQL Server Compact versión 3.5 se comercializó con Visual Studio 2008. A partir de SQL Server Compact versión 3.5:

- SQL Server Compact admite el ámbito de transacciones locales en equipos de escritorio.
- El Diseñador de tablas de SQL Server Compact en Visual Studio 2008 se ha mejorado para proporcionar una interfaz de usuario que permite crear relaciones de clave principal y clave externa entre tablas.
- SQL Server Compact admite ahora las instrucciones de Transact-SQL siguientes:
 - Consultas anidadas en una cláusula **FROM**
 - **CROSS APPLY** y **OUTER APPLY**
 - **CAST**
 - **TOP**
 - **SET IDENTITY INSERT**
- Las aplicaciones basadas en SQL Server Compact 3.5 se pueden desarrollar para equipos de escritorio con Visual Basic 2008 Express Edition y Visual C# 2008 Express Edition.
- Puede administrar una base de datos de SQL Server Compact almacenada en un dispositivo inteligente o en un equipo de escritorio con SQL Server Management Studio Express (SSMSE) en SQL Server 2008.

- SQL Server Compact implementa el tipo de datos **timestamp** (rowversion).
- SQL Server Compact admite LINQ to SQL. LINQ to SQL es un componente del proyecto Language Integrated Query (LINQ). Proporciona una infraestructura de tiempo de ejecución para administrar datos relacionales como objetos sin renunciar a la capacidad de consulta. Traduce las consultas de LINQ en Transact-SQL para que las ejecute SQL Server Compact y, después, traduce de nuevo los resultados tabulares en los objetos definidos por el programador de la aplicación.

4.2 SQL Server Compact 3.5 SP1

SQL Server Compact 3.5 Service Pack 1 (SP1) está incluido en SQL Server 2008 y en Visual Studio 2008 SP1. SQL Server Compact 3.5 SP1 proporciona varias mejoras y nuevas características para los programadores de software. La lista siguiente incluye algunas de las nuevas características entre otras:

- SQL Server Compact admite las intercalaciones con distinción entre mayúsculas y minúsculas en el nivel de base de datos.
- SQL Server Compact admite ADO.NET Entity Framework. Entity Framework permite trabajar con datos en forma de objetos y propiedades específicos del dominio, como clientes y direcciones de cliente, sin tener que preocuparse de las tablas y las columnas de las bases de datos subyacentes donde se almacenan dichos datos.
- SQL Server Compact admite LINQ to Entities. LINQ to Entities permite a los programadores crear consultas flexibles, con establecimiento inflexible de tipos, en el contexto de objetos de Entity Framework utilizando directamente las expresiones y los operadores de consulta estándar de LINQ desde el entorno de desarrollo.

- SQL Server Compact se puede ejecutar de forma nativa en un entorno de 64 bits.
- SQL Server Compact proporciona compatibilidad para la replicación de los nuevos tipos de datos en SQL Server 2008, como **date**, **time**, **datetime2**, **datetimeoffset**, **geography** y **geometry**.
- SQL Server Compact admite la replicación de datos con SQL Server 2000, SQL Server 2005 y SQL Server 2008 mediante Microsoft Synchronization Services for ADO.NET. Microsoft Synchronization Services for ADO.Net está disponible para equipos de escritorio y dispositivos móviles.
- SQL Server Compact admite la replicación de datos con SQL Server 2005 y SQL Server 2008 mediante la replicación de mezcla y el acceso a datos remotos (RDA).

5. Integración con SQL Server


Puede administrar una base de datos de SQL Server Compact en un equipo de escritorio o un dispositivo móvil usando SQL Server Management Studio, que es una herramienta de administración de SQL Server. Esta herramienta proporciona la misma experiencia para los usuarios tanto si se conectan a SQL Server como a SQL Server Compact. Las bases de datos de SQL Server Compact también se pueden administrar mediante SQL Server Management Studio Express.


Puede crear bases de datos de SQL Server Compact en el equipo local desde Management Studio y Management Studio Express. Estas bases de datos se pueden configurar, rellenar con datos y, a continuación, implementar en varios dispositivos. De este modo se ahorra mucho tiempo de desarrollo e implementación.


Autoevaluación

- ¿Qué es SQL Server CE?
- ¿Cuál es la última edición de SQL Server CE?

Para recordar

 Visual Studio proporciona compatibilidad para el desarrollo con SQL Server Compact. SQL Server proporciona compatibilidad para la administración de bases de datos.

 Si desea saber más acerca de estos temas, puede consultar las siguientes páginas.

 <http://msdn.microsoft.com/es-es/library/cc645984.aspx>

**UNIDAD DE
APRENDIZAJE****2****SEMANA****10**

ADO.NET en Compact Framework 1

OBJETIVOS ESPECÍFICOS

- Manejar adecuadamente ADO.NET en Compact Framework.

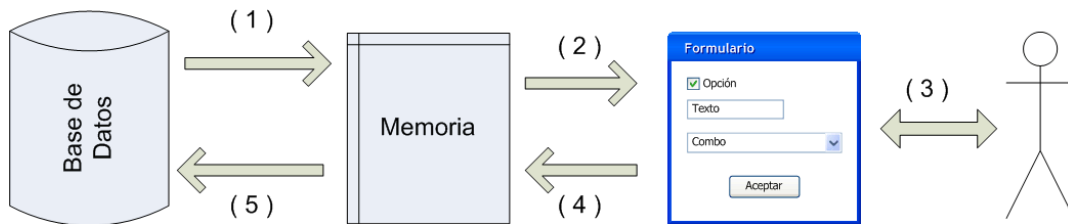
CONTENIDOS

- Arquitectura de Aplicaciones.
- Implementaciones con ADO.NET.
- Programación de las distintas implementaciones.

ACTIVIDAD

- Desarrollar una aplicación móvil que cree una base de datos y sus tablas.

1. Arquitectura de aplicaciones



1. Se cargan los registros de la base de datos a memoria.
2. Se rellena el formulario con los datos que hay en memoria.
3. El usuario ve y/o modifica el formulario.
4. Los datos que hay en el formulario se pasan a memoria.
5. Los datos en memoria se escriben en la base de datos.

2 Implementaciones con ADO.NET

- SQL
- DataSet
- DataBinding

2.1 Implementación con SQL

1. Se lee la base de datos con consultas SQL (SELECT).
2. El resultado de la consulta se procesa en memoria y se rellena el formulario.
3. El usuario ve y/o modifica el formulario.
4. Los datos que hay en el formulario se pasan a memoria y se procesan.
5. Los datos de memoria se escriben en la base de datos con sentencias SQL (INSERT, UPDATE, DELETE).

Ventajas:

- Desarrollo conocido por los desarrolladores
- Control total sobre los datos
- Control total sobre la IU
- Los datos siempre se salvan a disco

Inconvenientes:

- Lento (muchas consultas a base de datos)
- Mucha programación
- No se puede usar el DataGrid directamente
- Hay que rellenar los controles del formulario manualmente
- Hay que pasar el contenido de los controles a la base de datos manualmente

2.2 Implementación con DataSet

1. Se carga el DataSet una única vez con una consulta SELECT. El DataSet es una “réplica” de la base de datos en memoria
2. Con los datos del DataSet se rellena el formulario
3. El usuario ve y/o modifica el formulario
4. Los datos que hay en el formulario se pasan al DataSet
5. El DataSet se vuelca completo a la base de datos automáticamente una sola vez.

Ventajas:

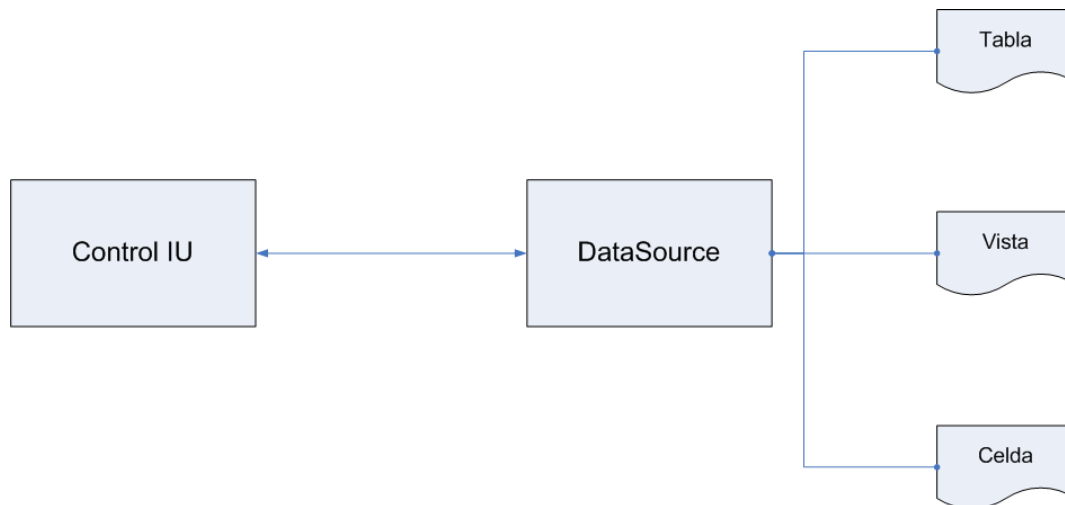
- Control total sobre la IU
- Rápido (lecturas y escrituras en memoria)

Inconvenientes:

- Mucha programación
- No se puede usar el DataGrid directamente
- Los datos están en memoria hasta que se salvan
- No se controlan totalmente los datos (ejemplo: DataRow.Delete() marca la fila como borrada pero no la quita del DataTable)
- Hay que rellenar los controles del formulario manualmente
- Hay que pasar el contenido de los controles a la base de datos manualmente

2.3 Implementación con DataBinding

1. Se carga el DataSet una única vez con una consulta SELECT. El DataSet es una “réplica” de la base de datos en memoria.
2. Se asocian los controles del formulario a una celda, tabla o vista del DataSet. El control se rellena automáticamente.
3. El usuario ve y/o modifica el formulario.
4. La celda, tabla o vista se actualiza automáticamente con las modificaciones del usuario en el formulario.
5. El DataSet se vuelca completo a la base de datos automáticamente.



Ventajas:

- Rápido (lecturas y escrituras en memoria)
- Se puede usar el DataGrid
- Muy poca programación (sólo hay que hacer el binding, y la sincronización es automática)

Inconvenientes:

- No se controlan los datos (ejemplo: el cambio de Position hace un EndCurrentEdit())
- automático, pero el último cambio no se guarda automáticamente)
- No se controla la IU (ejemplo: es muy difícil poner una FK en un DataGrid)
- Los datos están en memoria hasta que se salvan

3. Programación de las distintas implementaciones

3.1 Referencia a la librería de SQL Server CE

Para desarrollar aplicaciones con ADO.NET hay que añadir la referencia:

`Imports System.Data.SqlServerCe`

(Pulsando en el Solution Explorer con el botón derecho sobre el proyecto
> Add Reference...)

