

Digi3D 2011 en profundidad

Digi3D ha cambiado mucho en las dos últimas versiones.

Cuando pasamos de 2005 a 2007 pudimos ver un cambio muy importante en lo referente a sensores fotogramétricos, pasamos de tener una aplicación que únicamente nos permitía trabajar con pares estereoscópicos de cámara cónica a una aplicación fotogramétrica que admite múltiples sensores, de una aplicación que requería mucha memoria porque cargaba las imágenes completas a otra que cargaba las imágenes a toda velocidad y sin agotar nunca la memoria, de tener una aplicación que únicamente podía trabajar con archivos .bin a otra que nos permitía trabajar con archivos .dwg y .dgn de forma nativa...

En esta nueva versión, 2011, se han añadido sensores nuevos como el sensor Web Map Service, se han realizado muchos cambios que nos van a permitir trabajar con modelos de datos modernos, con codificación múltiple, se ha añadido un motor de bases de datos y lo que personalmente más me enorgullece, se ha convertido en una aplicación completamente programable. Puedes programar hasta tu propio sensor si quieres o tu propio modelo de distorsiones de cámaras, o añadir órdenes que realicen lo que quieras, programar tu propio importador/exportador de archivos de dibujo, lo que quieras.

Ahora tienes las mismas herramientas y juegas en igualdad de condiciones que nosotros, el equipo de Digi21.

Este libro está pensado para personas que quieran tener conocimientos avanzados de Digi3D 2011. Arquitectos de modelos de datos, responsables de edición y técnicos de soporte técnico de Digi3D 2011.

Está escrito en un lenguaje muy cercano y mi intención al escribirlo es que conozcas en profundidad el programa. Muchos capítulos pueden ser difíciles de comprender. Otros sin embargo te parecerán muy sencillos.

Advertencia:

Este es un libro pensado para personas con conocimientos previos de Digi3D, **no es un tutorial de Digi3D**. Si no conoces Digi3D, este libro no está pensado para ti. Este libro no te va a enseñar a instalar Digi3D o cómo se ejecuta una orden o cómo se configura el programa. Puedes obtener ayuda en <http://ayudaonline.digi21.net> y realizar consultas de soporte técnico en <http://incidencias.digi21.net> o en la página de soporte técnico de tu país.

Se requieren conocimientos básicos de XML para comprender y poder realizar satisfactoriamente todos los laboratorios de este libro. No hace falta que seas un experto. No es necesario saber XPath, ni DOM, ni SAX... Con que sepas cómo crear un XML bien formado, el concepto de nodo, atributo y espacio de nombres es suficiente. Si quieres aprender XML, un buen sitio (si sabes inglés) es <http://www.w3schools.com/xml/default.asp>

Se requieren conocimientos de bases de datos y de SQL. Tienes que saber qué es una base de datos, qué es un sistema gestor de bases de datos, qué es una tabla, un campo, una clave

principal, una relación, y saber hacer consultas básicas de SQL,... Busca en Google “Tutorial SQL” y descubrirás muchos y muy buenos tutoriales de SQL.

Sin embargo no es necesario tener conocimientos de TSQL (lenguaje SQL extendido por Microsoft que convierte el lenguaje de consultas SQL en un lenguaje de programación) ya que el libro hace una pequeña introducción más que suficiente para seguir los ejercicios con facilidad.

La mayoría de ejemplos de bases de datos se van a realizar con el servidor de bases de datos *Microsoft SQL Express* aunque no es necesario conocimientos de este servidor.

Puedes descargar la versión gratuita de la siguiente dirección:

<http://www.microsoft.com/express/Database/InstallOptions.aspx>

Descarga e instala la versión denominada *Database with Management Tools* porque vamos a ejecutar muchas veces la herramienta *Microsoft SQL Server Management Studio*, y esta herramienta no se instala con la versión normal (que únicamente incluye el servidor de bases de datos pero ninguna herramienta de administración).

Si quieres sacarle todo el partido a este libro, convendría que tuvieras conocimientos de programación en cualquier lenguaje .NET. Los ejemplos del libro están programados en C# (**y si Manuel Quirós me los traduce a Visual Basic pues estarán también en Visual Basic**).

Como herramientas de programación vamos a utilizar *Microsoft Visual C# 2010 Express*, *Microsoft Visual Basic Express* y SharpDevelop.