3.2 Crear la base de datos

La creación de la base de datos y las tablas es común en todas las implementaciones:

```
Dim engine As New SqlCeEngine()  
engine.LocalConnectionString = "Data Source=My" _  
+ "Documents\Cursos.sdf"  
engine.CreateDatabase()  
Dim connection As New SqlCeConnection()  
connection.ConnectionString = "Data Source=My" _  
+ "Documents\Cursos.sdf"  
connection.Open()  
Dim command As New SqlCeCommand()  
command.Connection = connection  
command.CommandText = "CREATE TABLE Cursos " & _  
+ "(" & " id INTEGER NOT NULL CONSTRAINT PKValores" _  
+ "PRIMARY KEY," & " nombre NVARCHAR(20) NOT NULL" & " )"  
command.ExecuteNonQuery()
```

3.3 Conexión con la base de datos

La conexión con la base de datos es común a todas las implementaciones:

```
Dim connection As New SqlCeConnection()  
connection.ConnectionString = "Data Source=" _  
+ "My Documents\Cursos.sdf"
```



```
connection.Open()
```

3.3.1 Cierre de la base de datos

El cierre de la base de datos es común a todas las implementaciones:

```
connection.Close()  
connection = Nothing
```

3.4 Programación con SQL

3.4.1 Lectura de los datos

```
Dim command As New SqlCeCommand()  
command.Connection = connection  
command.CommandText = "SELECT id, nombre " & "FROM" _ +  
+ "Cursos " & "WHERE id = '4737626HX' "  
Dim reader As SqlCeDataReader = command.ExecuteReader()  
While reader.Read()  
    Dim str As String = ""  
    str += reader.GetInt32(0)  
    str += ","  
    str += reader.GetString(1)  
    MessageBox.Show(str)  
End While  
reader.Close()
```

3.4.2 Paso de datos al formulario

```
Dim cursos_cb As New ComboBox()  
  
While reader.Read()  
    Dim curso As New Curso()
```

```
curso.id = reader.GetInt32(0)
curso.nombre = reader.GetString(1)
cursos_cb.Items.Add(curso)

End While
```

3.4.3 Paso de datos desde el formulario

```
Dim curso As Curso = CType(cursos_cb.SelectedItem, Curso)
```

3.4.4 Escritura de datos

```
Dim command As New SqlCommand()
command.Connection = connection
command.CommandText = "INSERT INTO Cursos " & _
" (id, nombre) " & "VALUES " & " (" + curso.id + "," + _
curso.nombre + ")"
command.ExecuteNonQuery()
```

3.5 Programación con DataSet

3.5.1 Carga del DataSet

```
Dim dataset As New DataSet("Datos")
Dim cursosCmd As New SqlCommand()
cursosCmd.Connection = connection
cursosCmd.CommandText = "SELECT id, nombre FROM Cursos"
Dim cursosAdapter As New SqlDataAdapter()
Dim cursosBuilder As New
SqlCommandBuilder(cursosAdapter)
cursosAdapter.SelectCommand = cursosCmd
cursosAdapter.Fill(dataset, "Cursos")
```

3.5.2 Lectura de datos del DataSet

Recorrer todas las filas de una tabla:

```
Dim cursosTable As DataTable = dataset.Tables("Cursos")
Dim cursos_lb As New ListBox()
cursos_lb.Items.Clear()
For i As Integer = 0 To cursosTable.Rows.Count - 1
    Dim row As DataRow = cursosTable.Rows(i)
    cursos_lb.Items.Add(row("nombre"))
Next
```

Actividad

Desarrollar una aplicación móvil que cree una base de datos y sus tablas de acuerdo a la actividad vista en la semana 4.

En esta actividad se creará una clase llamada BD la cual tendrá el método para crear la base de datos; también se crearán las clases Table_Producto, Table_Proveedor, Table_Usuario, Table_Producto y Table_Tipo las cuales tendrán los métodos para crear las tablas respectivas.

Clase BD

```
Imports System.Data
Imports System.Xml
Imports System.Data.SqlServerCe
Imports System.IO

Public Class BD

    Public Sub Crear_Base_Datos(ByVal BD_Name As String)
        'Metodo para crear la base de datos

        'Se consulta si existe la BD para crearla
        If Not File.Exists("\My Documents\" & BD_Name & ".sdf") Then

            Try

                Dim Engine As SqlCeEngine
```

```
'Ruta de creacion de la BD en el dispositivo
Engine = New SqlCeEngine( _
    "Data Source = \My Documents\" & BD_Name & ".sdf")

'este es el comando que crea la base de datos

Engine.CreateDatabase()

Catch ex As Exception

    MessageBox.Show("tabla no creada")

End Try

End If

End Sub

Public Function Llamar_Tabla(ByVal tabla As String) As DataSet

    'Metodo para llamar todos los registros
    'de una tabla

    Dim cn As SqlCeConnection

    'Se establece la conexion con la BD
    cn = New SqlCeConnection( _
        "Data Source=\My Documents\Cibertec.sdf")

    cn.Open()

    Dim cmd As SqlCeCommand = cn.CreateCommand

    'Se establece la consulta
    cmd.CommandText = "SELECT * FROM " & tabla

    Dim da As New SqlCeDataAdapter(cmd)

    Dim ds As New DataSet

    da.Fill(ds)

    Try

    Catch ex As Exception

    End Try

    'Se retorna el dataset lleno con los datos
    Return ds

End Function

End Class
```

Clase Table_Producto

```
Imports System.Data
Imports System.Xml
Imports System.Data.SqlServerCe
Imports System.IO

Public Class Table_Producto

    Public Sub Crear_Tabla(ByVal bd_Name As String)

        Dim sql_connection As New SqlCeConnection

        Try

            sql_connection = New SqlCeConnection( _
                "Data Source=\My Documents\" & bd_Name & ".sdf")

            sql_connection.Open()

            Dim sql_Command As SqlCeCommand = _
                sql_connection.CreateCommand

            ' Creación de la tabla de la base de datos

            sql_Command.CommandText = "CREATE TABLE Productos ( " & _
                "Descripcion_Prod nchar(20) NOT NULL PRIMARY KEY, " & _
                "Tipo_Prod nchar(15))"

            sql_Command.ExecuteNonQuery()

        Catch sql_ex As SqlCeException

            'capturamos los errores de creacion si es q hay

            Dim sql_Error As SqlCeError

            For Each sql_Error In sql_ex.Errors

                MessageBox.Show(sql_Error.Message)

            Next

        Catch ex As Exception

            MessageBox.Show("Error " & ex.Message)

        Finally

            If sql_connection.State <> ConnectionState.Closed Then

                sql_connection.Close()

            End If

        End Try

    End Sub

End Class
```

Clase Table_Proveedor

```
Imports System.Data
Imports System.Xml
Imports System.Data.SqlServerCe
Imports System.IO

Public Class Table_Proveedor

    Public Sub Crear_Tabla(ByVal bd_Name As String)

        Dim sql_connection As New SqlCeConnection

        Try

            sql_connection = New SqlCeConnection( _
                "Data Source=\My Documents\" & bd_Name & ".sdf")

            sql_connection.Open()

            Dim sql_Command As SqlCeCommand = _
                sql_connection.CreateCommand

            ' Creación de la tabla de la base de datos

            sql_Command.CommandText = "CREATE TABLE Proveedor ( " & _
                "Descripcion_Prov nchar(20) NOT NULL PRIMARY KEY, " & _
                "Ruc_Prov nchar(11))"

            sql_Command.ExecuteNonQuery()

        Catch sql_ex As SqlCeException

            Dim sql_Error As SqlCeError

            For Each sql_Error In sql_ex.Errors

                MessageBox.Show(sql_Error.Message)

            Next

        Catch ex As Exception

            MessageBox.Show("Error " & ex.Message)

        Finally

            If sql_connection.State <> ConnectionState.Closed Then

                sql_connection.Close()

            End If

        End Try

    End Sub

End Class
```

Clase Table_Tipo

```
Imports System.Data
Imports System.Xml
Imports System.Data.SqlServerCe
Imports System.IO

Public Class Table_Tipo

    Public Sub Crear_Tabla(ByVal bd_Name As String)

        Dim sql_connection As New SqlCeConnection

        Try

            sql_connection = New SqlCeConnection( _
                "Data Source=\My Documents\" & bd_Name & ".sdf")

            sql_connection.Open()

            Dim sql_Command As SqlCeCommand = _
                sql_connection.CreateCommand

            ' Creación de la tabla de la base de datos

            sql_Command.CommandText = "CREATE TABLE Tipo_Producto" & _
                "(Descripcion_Tipo nchar(15) NOT NULL PRIMARY KEY)"

            sql_Command.ExecuteNonQuery()

        Catch sql_ex As SqlCeException

            Dim sql_Error As SqlCeError

            For Each sql_Error In sql_ex.Errors

                MessageBox.Show(sql_Error.Message)

            Next

        Catch ex As Exception

            MessageBox.Show("Error " & ex.Message)

        Finally

            If sql_connection.State <> ConnectionState.Closed Then

                sql_connection.Close()

            End If

        End Try

    End Sub

End Class
```

Clase Table_Usuario

```
Imports System.Xml
Imports System.Data.SqlServerCe
Imports System.IO
Imports System.Data

Public Class Table_Usuario

    Public Sub Crear_Tabla(ByVal bd_Name As String)

        Dim sql_connection As New SqlCeConnection

        Try

            sql_connection = New SqlCeConnection _
                ("Data Source=\My Documents\" & bd_Name & ".sdf")

            sql_connection.Open()

            Dim sql_Command As SqlCeCommand = _
                sql_connection.CreateCommand

            ' Creación de la tabla de la base de datos

            sql_Command.CommandText = "CREATE TABLE Usuarios" & _
                "(ID_Usuario nchar(10) NOT NULL PRIMARY KEY, " & _
                "Nombre_Usuario nchar(10), " & _
                "Password_Usuario nchar(3))"

            sql_Command.ExecuteNonQuery()

        Catch sql_ex As SqlCeException

            Dim sql_Error As SqlCeError

            For Each sql_Error In sql_ex.Errors

                MessageBox.Show(sql_Error.Message)

            Next

        Catch ex As Exception

            MessageBox.Show("Error " & ex.Message)

        Finally

            If sql_connection.State <> ConnectionState.Closed Then

                sql_connection.Close()

            End If

        End Try

    End Sub

End Class
```



```
End Sub
```

Formulario de Ingreso

```
Imports System.Data
Imports System.Data.SqlClient

Public Class Form_Ingreso

    Dim bd_Obj As New BD

    Dim tb_Usuarios As New Table_Usuario

    Dim tb_Tipos As New Table_Tipo

    Private Sub Llenar_Usuarios()

        'Metodo para llenar el ComboBox de Usuarios

        ComboBox_User.Items.Add("jnolasco")

        ComboBox_User.Items.Add("avega")

        ComboBox_User.Items.Add("rvillanueva")

        ComboBox_User.SelectedIndex = 0

    End Sub

    Private Sub Button_Ingresar_Click(ByVal sender As System.Object, _
        ByVal e As System.EventArgs) Handles Button_Ingresar.Click

        'Validacion de Campos Vacíos
        If ComboBox_User.Text = "" Or TextBox_Pass.Text = "" Then

            MessageBox.Show("Debe llenar todos los campos", _
                "Advertencia", MessageBoxButtons.OK, _
                MessageBoxIcon.Exclamation, _
                MessageBoxDefaultButton.Button1)

        Else

            'Verificacion de Password
            If TextBox_Pass.Text = "123" Then

                Dim frm2 As New Form_Principal

                TextBox_Pass.Text = ""

                ComboBox_User.Text = ""

                Form_Principal.Show()

            Else

                MessageBox.Show("Password invalido", "Advertencia", _
                    MessageBoxButtons.OK, _
                    MessageBoxIcon.Exclamation, _
                    MessageBoxDefaultButton.Button1)
```

```
        End If

    End If

End Sub

Private Sub Button_Salir_Click(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles Button_Salir.Click

    If (MessageBox.Show("¿Desea salir?", "Advertencia", _
        MessageBoxButtons.YesNo, MessageBoxIcon.Exclamation, _
        MessageBoxDefaultButton.Button1) = _
        Windows.Forms.DialogResult.Yes) Then

        Application.Exit()

    End If

End Sub

Private Sub Form_Ingreso_Load(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles MyBase.Load

    'Creamos la Base de Datos

    bd_Obj.Crear_Base_Datos("Cibertec")

    'Creamos la tabla Tipos de producto

    'en la base de datos cibertec

    tb_Tipos.Crear_Tabla("Cibertec")

    'Creamos la tabla Usuarios

    'en la base de datos cibertec

    tb_Usuarios.Crear_Tabla("Cibertec")

    Llenar_Usuarios()


End Sub


End Class
```

Autoevaluación

- ¿Cuáles son las diferencias entre un Dataset y un Databinding?

Para recordar

 En una conexión SQL El resultado de la consulta se procesa en memoria y se rellena el formulario

 Si desea saber más acerca de estos temas, puede consultar las siguientes páginas.

 <http://www.mibolita.es/files/articles/CF-ADO.NET.Parte2.pdf>

**UNIDAD DE
APRENDIZAJE****2****SEMANA****11**

ADO.NET en Compact Framework 2

OBJETIVOS ESPECÍFICOS

- Manejar adecuadamente ADO.NET en Compact Framework

CONTENIDOS

- Namespace System.Data.
- Estados de los datos de una tabla.
- Restricciones en una tabla.
- Relaciones entre tablas.

ACTIVIDAD

- Desarrollar una aplicación móvil que maneje la información de una base de datos usando las clases del namespace System.Data.

1. El namespace System.Data

Las clases en este namespace permiten trabajar con datos relacionales en el Compact Framework (cargados de un fichero XML o desde una base de datos SQL Server CE). La clase más importante es DataSet, que sirve para trabajar con datos localmente (en memoria), hacer persistencia y enlazar con controles de interfaz de usuario. Cuando se trabaja con un objeto DataSet se dice que se trabaja en modo desconectado, ya que se utilizan datos en memoria, no en la base de datos original.

Un DataSet se puede construir a partir de una o más fuentes de datos (XML ó SQL Server CE) o de datos creados.

Un objeto DataSet es una base de datos residente en memoria. La estructura de un DataSet es como la de una base de datos relacional: tablas, columnas, filas y restricciones. Una vez rellenado un DataSet, se desconecta de los datos de origen. Un objeto DataSet contiene objetos DataTable, los cuales pueden tener constraints definidos (por ej. primary keys, foreign keys, unique). Se pueden especificar relaciones padre/hijo entre objetos DataTable. Un objeto DataTable contiene una instancia de objeto DataView, la cual ordena y filtra las filas del objeto DataTable.

Los objetos DataSet proporcionan las siguientes ventajas:

- pueden contener datos de una variedad de fuentes de datos (XML, SQL Server CE)
- funcionan desconectados de la fuente de datos
- permiten guardar los datos en formato XML en ficheros locales
- permiten mostrar su contenido en controles de formularios (data binding)

Clases más importantes que componen este namespace:

- Constraint: restricción para un DataColumn
- ConstraintCollection: restricciones de un DataTable
- DataColumn: Columna de un DataTable
- DataColumnCollection: Columnas de un DataTable
- DataRow: Fila de un DataTable
- DataRowCollection: Filas de un DataTable
- DataTable: Tabla en memoria
- DataTableCollection: Colección de tablas de un DataSet
- DataView: Vista de un DataTable
- DataSet: Datos almacenados en memoria (desconectado)

Un objeto DataSet tiene las siguientes propiedades:

- Tables (DataTable)
- Relations (DataRelation)

Un objeto DataTable tiene las siguientes propiedades:

- Columns (DataColumn)
- Rows (DataRow)
- Constraints (Constraint)
- ChildRelations (DataRelation)
- ParentRelations (DataRelation)

1.1 DataSet y XML

Los DataSet proporcionan soporte para XML: los datos del DataSet pueden almacenarse en un fichero XML y/o cargarse desde un fichero XML.

Métodos de DataSet:

- Clear: vacía el DataSet (filas y tablas).
- Clone: crea una copia del DataSet.
- AcceptChanges: Acepta todas las modificaciones que se han hecho al DataSet.
- HasChanges: Indica si se ha modificado un DataSet.
- ReadXml: Carga el DataSet con un fichero XML.
- RejectChanges: Hace rollback de los cambios que se han hecho al DataSet.
- Reset: Resetea el DataSet para volverlo a su estado inicial.
- WriteXml: Salva el DataSet como un fichero XML

Ejemplo de XML:

```
<?xml version="1.0" standalone="yes" ?>
<DatosPersonas>
  <Personas>
    <nombre>Pepe</nombre>
    <dni>245234523J</dni>
  </Personas>
  <Personas>
    <nombre>Juan</nombre>
    <dni>546563456K</dni>
  </Personas>
</DatosPersonas>
```

Trabajando con DataSet

- Crear un DataSet:

Dim dataset As New DataSet("datos")

- Obtener la lista de tablas de un DataSet:

```
For Each table As DataTable In dataset.Tables
....
Next
```

- Obtener una tabla concreta de un DataSet:

```
Dim table As DataTable = dataset.Tables("usuarios")
```

Trabajando con DataTable

- Crear una DataTable:

```
Dim table As New DataTable("personas")
dataset.Tables.Add(table)
```

- Obtener la lista de columnas de un DataTable:

```
For Each col As DataColumn In table.Columns
Next
```

- Acceso a las filas de un DataTable

```
For Each row As DataRow In table.Rows
Next
```

// otra forma

```
For i As Integer = 0 To table.Rows.Count - 1
Dim row As DataRow = table.Rows(i)
....
Next
```

Los estados de la fila pertenecen a la clase `DataRowState`, y pueden ser: `Added`, `ModifiedCurrent`, `Deleted`, `Unchanged`, `None`, etc.).

Trabajando con DataColumn

- Crear un `DataColumn` de un tipo determinado:

```
Dim column As New DataColumn("cantidad")
column.DataType = GetType(Integer)
table.Columns.Add(column)
Dim column2 As New DataColumn("nombre")
column2.DataType = GetType(String)
table.Columns.Add(column2)
```

- Accediendo a una columna:

```
Dim column As DataColumn = table.Columns("nombre")
```

Trabajando con DataRow

La clase `DataRow` representa una fila en un `DataTable`. Se utiliza para añadir, modificar, borrar y consultar filas.

Métodos más comunes de `DataRow`:

- `AcceptChanges`: hace commit de todos los cambios que se han hecho en la fila desde la última vez que se invocó a este método.
- `BeginEdit`: Inicia una edición de una fila

- CancelEdit: Cancela la edición de la fila
- Delete: Elimina la fila
- EndEdit: Completa la edición de la fila
- IsNull: Determina si una columna es nula
- Item: Permite acceder a las columnas de la fila

Añadir una fila:

```
Dim row As DataRow = table.NewRow()  
row("nombre") = "Julian Perez"  
row("DNI") = "4636529"  
table.Rows.Add(row)
```

Modificar una fila:

```
Dim row As DataRow = table.Rows(numFila)  
row("nombre") = "Pedro"
```

Ver el contenido de una fila:

```
Dim row As DataRow = table.Rows(numFila)  
Dim nombre As String = row("nombre")  
Dim dni As String = row("DNI")
```

Hay que tener mucho cuidado con el borrado de filas, ya que éstas no se borran físicamente del DataTable, simplemente se marcan como borradas. Por eso, si inmediatamente después de borrar la DataRow se comprueba la propiedad DataTable.Rows.Count, nos dará el número de filas total (incluyendo la fila borrada).

Hay dos soluciones a este problema:

- Salvar el DataSet inmediatamente después de borrar la(s) fila(s), teniendo la precaución de salvar primero las tablas hijas y luego las tablas padre.
- Crear una DataView con un filtro de filas en el RowState: `DataView.RowStateFilter = CurrentRows`

Obtener las filas hijas de una dada:

```
dataset.Relations.Add( "FKPedidoCliente", _  
tClientes.Columns["idCliente"], _  
tPedidos.Columns["idCliente"], true)  
Dim childs As DataRow() = _  
tClientes.Rows(0).GetChildRows("FKPedidoCliente")
```

Obtener la fila padre de una dada:

```
dataset.Relations.Add( "FKPedidoCliente", _  
tClientes.Columns["idCliente"], _  
tPedidos.Columns["idCliente"], true)  
Dim parent As DataRow = _  
tPedidos.Rows(0).GetParentRow("FKPedidoCliente")
```

2. Estados de los datos de una tabla

Los datos de una tabla pueden considerarse tridimensionales, o sea, para obtener un dato hay que especificar la fila, la columna y el estado. El estado puede ser Current (valor actual), Original (valor inicial), Proposed (utilizado para commit y rollback), Default (valor por defecto de una columna). Además, cada

una de las filas de una tabla puede estar en uno de los siguientes estados: Unchanged, Modified, Added, Deleted.

Cuando se crea una nueva fila en una tabla, los valores Original y Current son iguales, y el estado de la fila es Unchanged. Cuando se modifica un valor de la fila, el nuevo valor se puede obtener en Current, el anterior permanece en Original, y el estado pasa a Modified. Cuando se llama al método AcceptChanges() de DataRow, el valor Original es reemplazado con el de Current, y el estado pasa a Unchanged. También existe el método AcceptChanges() en DataSet y DataTable.

3. Restricciones en una tabla

3.1 Foreign Key Constraints

Una FK especifica la acción que se llevará a cabo en relaciones padre/hijo. Cuando se modifica o borra la fila padre, la FK determina cómo reaccionará la fila hija. Por ejemplo, si la fila padre se borra, se podría especificar que todas las filas hijas se borren, o bien, que todas las filas hijas pusieran su referencia a la fila padre a NULL o al valor por defecto.

La clase ForeignKeyConstraint tiene las siguientes propiedades:

- DeleteRule.
- UpdateRule.
- Y estas propiedades pueden tomar los siguientes valores:
- Rule.Cascade: borra o actualiza las filas hijas (acción por defecto).
- Rule.SetNull: pone los valores de las filas hijas a NULL.
- Rule.SetDefault: pone los valores de las filas hijas al valor por defecto.
- Rule.None: no hace nada.

3.2 Unique Constraints

Una UniqueKeyConstraint obliga a que los valores de una columna sean distintos. Se puede poner una UniqueKeyConstraint sobre varias columnas.

3.3 Primary Keys

Para definir la clave primaria de una tabla:

```
Dim pk(0 To 2 - 1) As DataColumn  
pk(0) = col1  
pk(1) = col2  
table.PrimaryKey = pk
```

Se puede hacer que una columna que vaya a actuar como clave primaria tenga las siguientes propiedades:

```
column.AutoIncrement = True  
column.ReadOnly = True  
column.AutoIncrementSeed = 1
```

4. Relaciones entre tablas

Un DataSet no tiene la capacidad que tienen las bases de datos para mantener relaciones padre/hijo entre tablas. Tenemos que crear un objeto DataRelation para mantener estas relaciones. El objeto DataSet tiene la propiedad Relations que mantiene todas las relaciones padre/hijo del DataSet. Por ejemplo, las tablas Clientes y Pedidos están relacionadas (cada registro de la tabla Pedidos está relacionado con un registro de la tabla Clientes). Esta relación puede ser a través del campo IdCliente. Pero incluso aunque ambas tablas tengan este campo común, el DataSet no mantiene la relación entre los registros de ambas

tablas. Para esto, tenemos que crear un objeto `DataRelation` que referencia a las tablas padre e hija (y sus correspondientes claves).

El objeto `DataRelation` proporciona las siguientes funcionalidades:

- proporciona el registro padre de uno dado
- proporciona los registros hijos de uno dado
- integridad referencial (elimina los registros hijos huérfanos)

Para añadir relaciones a un `DataSet` se puede hacer de una de las siguientes formas:

- `dataset.Relations.Add(string relationName, DataColumn parentColumn, DataColumn childColumn)`
- `dataset.Relations.Add(string relationName, DataColumn() parentColumn, DataColumn() childColumn)`

Cuando se invoca el método `Add()` se crea un nuevo objeto `DataRelation`, se añade un `UniqueConstraint` a la tabla padre, y se añade un `ForeignKeyConstraint` a la tabla hija.

Actividad 1

Desarrollar una aplicación móvil que maneje la información de una base de datos usando las clases del namespace System.Data tomando como base la actividad vista en la semana 10

Para esto utilizaremos las mismas clases de la actividad de la semana 10 a las cuales añadiremos lo siguiente en cada clase:

Clase Table_Producto

```
Public Function Llamar_Productos(ByVal tabla As _
String) As DataSet

    'Metodo para llamar a todos los productos

    Dim cn As SqlConnection

    cn = New SqlConnection( _
        "Data Source=\My Documents\Cibertec.sdf")

    cn.Open()

    Dim cmd As SqlCommand = cn.CreateCommand

    cmd.CommandText = "SELECT * FROM " & tabla

    Dim da As New SqlDataAdapter(cmd)

    Dim ds As New DataSet

    da.Fill(ds)

    Try

    Catch ex As Exception

    End Try

    Return ds

End Function
```

```
Public Function Llamar_Productos_Por_Tipo _
    (ByVal tipo As String) As DataSet

    'Metodo para llamar a los productos de acuerdo al tipo

    Dim cn As SqlConnection

    cn = New SqlConnection( _
        "Data Source=\My Documents\Cibertec.sdf")

    cn.Open()

    Dim cmd As SqlCommand = cn.CreateCommand

    cmd.CommandText = _
        "SELECT * FROM Productos where tipo_prod='" & tipo & "'"

    Dim da As New SqlDataAdapter(cmd)

    Dim ds As New DataSet

    da.Fill(ds)

    Try

    Catch ex As Exception

    End Try

    Return ds

End Function

Public Sub Registrar_Producto(ByVal descrip _
    As String, ByVal tipo As String)

    'Metodo para registrar un producto

    Dim sql_connection As New SqlConnection

    'Se establece la conexion

    sql_connection = New SqlConnection( _
        "Data Source=\My Documents\Cibertec.sdf")

    sql_connection.Open()

    Dim sql_Command As SqlCommand = _
        sql_connection.CreateCommand

    ' Agregamos registro

    sql_Command.CommandText = "INSERT INTO Productos" & _
        "(Descripcion_Prod, Tipo_Prod)" & _
        " VALUES (?, ?)"

    'Se mandan los parametros para el comando

    sql_Command.Parameters.Add(New SqlParameter( _
```

```

"@Descripcion_Prod", descrip))

sql_Command.Parameters.Add(New SqlCeParameter( _
"@Tipo_Prod", tipo))

sql_Command.Prepare()

'Se ejecuta el comando insert en la BD

sql_Command.ExecuteNonQuery()

End Sub