Puedes descargar Microsoft Visual C# 2010 Express de la dirección:

<http://www.microsoft.com/express/Downloads/#2010-Visual-CS>

Puedes descargar Microsoft Visual Basic 2010 Express de la dirección:

<http://www.microsoft.com/express/Downloads/#2010-Visual-Basic>

Este es libro comenzó a escribirse en Yakarta el día 18 de agosto de 2010 en casa de mi amigo Jon, y se terminó de escribir en Madrid, el día __/__/cuando lo termine

Desy, por esa paciencia infinita que tienes

Contenido

1. Codificación múltiple en Digi3D 2011	6
Codificación múltiple	7
Ventana de Códigos Activos	8
La orden COD ahora admite múltiples parámetros	11
Editor de códigos	11
Etiquetas en los códigos	13
Traducción de códigos incluso a nivel de atributo	15
Codificación múltiple en archivos de dibujo de terceros	18
2. Acceso a Bases de datos	42
Enlace con bases de datos	49
Códigos desde el punto de vista de Digi3D 2011	57
Cada entidad genera su propio conjunto de registros en la base de datos ¡Error! Marcador no definido.	
Conexión con la base de datos	51
Almacenando información en la base de datos en entidades nuevas	53
Editando información existente	65
El proceso de almacenar una entidad en profundidad	75
Cambios en Digi.tab.xml	80
Gestores de Modelo de Datos en Base de Datos	81
Conexión con la base de datos en profundidad	88
Creación del objeto Modelo de Datos	88
Conexión con la base de datos	90
Comprobación de compatibilidad Base de Datos <-> Modelo de Datos	91
Comprobación de compatibilidad Digi.Tab <-> Modelo de Datos	100
El campo clave primaria	111
Configurando los esquemas en la tabla de códigos	114
Configurando restricciones en el servidor	137
Apéndice 1: Sustituidores	148

Apéndice 2: Tutorial de TSQL	157
Aspectos generales del lenguaje TSQL	157
Impresión de información en la consola de Microsoft SQL Server Management Studio	157
Variables TSQL	158
Funciones TSQL	160
Bucles TSQL	162
Procedimientos almacenados TSQL	163
Instrucciones de control de flujo TSQL	165
Comunicación de errores TSQL	167
Captura de errores en TSQL	168
Funciones que únicamente son válidas en el contexto de un catch	170
Desencadenadores en TSQL	171

1. Codificación múltiple en Digi3D 2011

Digi3D 2011 presenta muchas novedades en lo referente a codificación múltiple con respecto a versiones anteriores.

Este capítulo presenta en profundidad los cambios incorporados al programa para hacer que el proceso de codificación múltiple sea más sencillo que en versiones anteriores presentando una serie de laboratorios para ver en acción todo lo expuesto en el capítulo.

Los laboratorios de este capítulo están ubicados en las carpetas Laboratorio 1, Laboratorio 2, Laboratorio 3, y así sucesivamente de la carpeta *Laboratorio de codificación múltiple*.

Cada laboratorio dispone de su propia tabla de códigos *Laboratorio.tab.xml* y ese patrón se repite en todos los laboratorios: todas las carpetas tienen un archivo *Laboratorio.tab.xml*.

Cada laboratorio se debe ejecutar con su tabla de códigos *Laboratorio.tab.xml* correspondiente.

En Digi3D 2011 podemos aprovecharnos de la característica *sustituidores*¹ para evitar tener que seleccionar la tabla de códigos correspondiente a cada laboratorio, de modo que en el cuadro de diálogo de *Nuevo Proyecto*, en la pestaña *Archivo de dibujo*, en el parámetro *Tabla de códigos* en la sección *Entorno* yo personalmente pondría para todos los laboratorios de este capítulo la siguiente cadena:

`$(PathOfDrawingFile)Laboratorio.tab.xml`

Digi3D 2011 dispone de una nueva ventana denominada *Ventana de menú* que permite cargar archivos con opciones que ejecutan un conjunto de órdenes. Puedes seleccionar la ruta al archivo en la carátula de entrada de DigiNG, en la sección *Entorno*.

Muchos de los laboratorios tienen un menú personalizado para realizar las tareas del propio laboratorio y en todos coincide que el menú está en el propio directorio del laboratorio y su nombre siempre es: *Laboratorio.menu.xml*.

Podemos aprovecharnos aquí también de los sustituidores y poner en el campo de menú la siguiente cadena:

`$(PathOfDrawingFile)Laboratorio.menu.xml`

¹ Consulta el apéndice 2 para ver una lista de posibles sustituidores.

Codificación múltiple

Una de las principales novedades de Digi3D 2011 consiste en que este facilita mucho el crear modelos con codificación múltiple.

Históricamente, las entidades Digi tenían un código principal y se podían añadir códigos secundarios mediante atributos.

Una entidad podía tener hasta 65535 atributos de base de datos, y estos atributos tenían un campo opcional donde se podía almacenar un código, de modo que una entidad podía tener 65536 códigos, el principal + 65535 atributos con el campo "código" relleno.

En versiones anteriores de Digi3D, el formato de los atributos estaba íntimamente relacionado con el formato interno de los archivos .bin, así que vamos a ver cómo es el formato interno de los atributos en un archivo .bin:

Tabla	Registro	Resto
<ul style="list-style-type: none"> • Número de tabla de Base de Datos 	<ul style="list-style-type: none"> • Valor del campo de tipo "clave primaria" para el registro con la información a la que apunta el atributo en la tabla. 	<ul style="list-style-type: none"> • Espacio libre (con espacio para almacenar como máximo 6 caracteres) donde se puede almacenar cualquier tipo de información ASCII, como por ejemplo códigos.

Para añadir un atributo a una entidad, había que ejecutar la orden ALTA que mostraba un cuadro de diálogo preguntando por los valores a asignar en el atributo a los campos Tabla, Registro y Resto. Había que poner un 1 en la tabla, dejar vacío (o poner un 0 en el campo Registro) y por último teclear el código a añadir en el campo Resto.

La siguiente entidad que se almacenara tendría ese atributo por lo tanto tendría doble codificación, la del código que había seleccionado cuando se registró la entidad + el código que aporta el atributo dado de alta antes de finalizar la entidad.

Como la orden ALTA no era una orden pensada para añadir códigos, sino que estaba pensada para añadir atributos de base de datos, no permitía seleccionar códigos de la lista de códigos activos, lo que requería que el operador memorizase y teclease los códigos a añadir.

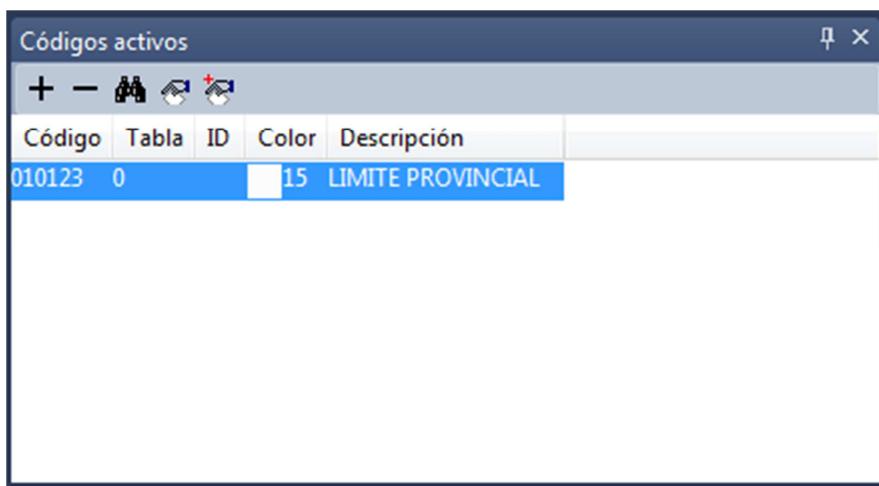
Digi3D 2011 incorpora órdenes específicamente pensadas para codificación múltiple como COD (que ahora permite marcar como código activo un conjunto de códigos), COD+, COD-,

EDITAR_COD ... que muestran cuadros de diálogo que permiten buscar códigos de forma sencilla.

La codificación múltiple está plenamente integrada con el sistema, de modo que las órdenes que realizan procesos con códigos como BORRA_COD, BUSCAR, COMPRIMIR ... tratan cada uno de los códigos que forman parte de una entidad.