```

Clase Table_Proveedor

```

Public Sub Registrar_Proveedor _
(ByVal descrip As String, ByVal ruc As String)

    Dim sql_connection As New SqlCeConnection

    sql_connection = New SqlCeConnection _
("Data Source=\My Documents\Cibertec.sdf")

    sql_connection.Open()

    Dim sql_Command As SqlCeCommand = _
sql_connection.CreateCommand

    ' Agregamos registro

    sql_Command.CommandText = "INSERT INTO Proveedor" & _
"(Descripcion_Prov, Ruc_Prov)" & _
" VALUES (?, ?)"

    'Se mandan los parametros para el comando

    sql_Command.Parameters.Add(New SqlCeParameter _
("@Descripcion_Prov", descrip))

    sql_Command.Parameters.Add(New SqlCeParameter _
("@Ruc_Prov", ruc))

    sql_Command.Prepare()

    sql_Command.ExecuteNonQuery()

End Sub

```

```

Public Function Llamar_Proveedores() As DataSet

    'Metodo para llamar a todos los proveedores

    Dim cn As SqlCeConnection

    cn = New SqlCeConnection( _
"Data Source=\My Documents\Cibertec.sdf")

```

```
cn.Open()

Dim cmd As SqlCeCommand = cn.CreateCommand

cmd.CommandText = "SELECT * FROM Proveedor"

Dim da As New SqlCeDataAdapter(cmd)

Dim ds As New DataSet

da.Fill(ds)

Try

Catch ex As Exception

End Try

Return ds

End Function
```

Clase Table_Tipo

```
Public Sub Registrar_Tipo(ByVal descrip As String)

    Dim sql_connection As New SqlCeConnection

    sql_connection = New SqlCeConnection( _
        "Data Source=\My Documents\Cibertec.sdf")

    sql_connection.Open()

    Dim sql_Command As SqlCeCommand = _
        sql_connection.CreateCommand

    ' Agregamos registro

    sql_Command.CommandText = "INSERT INTO Tipo_Producto" & _
        "(Descripcion_Tipo)" & _
        " VALUES (?)"

    'Se mandan los parametros para el comando

    sql_Command.Parameters.Add(New SqlCeParameter( _
        "@Descripcion_Tipo", descrip))

    sql_Command.Prepare()

    sql_Command.ExecuteNonQuery()

End Sub
```

Clase Table_Usuario

```
Public Function Validar_Usuario(ByVal _
usuario As String) As String

    'Metodo para validar usuarios

    Dim pass As String

    Dim cn As SqlConnection

    cn = New SqlConnection _
        ("Data Source=\My Documents\Cibertec.sdf")

    cn.Open()

    Dim cmd As SqlCommand = cn.CreateCommand

    cmd.CommandText = _
        "SELECT Password_Usuario FROM Usuarios where ID_Usuario='" & _
        usuario & "'"

    Dim da As New SqlDataAdapter(cmd)

    Dim ds As New DataSet

    da.Fill(ds)

    pass = ds.Tables(0).Rows(0).Item(0)

    Try

    Catch ex As Exception

    End Try

    Return pass

End Function

Public Sub Crear_Usuarios(ByVal id _
As String, ByVal nombre As String, ByVal pass As String)

    'Metodo para crear un usuario

    Dim sql_connection As New SqlConnection

    sql_connection = New SqlConnection( _
        "Data Source=\My Documents\Cibertec.sdf")

    sql_connection.Open()

    Dim sql_Command As SqlCommand = _
        sql_connection.CreateCommand

    ' Agregamos registro

    sql_Command.CommandText = "INSERT INTO Usuarios" & _
```

```

"(ID_Usuario, Nombre_Usuario, Password_Usuario)" & _
" VALUES (?, ?, ?)"

'Se mandan los parametros para el comando insert
sql_Command.Parameters.Add(New SqlCeParameter _
("@ID_Usuario", id))

sql_Command.Parameters.Add(New SqlCeParameter _
("@Nombre_Usuario", nombre))

sql_Command.Parameters.Add(New SqlCeParameter _
("@Password_Usuario", pass))

sql_Command.Prepare()

sql_Command.ExecuteNonQuery()

End Sub

Public Function Lllamar_Usuarios() As DataTable

'Metodo para llamar a todos los usuarios

Dim cn As SqlCeConnection

cn = New SqlCeConnection _
("Data Source=\My Documents\Cibertec.sdf")

Dim dt_usuarios As New DataTable

Dim cmd As SqlCeCommand = cn.CreateCommand

Dim dr_usuarios As SqlCeDataReader

Dim x As Integer

cn.Open()

cmd.CommandText = "SELECT * FROM Usuarios"

dr_usuarios = cmd.ExecuteReader()

'Se crean las columnas del datatable

'de acuerdo a los datos en el datareader

For x = 0 To (dr_usuarios.FieldCount - 1)

    Dim column As New DataColumn(dr_usuarios.GetName(x))

    column.DataType = GetType(String)

    dt_usuarios.Columns.Add(column)

Next

x = 0

'Se llena las filas que llenaran nuestro

'datatable con los datos del datareader

```

```

While dr_usuarios.Read

    Dim dr As DataRow = dt_usuarios.NewRow()

    dr("ID_Usuario") = dr_usuarios(0).ToString
    dr("Nombre_Usuario") = dr_usuarios(1).ToString
    dr("Password_Usuario") = dr_usuarios(2).ToString

    dt_usuarios.Rows.Add(dr)

End While

Try

Catch ex As Exception

End Try

Return dt_usuarios

End Function

```

Los formularios quedarían del siguiente modo:

Formulario de Ingreso

```

Imports System.Data
Imports System.Data.SqlClient

Public Class Form_Ingreso

    Dim bd_Obj As New BD

    Dim tb_Usuarios As New Table_Usuario

    Dim tb_Tipos As New Table_Tipo

    Private Sub Llenar_Usuarios()

        'Metodo para llenar el ComboBox de Usuarios

        ComboBox_User.DataSource = tb_Usuarios.Llamar_Usuarios()

        ComboBox_User.DisplayMember = tb_Usuarios. _
        Llamar_Usuarios().Columns(0).ToString

        ComboBox_User.SelectedIndex = 0

    End Sub

    Private Sub Button_Ingresar_Click(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles Button_Ingresar.Click

```

```

'Validacion de Campos Vacíos
If ComboBox_User.Text = "" Or TextBox_Pass.Text = "" Then

    MessageBox.Show("Debe llenar todos los campos", _
        "Advertencia", MessageBoxButtons.OK, _
        MessageBoxIcon.Exclamation, _
        MessageBoxDefaultButton.Button1)

Else
    'Verificacion de Password
    If TextBox_Pass.Text = tb_Usuarios.Validar_Usuario _
        (ComboBox_User.Text.Trim) Then

        Dim frm2 As New Form_Principal

        TextBox_Pass.Text = ""

        ComboBox_User.SelectedIndex = 0

        Form_Principal.Show()

    Else

        MessageBox.Show("Password invalido", "Advertencia", _
            MessageBoxButtons.OK, _
            MessageBoxIcon.Exclamation, _
            MessageBoxDefaultButton.Button1)

    End If

End If

End Sub

Private Sub Button_Salir_Click(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles Button_Salir.Click

    If (MessageBox.Show("¿Desea salir?", "Advertencia", _
        MessageBoxButtons.YesNo, MessageBoxIcon.Exclamation, _
        MessageBoxDefaultButton.Button1) = _
        Windows.Forms.DialogResult.Yes) Then

        Application.Exit()

    End If

End Sub

Private Sub Form_Ingreso_Load(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles MyBase.Load

    'Creamos la Base de Datos

    bd_Obj.Crear_Base_Datos("Cibertec")

    'Creamos la tabla Tipos de producto

    'en la base de datos cibertec

    tb_Tipos.Crear_Tabla("Cibertec")

```



```
'Se llena la tabla tipo de productos
tb_Tipos.Registrar_Tipo("Entrada")
tb_Tipos.Registrar_Tipo("Plato de Fondo")
tb_Tipos.Registrar_Tipo("Postre")
tb_Tipos.Registrar_Tipo("Bebida")

'Creamos la tabla Usuarios
'en la base de datos cibertec
tb_Usuarios.Crear_Tabla("Cibertec")

'Se llena la tabla usuarios
tb_Usuarios.Crear_Usuarios("jnolasco", "Johan", "123")
tb_Usuarios.Crear_Usuarios("avega", "Alejandro", "123")
tb_Usuarios.Crear_Usuarios("rvilla", "Roland", "123")

Llenar_Usuarios()

End Sub

End Class
```

Formulario Principal

```
Public Class Form_Principal

    Private Sub Button_Productos_Click(ByVal sender As System.Object _
    , ByVal e As System.EventArgs) Handles Button_Producto.Click

        'Llama al formulario productos
        Form_Productos.Show()

    End Sub

    Private Sub Button_proveedor_Click(ByVal sender As System.Object _
    , ByVal e As System.EventArgs) Handles Button_Proveedor.Click

        'Llama al formulario proveedores
        Form_Proveedores.Show()

    End Sub

    Private Sub Button_Salir_Click(ByVal sender As System.Object _
    , ByVal e As System.EventArgs) Handles Button_Salir.Click

        If (MessageBox.Show("¿Deseas salir", "Advertencia", _
        MessageBoxButtons.YesNo, MessageBoxIcon.Exclamation, _
```

```
        MessageBoxDefaultButton.Button1) = _  
        Windows.Forms.DialogResult.Yes) Then  
  
            Me.Close()  
  
        End If  
  
    End Sub  
  
End Class
```

Formulario Productos

```
Imports System.Data  
  
Public Class Form_Productos  
  
    'Variable nodo de tipo TreeNode  
  
    Dim nodo As New TreeNode  
  
    Dim bd_Obj As New BD  
  
    Dim tb_Producto As Table_Producto  
  
  
    Private Sub Form_Productos_Load(ByVal sender As System.Object _  
    , ByVal e As System.EventArgs) Handles MyBase.Load  
  
        tb_Producto = New Table_Producto  
  
        tb_Producto.Crear_Tabla("Cibertec")  
  
        Llenar_Tipos()  
  
    End Sub  
  
    Private Sub Button_Cancelar_Click(ByVal sender As System.Object _  
    , ByVal e As System.EventArgs) Handles Button_Cancelar.Click  
  
        TextBox_Descrip.Text = ""  
  
        ComboBox_Tipo.SelectedIndex = 0  
  
    End Sub  
  
    Private Sub Button_Registrar_Click(ByVal sender As System.Object _  
    , ByVal e As System.EventArgs) Handles Button_Registrar.Click  
  
        tb_producto = New Table_Producto  
  
        If TextBox_Descrip.Text = "" Then  
  
            MessageBox.Show("Debe llenar todos los campos", "Aviso", _  
            MessageBoxButtons.OK, MessageBoxIcon.Exclamation, _  
            MessageBoxDefaultButton.Button1)
```

```
Else

    'De acuerdo al indice seleccionado del combobox

    'se manda la variable al metodo Llenar_Nodos()

    tb_producto.Registrar_Producto _
    (TextBox_Descrip.Text, ComboBox_Tipo.Text)

    Llenar_Nodos(ComboBox_Tipo.SelectedIndex)

End If

End Sub

Private Sub Button_Regresar_Click(ByVal sender As System.Object _
, ByVal e As System.EventArgs) Handles Button_Regresar.Click

    If (MessageBox.Show("¿Desea regresar?", "Advertencia", _
    MessageBoxButtons.YesNo, MessageBoxIcon.Exclamation, _
    MessageBoxDefaultButton.Button1) = _
    Windows.Forms.DialogResult.Yes) Then

        Me.Hide()

    End If

End Sub

Private Sub Llenar_Nodos(ByVal x As Integer)

    'Metodo para llenar los subnodos del TreeView de Productos

    'donde x sera el numero de nodo del TreeView

    Dim y As Integer

    tb_Producto = New Table_Producto

    'Se iguala la variable TreeNode al Nodo principal del TreeView
    'establecido por la variable x

    nodo = TreeView_Prod.Nodes(x)

    nodo.Nodes.Clear()

    'Se cierran los nodos del TreeView
    TreeView_Prod.CollapseAll()

    'Se aumentan subnodos al nodo principal

    For y = 0 To tb_Producto.Llamar_Productos_Por_Tipo _
    (ComboBox_Tipo.Text).Tables(0).Rows.Count - 1

        nodo.Nodes.Add(tb_Producto.Llamar_Productos_Por_Tipo _
        (ComboBox_Tipo.Text).Tables(0).Rows(y).Item(0).ToString)

    Next

    MessageBox.Show("Producto registrado", "Aviso", _
```

```

        MessageBoxButtons.OK, MessageBoxIcon.Exclamation, _
        MessageBoxDefaultButton.Button1)

    Tab_Producto.SelectedIndex = 1

    'Se expande el nodo principal seleccionado del TreeView
    nodo.Expand()

    TextBox_Descrip.Text = ""

    ComboBox_Tipo.SelectedIndex = 0

End Sub

Private Sub Llenar_Tipos()

    'Metodo para llenar el ComboBox de Tipos de productos

    ComboBox_Tipo.DataSource = bd_Obj.Llamar_Tabla _
    ("Tipo_Producto").Tables(0)

    ComboBox_Tipo.DisplayMember = bd_Obj.Llamar_Tabla _
    ("Tipo_Producto").Tables(0).Columns(0).ToString

    ComboBox_Tipo.SelectedIndex = 0

End Sub

End Class