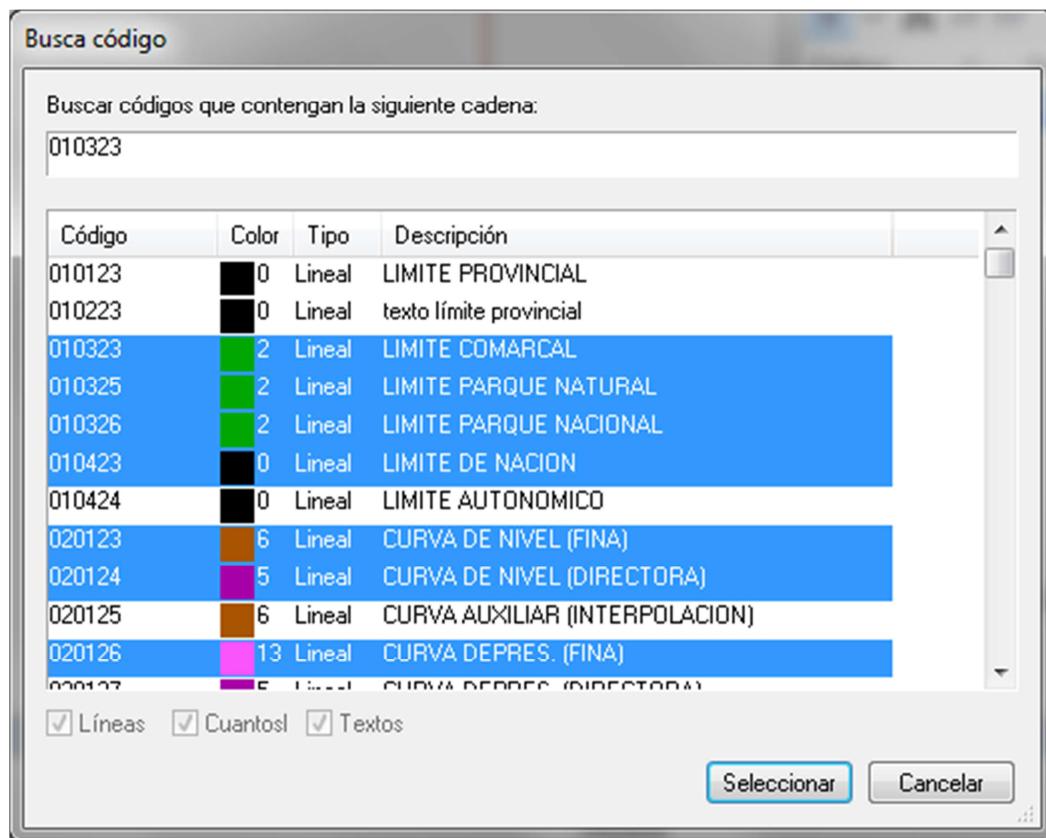
Ventana de Códigos Activos

Ha desaparecido la barra de códigos de Digi3D pero se incorpora una barra acoplable de Códigos activos que permite de forma intuitiva seleccionar los códigos activos que son el conjunto de códigos que se añadirán a las entidades registradas.



Esta ventana dispone de una barra de herramientas con 5 botones que ejecutan órdenes pensadas para codificación múltiple.

1. El botón marcado con el signo más (+), permite añadir códigos a la lista de códigos activa. En realidad lo que hace es ejecutar la orden COD+ que se ha incorporado al conjunto de órdenes de Digi3D y muestra el cuadro de diálogo de selección de códigos que ahora admite selección múltiple como se puede comprobar en la siguiente captura de pantalla:



Si pulsásemos seleccionar con esta selección tras ejecutar la orden COD+, se añadiría a la lista de códigos activos en el momento de ejecutar la orden COD+ la lista de códigos seleccionados en el cuadro de diálogo *Busca código* de modo que la ventana de códigos activos nos mostraría la siguiente información

Códigos activos				
Código	Tabla	ID	Color	Descripción
010123		15	LIMITE PROVINCIAL	
010323		2	LIMITE COMARCAL	
010325		2	LIMITE PARQUE NATURAL	
010326		2	LIMITE PARQUE NACIONAL	
010423		15	LIMITE DE NACION	
020123		6	CURVA DE NIVEL (FINA)	
020124		5	CURVA DE NIVEL (DIRECTORA)	
020126		13	CURVA DEPRES. (FINA)	

2. El botón marcado con el signo menos (-) elimina de la lista de códigos activos el código seleccionado, de modo que si nuestra ventana de códigos activos tiene seleccionado como en la captura anterior el código 010123 y pulsamos el botón menos, la lista de códigos activos pasará a ser:

Código	Tabla	ID	Color	Descripción
010323		2	2	LIMITE COMARCAL
010325		2	2	LIMITE PARQUE NATURAL
010326		2	2	LIMITE PARQUE NACIONAL
010423		15		LIMITE DE NACION
020123		6	6	CURVA DE NIVEL (FINA)
020124		5	5	CURVA DE NIVEL (DIRECTORA)
020126		13	13	CURVA DEPRES. (FINA)

3. El botón de los prismáticos ejecuta la orden COD y lo que hace es sustituir la lista de códigos activos por el/los códigos seleccionados en el cuadro de diálogo de códigos activos, de modo que si lo pulsamos y seleccionamos el código 010123, al aceptar el cuadro de diálogo *Busca código* la lista de códigos pasará a ser

Código	Tabla	ID	Color	Descripción
020123		6	6	CURVA DE NIVEL (FINA)

4. El siguiente botón ejecuta la orden CLONAR_COD, que toma los códigos de la entidad seleccionada y los selecciona como códigos activos, de modo que si seleccionamos una línea con códigos 020123 y 020124, nuestra ventana de códigos activos pasará a ser:
- 5.