```

Formulario Proveedores

```

Public Class Form_Proveedores

    Dim tb_Proveedores As Table_Proveedor

    Private Sub Button_Registrar_Click(ByVal sender As System.Object _
    , ByVal e As System.EventArgs) Handles Button_Registrar.Click

        Dim x As Integer

        tb_Proveedores = New Table_Proveedor

        'Se validan los campos vacios
        If TextBox_Descrip.Text = "" Or TextBox_Ruc.Text = "" Then

            MessageBox.Show("Debe llenar todos los campos", "Aviso", _
            MessageBoxButtons.OK, MessageBoxIcon.Exclamation, _
            MessageBoxDefaultButton.Button1)

        Else

            tb_Proveedores.Registrar_Proveedor _
            (TextBox_Descrip.Text, TextBox_Ruc.Text)

            'Variable de tipo ListViewItem

            Dim ListView_Item As ListViewItem

```

```

        'Limpiamos los subitems

Listview_Prov.Items.Clear()

For x = 0 To tb_Proveedores.Llamar_Proveedores. _
Tables(0).Rows.Count - 1

    'Se crea el Item Principal para el ListView

    ListView_Item = New ListViewItem _
    (tb_Proveedores.Llamar_Proveedores.Tables(0). _
    Rows(x).Item(0).ToString.Trim)

    'Se añade el subitem al Item Principal del ListView

    ListView_Item.SubItems.Add(tb_Proveedores. _
    Llamar_Proveedores.Tables(0).Rows(x).Item _
    (1).ToString.Trim)

    'Se añade el Item principal al ListView

    Listview_Prov.Items.Add(ListView_Item)

Next

MessageBox.Show("Proveedor registrado", "Aviso", _
MessageBoxButtons.OK, MessageBoxIcon.Exclamation, _
MessageBoxDefaultButton.Button1)

Tab_Prov.SelectedIndex = 1

Borrar_Campos()

End If

End Sub

Private Sub Button_Cancelar_Click(ByVal sender As System.Object _
, ByVal e As System.EventArgs) Handles Button_Cancelar.Click

    Borrar_Campos()

End Sub

Private Sub Borrar_Campos()

    TextBox_Descrip.Text = " "

    TextBox_Ruc.Text = " "

End Sub

Private Sub Button_Regresar_Click(ByVal sender As System.Object, _
ByVal e As System.EventArgs) Handles Button_Regresar.Click

    If (MessageBox.Show("¿Deseas regresar?", "Advertencia", _
    MessageBoxButtons.YesNo, MessageBoxIcon.Exclamation, _
    MessageBoxDefaultButton.Button1) = _
    Windows.Forms.DialogResult.Yes) Then

        Me.Hide()
    
```

```
End If

End Sub

Private Sub Form_Proveedores_Load(ByVal sender As Object, _
ByVal e As System.EventArgs) Handles MyBase.Load

    tb_Proveedores = New Table_Proveedor

    'Se crea la tabla proveedores

    tb_Proveedores.Crear_Tabla("Cibertec")

End Sub

End Class
```

Actividad 2

Desarrollar una aplicación móvil web que maneje la información de una base de datos usando las clases del namespace System.Data

Base de datos

```
CREATE DATABASE BDCIBERTEC
GO
USE BDCIBERTEC
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[Pedido]') AND type in (N'U'))
BEGIN
CREATE TABLE [dbo].[Pedido](
    [Id_Pedido] [int] IDENTITY(1,1) NOT NULL,
    [Id_Producto] [int] NULL,
    [Cantidad] [int] NULL,
    CONSTRAINT [PK_Pedido] PRIMARY KEY CLUSTERED
(
    [Id_Pedido] ASC
)WITH (IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
END
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
```

```
IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[Producto]') AND type in (N'U'))
BEGIN
CREATE TABLE [dbo].[Producto](
    [ID_Producto] [int] NOT NULL,
    [NombreProd] [varchar](50) NULL,
    [CantidadProd] [int] NULL,
    CONSTRAINT [PK_Producto] PRIMARY KEY CLUSTERED
(
    [ID_Producto] ASC
)WITH (IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
END
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[realizar_pedido]') AND type in (N'P',
N'PC'))
BEGIN
EXEC dbo.sp_executesql @statement = N'create proc
[dbo].[realizar_pedido]
(
    @id int,
    @cant int
)
as
begin
insert into pedido values (@id,@cant)
update producto set cantidadprod=(cantidadprod-@cant) where
id_producto=@id
end'
END
```


Aplicación Web

```

<%@ Page Inherits= "System.Web.UI.MobileControls.MobilePage"
    Language="VB" %>

<%@ Register TagPrefix="mobile"
    Namespace="System.Web.UI.MobileControls"
    Assembly="System.Web.Mobile" %>

<script language=vb runat="server">

    Private xitem As String
    Private xcantidad As Integer
    Private xcodigo As String
    Private cn = New Data.SqlClient.SqlConnection( _
        "data source=.;uid=sa;pwd=abcDEF123;initial" & _
        "catalog=BDCibertec")
    Dim da As Data.SqlClient.SqlDataAdapter
    Dim cmd As Data.SqlClient.SqlCommand
    Dim ds As Data.DataSet

    'Se crea la clase Productos con sus
    'respectivas propiedades

    Public Class Productos

        Private _codigo, _item, _cantidad As String

        Public Sub New(ByVal Codigo As String, _
            ByVal item As String, _
            ByVal cantidad As String)

            _codigo = Codigo

            _item = item

            _cantidad = cantidad

        End Sub

        Public ReadOnly Property Codigo() As String

            Get

                Return _codigo

            End Get

        End Property

        Public ReadOnly Property Producto() As String

            Get

                Return _item

            End Get

        End Property
    End Class

```

```
End Property

Public ReadOnly Property cantidad() As String

    Get

        Return _cantidad

    End Get

End Property

End Class

'Al cargar la pagina inicial se llama al metodo
'Cargar_Datos()

Public Sub Page_Load(ByVal o As Object, _
ByVal e As EventArgs)

    If Not IsPostBack Then

        Cargar_Datos()

    End If

End Sub

'Metodo para conectarse a la BD y llenar el ObjectList
Private Sub Cargar_Datos()

    cmd = New Data.SqlClient.SqlCommand( _
        "select Id_Producto as ID," _
        & "NombreProd as Nombre, CantidadProd" _
        & " as Cantidad from producto", cn)

    da = New Data.SqlClient.SqlDataAdapter(cmd)

    ds = New Data.DataSet

    da.Fill(ds)

    cn.Close()

    Dim x As Integer

    Dim arr As New ArrayList()

    For x = 0 To ds.Tables(0).Rows.Count - 1

        arr.Add(New Productos(ds.Tables(0). _
            Rows(x).Item(0).ToString, _
            ds.Tables(0).Rows(x).Item(1).ToString, _
            ds.Tables(0).Rows(x).Item(2).ToString))

    Next

    List1.DataSource = arr
```

```
List1.DataBind()

ActiveForm = Frm_Almacen

End Sub

'Se utilizan los valores de las propiedades
'del producto seleccionado para el Frm_Pedido

Public Sub List1_Click(ByVal sender As Object, _
    ByVal e As ObjectListCommandEventArgs)

    If e.CommandName = "Reserve" Then

        xitem = e.ListItem("producto").ToString
        xcantidad = e.ListItem("cantidad").ToString

        message.Text = xitem & " disponibles " _
            & xcantidad

        txt_cantidad.Text = xcantidad

        txt_cantidad.Visible = False

        txt_codigo.Text = e.ListItem("codigo").ToString

        txt_codigo.Visible = False

        ActiveForm = Frm_Pedido

    End If

End Sub

'Metodo para realizar el proceso de Pedido
Private Sub R_Pedido(ByVal id As Integer, _
    ByVal cant As Integer)

    cn.open()

    cmd = New Data.SqlClient.SqlCommand( _
        "realizar_pedido", cn)

    cmd.CommandType = Data.CommandType. _
        StoredProcedure

    cmd.Parameters.Add("@id", _
        Data.SqlDbType.Int).Value = id

    cmd.Parameters.Add("@cant", _
        Data.SqlDbType.Int).Value = cant

    cmd.ExecuteNonQuery()

    cn.close()

End Sub
```

```
'Validacion de la cantidad requerida para realizar el pedido
Sub Btn_pedido(ByVal Sender As Object, _
ByVal E As EventArgs)
```

```
    If Page.IsValid Then
```

```
        If txt_cant_ped.Text = "" Then
```

```
            Exit Sub
```

```
        End If
```

```
        R_Pedido(Val(txt_codigo.Text), _
Val(txt_cant_ped.Text))
```

```
        ActiveForm = Frm_Confirmacion
```

```
    End If
```

```
End Sub
```

```
Sub Btn_Menu_Principal(ByVal Sender As Object, _
ByVal E As EventArgs)
```

```
    Cargar_Datos()
```

```
End Sub
```

```
</script>
```

```
'Formulario Almacen
```

```
<Mobile:Form id="Frm_Almacen" runat="server"
    BackColor="LightBlue">
```

```
<Mobile:Label runat="server" text=" Almacen Central"
    id="label" Font-Bold="true" Font-Size="large"/>
```

```
<mobile:ObjectList id="List1" runat="server"
    LabelField="Producto" OnItemCommand="List1_Click">
```

```
    <Command Name="Reserve" Text="Pedido"/>
```

```
</mobile:ObjectList>
```

```
</mobile:Form>
```

```
'Formulario Pedido
```

```
<Mobile:Form id="Frm_Pedido" runat="server"
    BackColor="LightBlue">
```

```
<Mobile:Label runat="server" text=" Realizar Pedido"
    id="lbl_pedido" Font-Bold="true"
    Font-Size="large"/>
```

```

<Mobile:CompareValidator runat="server"
    ControlToValidate="txt_cantidad"
    ControlToCompare="txt_cant_ped"
    Type="Integer"
    Operator="GreaterThanEqual">
La cantidad requerida es menor a la cantidad disponible

</Mobile:CompareValidator>

<Mobile:Label runat="server" id="message" />

<Mobile:TextBox runat="server" id="txt_cantidad" />

<Mobile:TextBox runat="server" id="txt_codigo" />

<Mobile:Label runat="server" id="lbl_message">
    ¿Cuanto desea llevar?</Mobile:Label>

<Mobile:TextBox runat="server" id="txt_cant_ped" />

<Mobile:Command runat="server"  OnClick="Btn_pedido"
    Text="Submit" />

    <mobile:Link id="Return" NavigateURL="#Frm_Almacen"
        runat="server" text="Return" />

</mobile:Form>

'Formulario Confirmacion
<Mobile:Form id="Frm_Confirmacion" runat="server"
    BackColor="LightBlue">

<Mobile:Label runat="server">El Pedido se realizo con exito
</Mobile:Label>

<Mobile:Command runat="server" Name="Menu"
    OnClick="Btn_Menu_Principal" Text="Menu Principal"/>


</mobile:Form>


```

Autoevaluación

- ¿Qué ventajas nos ofrece un objeto dataset?
- ¿Qué especifica un Foreign Key Constraints?

Para recordar

 Un objeto DataSet es una base de datos residente en memoria. La estructura de un DataSet es como la de una base de datos relacional: tablas, columnas, filas y restricciones. Una vez rellenado un DataSet, se desconecta de los datos de origen.

 Si desea saber más acerca de estos temas, puede consultar las siguientes páginas.

 <http://www.mibolita.es/files/articles/CF-ADO.NET.Parte1.pdf>

 http://www.w3schools.com/dotnetmobile/mobile_reference.asp

**UNIDAD DE
APRENDIZAJE**

2

SEMANA

12

Remote data Access

OBJETIVOS ESPECÍFICOS

- Conocer el funcionamiento de un acceso remoto a datos

CONTENIDOS

- Remote Data Access
- Operación Pull
- Operación Push

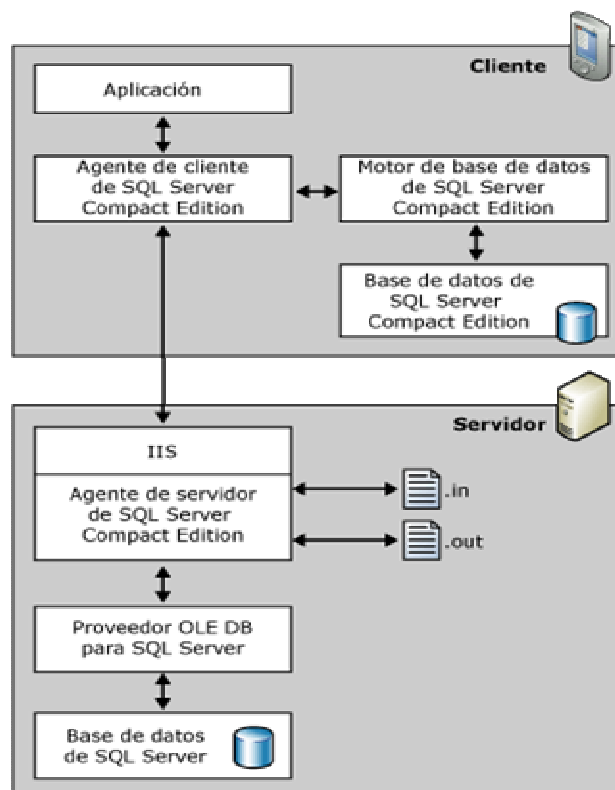
ACTIVIDAD

- Desarrollar una aplicación móvil que realice lo procesos pull y push

1. Remote Data Access

RDA permite mantener la sincronización entre una base de datos en un dispositivo móvil y una base de datos remota, sin necesitar una conexión constante (este tipo de conexiones se denominan **Loosely coupled connection**). Una vez que se han recuperado los datos del servidor remoto, éstos son almacenados y tratados en el dispositivo móvil mediante el engine de **SqlCe**. Los datos almacenados, así como sus cambios e inserciones, pueden ser llevados de nuevo al servidor remoto.

El RDA utiliza los Servicios de Microsoft Internet Information Server (IIS) como mecanismo de comunicación entre la base de datos de SQL Server en el servidor y la base de datos de SQL Server Compact Edition en el dispositivo. El Database Engine (Motor de base de datos) de SQL Server Compact Edition, el Agente de cliente de SQL Server Compact Edition y el Agente de servidor de SQL Server Compact Edition trabajan juntos para lograr el acceso a datos remotos (RDA), como se muestra en el siguiente diagrama.



1.1 Motor de base de datos de SQL Server Compact Edition

El Database Engine (Motor de base de datos) de SQL Server Compact Edition administra el almacén de datos de SQL Server en el dispositivo. Para las operaciones de extracción en las que se efectúa un seguimiento, Database Engine (Motor de base de datos) realiza el seguimiento de todos los registros de base de datos que se hayan insertado, actualizados o eliminados manteniendo una pequeña cantidad de datos de seguimiento de cambios con cada registro. Si existen índices en los datos de SQL Server, el RDA también permitirá la creación de índices en los datos locales.

1.2 Agente de cliente de SQL Server Compact Edition

El Agente de cliente de SQL Server Compact Edition, ubicado en el dispositivo, es el principal componente utilizado para RDA. El Agente de cliente implementa la interfaz de objetos de RDA. Las aplicaciones llaman a esta interfaz para controlar el RDA mediante programación.

Las acciones que el Agente de cliente de SQL Server Compact Edition lleva a cabo en respuesta a las llamadas de métodos de objeto de RDA se describen en la siguiente tabla.

Método	Acción
Pull	Reenvía la solicitud al Agente de servidor de SQL Server Compact Edition a través de HTTP. Cuando el Agente de cliente de SQL Server Compact Edition recibe el conjunto de registros de SQL Server, lo almacena en la base de datos de SQL Server Compact Edition.
Push	Extrae todos los registros insertados, actualizados y eliminados de la base de datos de SQL Server Compact Edition y los envía al Agente de servidor de SQL Server Compact Edition a través de HTTP.
SubmitSQL	Reenvía la solicitud SQL especificada al Agente de servidor de SQL Server Compact Edition a través de HTTP.

1.3 Agente de servidor de SQL Server Compact Edition

El Agente de servidor de SQL Server Compact Edition, ubicado en el equipo que ejecuta ISS, controla las solicitudes de HTTP realizadas por el Agente de cliente de SQL Server Compact Edition. Utiliza archivos de mensajes temporales (*.in y *.out) para administrar el intercambio de datos entre SQL Server y SQL Server Compact Edition.

Las acciones que el Agente de servidor de SQL Server Compact Edition lleva a cabo en respuesta a las llamadas de métodos de objeto de RDA se describen en la siguiente tabla.

Método	Acción
Pull	Recibe la solicitud del Agente de cliente de SQL Server Compact Edition, se conecta a SQL Server a través del proveedor OLE DB para SQL Server y solicita la instrucción SQL del cliente. El Agente de servidor de SQL Server Compact Edition devuelve el conjunto de registros resultante al Agente de cliente de SQL Server Compact Edition a través de HTTP.
Push	Recibe todos los registros insertados, actualizados y eliminados del Agente de cliente de SQL Server Compact Edition, se conecta a SQL Server a través de OLE DB, e inserta, actualiza o elimina los registros de la base de datos de SQL Server. Si se producen errores, el Agente de servidor de SQL Server Compact Edition informa de los errores al Agente de cliente de SQL Server Compact Edition a través de HTTP.
SubmitSQL	Recibe la solicitud de SQL especificada del Agente de cliente de SQL Server Compact Edition a través de HTTP, se conecta a SQL Server a través de OLE DB y solicita la instrucción SQL del cliente. Si se producen errores, el Agente de servidor de SQL Server Compact Edition informa de los errores al Agente de cliente de SQL Server Compact Edition a través de HTTP.

2. Operación Pull

El proceso de Pull recupera los datos de la base de datos remota y los almacena en la base de datos del dispositivo móvil. Esta operación crea el esquema de la tabla y añade los datos demandados en la operación. Es importante destacar que para que el proceso de Pull se haga correctamente la base de datos en el dispositivo móvil NO debe poseer una tabla o subconjunto de datos como el que se ha demandado. La concurrencia en las operaciones de Pull es tratada en el servidor remoto mediante la implementación de **Concurrencia Optimista**. Las operaciones de Pull pueden ser realizadas de distinta manera en función del propósito de la misma:

- **TrackingOff:** No se proporciona seguimiento a los datos replicados; además, los **Constraints** de la base de datos remota no son tenidos en cuenta para la creación de la misma en el dispositivo móvil.
- **TrackinOn:** Con esta opción sí se proporciona un seguimiento a los datos replicados; para ello, la tabla replicada no solo consta de los campos que se traen de la base de datos remota sino que además se añade una serie de campos para proporcionar este seguimiento. Las restricciones **Primary Key** son tenidas en cuenta y replicadas en la base de datos del dispositivo móvil, no así los **Indexes**.
- **TrackingOffWithIndexes:** Al igual que con **TrackingOff**, no se proporciona seguimiento a los datos replicados, aunque en este caso las restricciones de integridad referencial sí son añadidas a la tabla creada en el dispositivo móvil.
- **TrackingOnWithIndexes:** Lo mismo que **TrackingOn** y a mayores los **Indexes**.

3. Operación Push

El proceso de **Push** actualiza los cambios producidos en la base de datos del dispositivo móvil en la base de datos remota. Si durante esta transacción ocurre algún tipo de fallo, ésta es capaz de realizar un **Rolleback** a su estado original. Existen 2 formas de tratar el Push de datos:

- **RdaBachingOn:** Mediante esta opción todas las filas del Push se procesan en una única transacción.
- **RdaBachingOff:** Este es el valor por defecto; con esta opción el Push se hace de cada fila independientemente.

Uno de los problemas que surgen cuando se realizan las operaciones de Push en el servidor remoto es la duplicación o los problemas de tratamiento de las claves primarias. Veremos para el caso de 'Merge' Replication cómo resolver este problema mediante **IDENTITIES** y rangos para cada subscritor. Desgraciadamente, RDA no soporta el tratamiento de IDENTITIES mediante la opción **not for replication** de SQL Server. ¿Qué hacemos entonces cuando es habitual la presencia de clientes en el mismo conjunto de datos, los cuales pueden insertar nuevos datos? La respuesta a esta pregunta sería 'Merge' Replication; aún así, si nos decidimos por RDA, debemos proporcionar los métodos necesarios para que cada cliente sea capaz de crear identificadores únicos y que éstos no se puedan imitar en el resto de los clientes.

Actividad

Desarrollar una aplicación móvil que realice lo procesos pull y push. Tener en cuenta que se debe crear una base de datos con su primary key obligatoriamente y usar sql Server 2005. Recuerda crear el directorio virtual con el agente SQLCE.

Directorio Virtual

La configuración del servicio para Internet Information Services (IIS) requiere de los siguientes pasos:

1. Creamos un directorio virtual en IIS;
2. Especificamos el **alias** para ese directorio virtual;
3. Especificamos la ruta de acceso al contenido para ese directorio virtual.

En este directorio debe existir una copia del **Agente de SQL Server CE (sqlcesa.dll)**.

4. Concedemos el permiso de ejecución para este directorio;
5. Configuración de la autenticación de IIS: SqlServerCE soporta 3 métodos de autenticación:

- Acceso Anónimo
- Basic Authentication
- Integrated Windows

Base de datos

```
CREATE DATABASE BDCIBERTEC
GO
USE BDCIBERTEC
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[Pedido]') AND type in (N'U'))
BEGIN
CREATE TABLE [dbo].[Pedido](
    [Id_Pedido] [int] IDENTITY(1,1) NOT NULL,
    [Id_Producto] [int] NULL,
    [Cantidad] [int] NULL,
    CONSTRAINT [PK_Pedido] PRIMARY KEY CLUSTERED
(
    [Id_Pedido] ASC
)WITH (IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
END
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[Producto]') AND type in (N'U'))
BEGIN
CREATE TABLE [dbo].[Producto](
    [ID_Producto] [int] NOT NULL,
    [NombreProd] [varchar](50) NULL,
    [CantidadProd] [int] NULL,
    CONSTRAINT [PK_Producto] PRIMARY KEY CLUSTERED
(
```



```
[ID_Producto] ASC
)WITH (IGNORE_DUP_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
END
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
IF NOT EXISTS (SELECT * FROM sys.objects WHERE object_id =
OBJECT_ID(N'[dbo].[realizar_pedido]') AND type in (N'P',
N'PC'))
BEGIN
EXEC dbo.sp_executesql @statement = N'create proc
[dbo].[realizar_pedido]
(
@id int,
@cant int
)
as
begin
insert into pedido values (@id,@cant)
update producto set cantidadprod=(cantidadprod-@cant) where
id_producto=@id
end'
END
```

Formulario Pedidos

Antes de realizar la siguiente parte del ejemplo, recuerda hacer referencia a las librerías `System.Data`, `System.Data.SqlClient` y `System.Xml`. También tener en cuenta que los datos del servidor deben estar de acuerdo al servidor al cual se va a apuntar.

```
Imports System.Data
Imports System.Data.SqlServerCe
Imports System.IO

Public Class Form_Pedido

    Private ispull As Boolean

    Private rda As SqlCeRemoteDataAccess

    Private id As Integer

    'Linea de Conexión con el servidor

    Private rdaOleDbConnectionString As String = _
        "Provider=SQLOLEDB;Data Source=192.168.1.34;Initial " _
        & "Catalog=DBCibertec;user Id=sa;Password=abcDEF123"

    Private Sub Form_Pedido_Load(ByVal sender As _
        System.Object, ByVal e As System.EventArgs) _
        Handles MyBase.Load

        Crear_Base_Datos()

        'Se instancian los parametros para

        'el RDA

        rda = New SqlCeRemoteDataAccess()

        rda.InternetLogin = String.Empty

        rda.InternetPassword = String.Empty

        'Url del directorio virtual con el agente

        'del SQLCE

        rda.InternetUrl = _
            "http://192.168.1.34/sqlce/sqlcesa35.dll"

        'Linea de conexion con la Base de datos de SQLCE

        rda.LocalConnectionString = _
            "Data Source=\My Documents\DBCibertec.sdf"

        ispull = False

    End Sub

    Public Sub Crear_Base_Datos()

        'Se consulta si existe la BD para crearla
        If Not File.Exists("\My Documents\DBCibertec.sdf") Then

            Try

                Dim Engine As SqlCeEngine
```

```
'Ruta de creacion de la BD en el dispositivo
Engine = New SqlCeEngine( _
    "Data Source = \My Documents\DBCibertec.sdf")

'este es el comando que crea la base de datos

Engine.CreateDatabase()

Catch ex As Exception

    MessageBox.Show("Base de Datos no creada")

End Try

End If

End Sub

Private Sub MenuItem_Traer_Click(ByVal _
    sender As System.Object, ByVal e As _
    System.EventArgs) Handles MenuItem_Traer.Click

    Traer_Datos()

End Sub

Public Function Llenar_datos() As DataSet

    'Metodo para llamar todos los registros

    'de una tabla

    Dim cn As SqlCeConnection

    'Se establece la conexion con la BD

    cn = New SqlCeConnection( _
        "Data Source=\My Documents\DBCibertec.sdf")

    cn.Open()

    Dim cmd As SqlCeCommand = cn.CreateCommand

    'Se establece la consulta
    cmd.CommandText = "select Id_Producto as ID," _
        & "NombreProd as Nombre, CantidadProd" _
        & " as Cantidad from producto"

    Dim da As New SqlCeDataAdapter(cmd)

    Dim ds As New DataSet

    da.Fill(ds)

    cn.Close()

    Try

    Catch ex As Exception
```

```
End Try

'Se retorna el dataset lleno con los datos
Return ds

End Function

Private Sub Grid_Producto_Click( _
ByVal sender As Object, ByVal e As _
System.EventArgs) Handles Grid_Producto.Click

    Cargar_Texto()

    TextBox_Cantidad.Enabled = True

End Sub

Private Sub Button_Pedido_Click(ByVal _
sender As System.Object, ByVal e As _
System.EventArgs) Handles Button_Pedido.Click

    'Verificamos si la cantidad pedida
    'es suficiente

    If Grid_Producto.Item(Grid_Producto. _
CurrentRowIndex, 2) < TextBox_Cantidad.Text Then

        MessageBox.Show( _
            "No hay suficientes Productos para el pedido")

    Else

        rda.SubmitSql("exec realizar_pedido " & id _
            & "," & Integer.Parse(TextBox_Cantidad.Text), _
            rdaOleDbConnectionString)

        TextBox_Cantidad.Text = ""

        TextBox_Producto.Text = ""

        Traer_Datos()

    End If

End Sub

Public Sub Cargar_Texto()

    'Capturamos el id y el nombre
    'de la de la grilla

    id = Grid_Producto.Item( _
Grid_Producto.CurrentRowIndex, 0)

    TextBox_Producto.Text = _
Grid_Producto.Item(Grid_Producto. _
CurrentRowIndex, 1)

End Sub
```

```
Public Sub Traer_Datos()  
  
    'Se consulta si ya se hizo un pull  
  
    'anteriormente  
  
    If ispull Then  
  
        Try  
  
            Dim cn As SqlConnection  
  
            'Se establece la conexion con la BD  
  
            cn = New SqlConnection( _  
                "Data Source=\My Documents\DBCibertec.sdf")  
  
            cn.Open()  
  
            Dim resultado As Integer  
  
            'Consultas para eliminar las tablas creadas  
  
            'despues de traer los datos  
  
            Dim cmd As SqlCommand = cn.CreateCommand()  
  
            Dim cmd2 As SqlCommand = cn.CreateCommand()  
  
            cmd.CommandText = "select count (*) from producto"  
  
            cmd2.CommandText = "DROP TABLE producto"  
  
            resultado = cmd.ExecuteScalar  
  
            If resultado >= 0 Then  
  
                cmd2.ExecuteNonQuery()  
  
            End If  
  
            cn.Close()  
  
            'Se establecen los parametros para traer los datos  
  
            rda.Pull("Producto", "select * from Producto", _  
                rdaOleDbConnectionString, RdaTrackOption. _  
                TrackingOnWithIndexes, "ErrorTable")  
  
            Grid_Producto.DataSource = Llenar_datos.Tables(0)  
  
            Catch _ERR As SqlCeException  
  
                MsgBox(_ERR.Message.ToString)  
  
            End Try  
  
        Else  
  
            'Si no se ha hecho pull no se realiza el borrado de tablas
```

```
Try
    rda.Pull("Producto", "select * from Producto", _
    rdaOleDbConnectionString, RdaTrackOption. _
    TrackingOnWithIndexes, "ErrorTable")

    MenuItem_Salir.Enabled = True

    ispull = True

    Grid_Producto.DataSource = Llenar_datos.Tables(0)

Catch ex As Exception

    MsgBox(ex.Message.ToString)

End Try

End If

End Sub

Private Sub MenuItem_Salir_Click(ByVal sender As _
System.Object, ByVal e As System.EventArgs) _
Handles MenuItem_Salir.Click

    Me.Close()

End Sub

Private Sub TextBox_Cantidad_TextChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
TextBox_Cantidad.TextChanged

    'validamos si la caja de texto cantidad

    'esta vacia para desabilitar el boton

    'pedido, si no se abilita

    If TextBox_Cantidad.Text = "" Then

        Button_Pedido.Enabled = False

    Else

        Button_Pedido.Enabled = True

    End If

End Sub

Private Sub TextBox_Cantidad_KeyPress(ByVal _
sender As Object, ByVal e As _
System.Windows.Forms.KeyPressEventArgs) _
Handles TextBox_Cantidad.KeyPress

    'validamos que el textbox acepte solo




    'las teclas de numeros
```

```
If (Microsoft.VisualBasic.Asc(e.KeyChar) < 48) _  
Or (Microsoft.VisualBasic.Asc(e.KeyChar) > 57) Then  
  
    e.Handled = True  
  
End If  
  
If (Microsoft.VisualBasic.Asc(e.KeyChar) = 8) Then  
  
    e.Handled = False  
  
End If  
  
End Sub  
  
End Class
```

Autoevaluación

- ¿En qué consiste el RDA?
- ¿Cómo funcionan el Pull process y el push process?

Para recordar

-  El RDA utiliza los Servicios de Microsoft Internet Information Server (IIS) como mecanismo de comunicación entre la base de datos de SQL Server en el servidor y la base de datos de SQL Server Compact Edition en el dispositivo.
-  Recordar que RDA no sincroniza con SQLEXPRESS ya que ese último no brinda la capacidad de sincronización.
-  Si desea saber más acerca de estos temas, puede consultar las siguientes páginas.

 <http://msdn.microsoft.com/es-es/library/bb972256.aspx>

**UNIDAD DE
APRENDIZAJE**

2

SEMANA

13

Sincronización de Merge-Replication con SQL Server de escritorio

OBJETIVOS ESPECÍFICOS

- Conocer los procesos de Sincronización y Merge

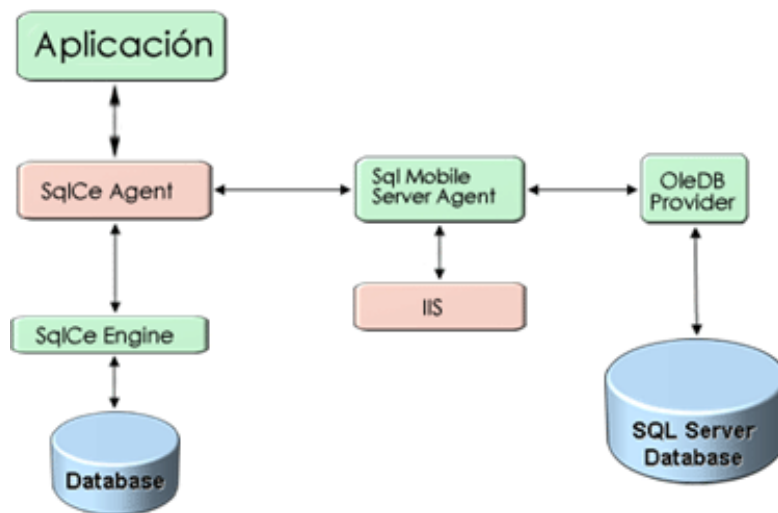
CONTENIDOS

- Qué es la sincronización de datos.
- 'Merge' Replication.

1. Qué es la sincronización de datos

La sincronización de datos entre un dispositivo móvil y una base de datos remota es un proceso complicado que requiere de un estudio exhaustivo para realizar una buena elección de las técnicas a utilizar.

Con la llegada de **SQL Server 2008 Mobile Edition**, la arquitectura del sistema utiliza el agente móvil del cliente SQL para tener acceso a la base de datos local a través del motor móvil de SQL. Sin embargo, para la sincronización con una base de datos remota es necesaria una petición desde el cliente móvil al servidor SQL con los servicios que proporciona IIS (**Internet Information Services**); el cliente móvil entonces puede utilizar **OLE DB** para acceder al proveedor, el cual responderá con los datos solicitados usando la misma cadena de objetos.



2. 'Merge' Replication

'Merge' Replication es una técnica ideal en dispositivos móviles, ya que aporta autonomía e independencia al dispositivo a la vez que facilita el sincronismo de los datos cuando desean ser volcados al servidor. Dentro de esta técnica se deben distinguir 2 miembros, los **Publicadores** y los **Subscriptores**. Los Publicadores envían los datos y éstos son recibidos por los Subscriptores; en el caso que nos ocupa, el Publicador es la base de datos remota, y el Subscriptor es la base de datos del dispositivo móvil.

En un entorno real, los datos tanto en local como en la base de datos cambian con el tiempo; empleando este modelo, la sincronización de los datos se realiza tanto en el servidor remoto como en los clientes, recuperando datos nuevos o las modificaciones de los datos existentes.

Si bien es cierto que el despliegue y el mantenimiento de 'Merge' Replication requiere de mucho trabajo, esta técnica tiene ciertas ventajas:

- La replicación posee características para resolver los conflictos de sincronización.
- Permite la sincronización de datos de múltiples tablas en 'un tiempo'. En RDA esto no era posible, únicamente se hacía un Pull del conjunto de datos a traer.
- Permite el monitoreo de cada publicación.

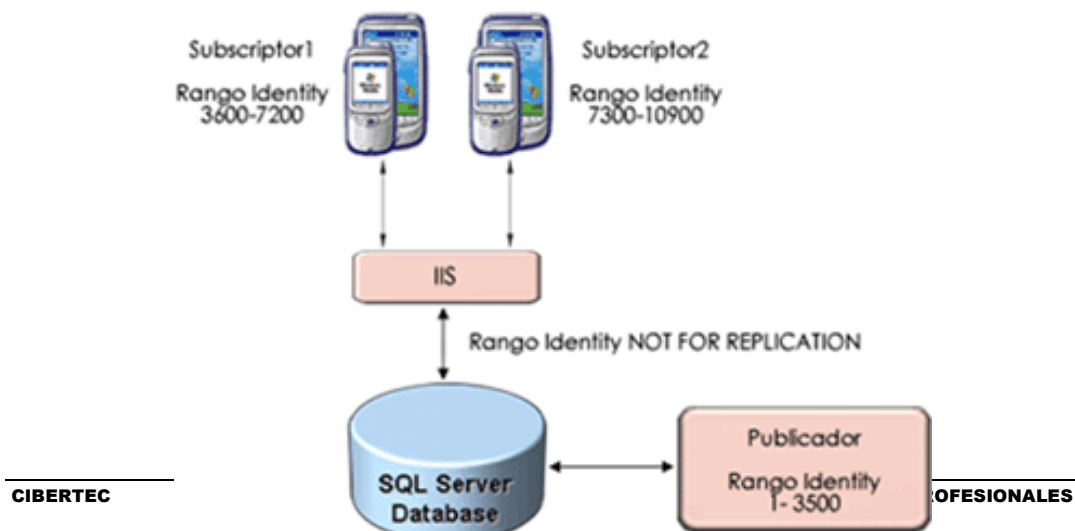
Es importante comprender y tener en cuenta que la replicación crea una cantidad de carga notable en el servidor. Cuando una base de datos se agrega como Publicador, la **Metadata** de dicha base de datos es modificada creando diversos **Disparadores** y **Procedimientos Almacenados** para facilitar la sincronización y la resolución de conflictos.

Adicionalmente, a todas las tablas replicadas se les añade un **ROWGUIDCOL** con el fin de mantener las tablas sincronizadas y capacitar a las filas de un identificador único. Esta nueva columna en la tabla causa un aumento del tráfico y del tamaño de la memoria, por ejemplo en una tabla con únicamente 32 registros, el aumento al añadir el ROWGUIDCOL es de $1\text{kb} = (16\text{ bytes en el registro} + 16\text{ bytes en el índice}) * 32$.

Para reducir el coste de la replicación, es recomendable reducir la cantidad de datos que se van a sincronizar fijando los filtros de los datos que se van a publicar. Como comentamos, en RDA no está soportado el uso de IDENTITIES; sin embargo en 'Merge' Replication su uso es una buena técnica de creación de claves primarias. Las columnas en SQL Server pueden ser marcadas con la restricción **not for replication** la cual permite inserciones de valores para un **identity** fuera del **seed** actual. El método a seguir consistirá en asignar rangos de identities al Publicador y a cada Subscriptor, estableciendo la restricción **not for replication** en la tabla de SQL Server. El rango para cada **servidor** se usa empleando el **constraint check**; un ejemplo de esto sería:

ALTER TABLE WITH NOCHECK id_table ADD CONSTRAINT


id_range_check CHECK (ID BETWEEN 301 and 400)




Autoevaluación

- ¿Qué es la sincronización de datos?
- ¿Qué es Merge?

Para recordar

 Es importante comprender y tener en cuenta que la replicación crea una cantidad de carga notable en el servidor. Cuando una base de datos se agrega como Publicador, la Metadata de dicha base de datos es modificada creando diversos Disparadores y Procedimientos Almacenados para facilitar la sincronización y la resolución de conflictos.

 Si desea saber más acerca de estos temas, puede consultar las siguientes páginas.

 <http://msdn.microsoft.com/es-es/library/bb972256.aspx>

**UNIDAD DE
APRENDIZAJE**

3

SEMANA

14

Empaquetado y despliegado de aplicaciones

OBJETIVOS ESPECÍFICOS

- Conocer el proceso de empaquetado y despliegue de aplicaciones móviles.

CONTENIDOS

- Información general sobre el empaquetado de soluciones para dispositivos.
- Empaquetar una solución Smart Device para su implementación.

1. Información general sobre el empaquetado de soluciones para dispositivos

Se conoce como implementación el proceso mediante el cual se transfiere una aplicación o componente al dispositivo o dispositivos de destino donde se pretende que se ejecuten. Antes de poder implementar una solución determinada, hay que empaquetarla en un archivo CAB. Un archivo CAB es un tipo de archivo de almacenamiento ejecutable que puede contener una determinada aplicación, dependencias como archivos DLL, recursos, archivos de ayuda y cualquier otro tipo de archivo que se desea incluir. Al generar un proyecto CAB, Microsoft Visual Studio 2008 genera un archivo INF que se utiliza, a su vez, para crear el archivo CAB. El archivo INF describe las carpetas en las que se va a instalar cada archivo, las versiones de Windows CE en las que se desea ejecutar la aplicación, si se permite desinstalar la aplicación, etcétera. También es posible incluir en el archivo CAB un archivo DLL nativo personalizado para realizar cualquier paso de la instalación personalizada como la comprobación del número de versión de Windows CE o .NET Compact Framework, determinando si hay otros componentes presentes, etcétera. Tras copiar el archivo CAB en el dispositivo de destino, el usuario hace clic en él para comenzar el proceso de instalación. Esto se conoce como explotar el archivo CAB.

Visual Studio 2008 hace posible, en la mayoría de los casos, la realización de todo el trabajo de empaquetado necesario directamente en el entorno de desarrollo integrado (IDE) de Visual Studio. Es posible crear un archivo CAB, primero, agregando un proyecto CAB para Smart Device a la solución existente, a continuación, agregando archivos, accesos directos y entradas de registro con la misma interfaz de usuario utilizada con proyectos de instalación de escritorio. Al generar el proyecto de instalación, se crea el archivo CAB.

Hay algunas diferencias entre los archivos CAB files creados para una aplicación Pocket PC y aquellos otros creados para una aplicación Smartphone. Los archivos Pocket PC basados en Windows Mobile 2003SE o versiones anteriores no admiten archivos CAB comprimidos o archivos CAB firmados. Los archivos CAB de Smartphone deben comprimirse y tanto el

archivo EXE como el archivo DLL, y el propio archivo CAB, deben firmarse digitalmente antes de poder ser instalados en el dispositivo.

Una vez que creado el archivo CAB con Visual Studio, el siguiente paso consiste en transferirlo al dispositivo de destino utilizando cualquiera de los medios habituales para transferir archivos: solicitudes FTP o HTTP desde el dispositivo, copia manual desde el equipo de desarrollo de escritorio en una carpeta de un dispositivo conectado utilizando Windows Explorer, transferencia OTA para Smartphones, etcétera.

2. Empaquetar una solución Smart Device para su Implementación

2.1 Establecer el proyecto Cab

Para agregar un proyecto Cab de Smart Device a la solución:

1. Abra el proyecto de Smart Device existente y asegúrese de que el **Explorador de soluciones** se encuentra visible.
2. En el menú **Archivo**, elija **Agregar** y, después, haga clic en **Nuevo proyecto**.

Aparecerá el cuadro de diálogo **Agregar nuevo proyecto**.

3. En el panel **Tipos de proyecto** que aparece en la izquierda, expanda el nodo **Otros tipos de proyectos** y, a continuación, haga clic en **Instalación e implementación**.
4. En el panel **Plantillas** que aparece a la derecha, seleccione **Proyecto CAB de Smart Device**.

Éste es el único tipo de proyecto Cab válido para dispositivos inteligentes. Los otros tipos de proyecto son sólo para soluciones de escritorio.

5. En el cuadro **Nombre**, escriba **CABProject** y, a continuación, haga clic en **Aceptar**.

El proyecto Cab se agrega a la solución pertinente y se encuentra visible en el **Explorador de soluciones**. Ahora aparecen los dos paneles del **Editor del sistema de archivos**.

2.2 Personalizar el proyecto Cab

Para cambiar el nombre del producto y otras propiedades del proyecto

1. En el **Explorador de soluciones**, seleccione **CABProject** si todavía no se encuentra seleccionado.
2. En el menú **Ver**, haga clic en **Ventana Propiedades** para abrir la ventana **Propiedades**.
3. En el campo **ProductName** de la cuadrícula de propiedad, cambie el valor a **MyProduct**.

El valor de la propiedad **ProductName** determina el nombre que se mostrará para la aplicación en los nombres de carpeta y en el cuadro de diálogo **Agregar o quitar programas**.

- También es posible utilizar esta ventana para cambiar el nombre del fabricante y especificar las versiones mínima y máxima del sistema operativo.
- Es posible establecer la propiedad **OSVersionMin** en 4.21 para indicar que la aplicación para Pocket PC tiene conocimiento sobre la orientación de pantalla. Sin embargo, estableciendo esta propiedad en 4.21 se evitará que la aplicación se instale en Pocket PC basada en Windows Mobile 2003 y anterior. Para permitir que se realice la instalación en tales dispositivos y, a la vez, indicar el conocimiento sobre la orientación de la pantalla hacia los dispositivos más actuales, es necesario editar de forma manual el archivo .inf con el fin de establecer la propiedad **BuildMax** en uno de los siguientes valores:

- 0xA0000000 para indicar que la aplicación admite pantallas cuadradas (240x240 píxeles)
- 0xC0000000 para indicar que la aplicación admite la rotación de pantallas

O bien

0xE0000000 para indicar que la aplicación admite pantallas cuadradas y la rotación de pantallas.

- Para soluciones Pocket PC basadas en Windows Mobile 2003SE y anteriores, las propiedades **Compress** y **NoUninstall Device Deployment** deben ser falsas. Hay que tener en cuenta que esta opción se puede establecer en **true** para dispositivos equipados con Compact Framework 3.5.
- Si se está utilizando una DLL de instalación de Windows CE, utilice esta cuadrícula de propiedad para especificar el nombre de archivo y la ubicación. Para obtener más información sobre las DLL de instalación de Windows CE, consulte la documentación relacionada con Pocket PC o Smartphone SDK.

2.3 Para cambiar el nombre del archivo CAB y agregar autenticación

1. En el **Explorador de soluciones**, haga clic con el botón secundario del mouse en **CABProject** y, a continuación, en **Propiedades**.

Aparecerá el cuadro de diálogo **Páginas de propiedades** del proyecto Cab pertinente. En el cuadro **Nombre del archivo de resultados**, cambie el nombre del archivo CAB y la ruta de acceso a **Debug\MyApp.cab** y, a continuación, haga clic en **Aceptar**.

2. También es posible utilizar esta página de propiedades para agregar autenticación al proyecto en cuestión. Es necesaria la

autenticación para soluciones Smartphone y no se admite en soluciones Pocket PC basadas en Windows Mobile 2003 SE y anteriores.

2.4 Para agregar la aplicación del proyecto de dispositivos al proyecto Cab

1. En el panel que se encuentra a la izquierda del **Editor del sistema de archivos**, seleccione el nodo **Carpeta de la aplicación** con el fin de especificar que los archivos que se seleccionen en los siguientes pasos se instalarán en la carpeta del dispositivo de destino.

Si el editor del **Sistema de archivos** no se encuentra visible, haga clic con el botón secundario del mouse en el nombre del proyecto Cab del **Explorador de soluciones**, seleccione **Ver** y haga clic en **Sistema de archivos**.

2. En el menú **Acción** de Visual Studio, elija **Agregar** y, a continuación, haga clic en **Resultados del proyecto**.
3. En el cuadro de diálogo **Agregar grupo de resultados del proyecto**, seleccione el proyecto de Smart Device correspondiente de la lista desplegable **Proyecto**.
4. En la lista de resultados, seleccione **Resultado principal** y, a continuación, haga clic en **Aceptar**.

2.5 Para crear un acceso directo para la aplicación del proyecto de dispositivos

1. En el panel derecho del **Editor del sistema de archivos**, seleccione **Resultado principal** desde **<nombre del proyecto de la aplicación>**.

2. En el menú **Acción**, seleccione **Crear acceso directo al resultado primario a partir del <nombre del proyecto de la aplicación>**.

Este comando agrega un elemento Acceso directo debajo del elemento Resultados.

3. Haga clic con el botón secundario del mouse en el elemento Acceso directo, haga clic en **Cambiar nombre** y cambie el nombre del acceso directo a algún nombre más apropiado para un acceso directo.

2.6 Para agregar una entrada de Registro

1. En el **Explorador de soluciones**, seleccione el proyecto Cab.
2. En el menú **Ver**, elija **Editor** y, a continuación, haga clic en **Registro**.
3. En el **Editor del Registro**, haga clic con el botón secundario del mouse en HKEY_CURRENT_USER y, a continuación, haga clic en **Nueva clave** en el menú contextual.
4. Cuando aparezca la entrada **Nueva clave** en el **Editor del Registro**, cámbiele el nombre a **SOFTWARE**.
5. Haga clic con el botón secundario del mouse en esta nueva clave, señale **Nueva** y, a continuación, haga clic en **Clave**.
6. Cuando aparezca la entrada **Nueva clave** en el **Editor del Registro**, cámbiele el nombre a **MyCompany**.
7. Haga clic con el botón secundario del mouse en la entrada **MyCompany** y, a continuación, haga clic en Ventana Propiedades del menú contextual.

El valor **Name** ha cambiado a **MyCompany**.

2.7 Generar e implementar el archivo CAB

1. En el menú **Generar**, haga clic en **Generar CABProject**.

O bien

Haga clic con el botón secundario del mouse en **CABProject** en el **Explorador de soluciones** y, a continuación, haga clic en **Generar**.

2. En el menú **Archivo**, haga clic en **Guardar todo**.

Los archivos CAB para soluciones Smartphone deben estar firmados digitalmente antes de implementarse en un dispositivo de usuario final. La firma digital no se admite en soluciones Pocket PC basadas en Windows Mobile 2003SE y anteriores.


2.8 Para implementar el archivo CAB en el dispositivo


1. En el **Explorador de Windows**, desplácese a la carpeta donde se encuentra almacenada esta solución. Encontrará el archivo CAB en la carpeta **CABProject\Release** de la solución.
2. Copie el archivo CAB en un dispositivo que se encuentre conectado a ActiveSync 4.0 o posterior.
3. Cuando un usuario determinado puntee sobre el nombre del archivo CAB en el **Explorador de archivos** del dispositivo,
4. Windows CE hará explotar el CAB y procederá a la instalación de la aplicación en el dispositivo.


Autoevaluación

- ¿A qué se le conoce como implementación?

Para recordar

 Visual Studio 2008 hace posible, en la mayoría de los casos, la realización de todo el trabajo de empaquetado necesario directamente en el entorno de desarrollo integrado (IDE) de Visual Studio. Es posible crear un archivo CAB, primero, agregando un proyecto CAB para Smart Device a la solución existente, a continuación, agregando archivos, accesos directos y entradas de registro con la misma interfaz de usuario utilizada con proyectos de instalación de escritorio. Al generar el proyecto de instalación, se crea el archivo CAB.

 Si desea saber más acerca de estos temas, puede consultar las siguientes páginas.

 [http://msdn.microsoft.com/es-es/library/sz5xy1zx\(VS.80\).aspx](http://msdn.microsoft.com/es-es/library/sz5xy1zx(VS.80).aspx)