Código	Tabla	ID	Color	Descripción
020123		6	6	CURVA DE NIVEL (FINA)
020124	0	5	5	CURVA DE NIVEL (DIRECTORA)

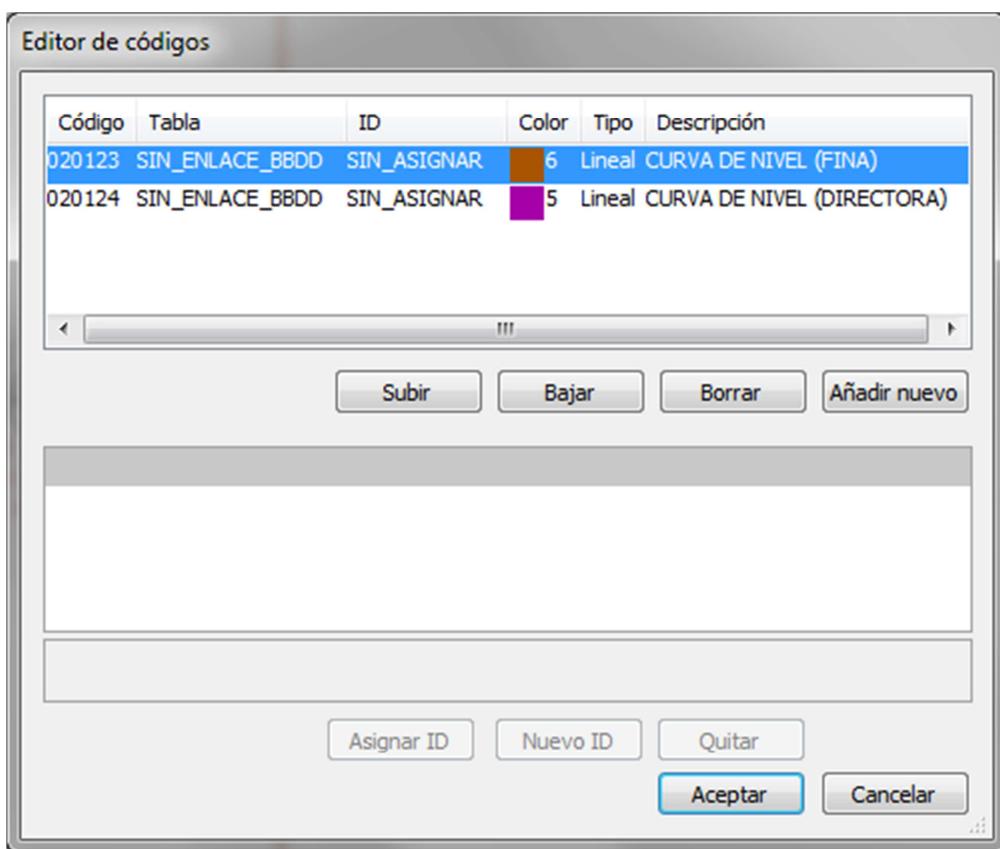
6. El último botón ejecuta la orden CLONAR+ que como te podrás imaginar añade a la lista de códigos activos el o los códigos de la entidad que selecciones.

La orden COD ahora admite múltiples parámetros

La orden COD ahora admite múltiples parámetros, de modo que podemos ejecutar la orden COD=020123 020124 y se seleccionarán como códigos activos estos dos códigos.

Editor de códigos

Se ha añadido además una nueva orden EDITAR_COD que muestra un cuadro de diálogo que permite añadir, eliminar, cambiar el orden de los códigos de una determinada entidad.



El manejo de este cuadro de diálogo es muy sencillo, en la parte superior de este tenemos botones para cambiar el orden² de los códigos, añadir o eliminar códigos y en la parte inferior nos permite cambiar los atributos de base de datos para cada código.

Una entidad se considerará borrada en el momento en el que se eliminan todos sus códigos de modo que podríamos eliminar una entidad seleccionándola con la orden EDITAR_COD, eliminándole todos sus códigos en el cuadro de diálogo *Editor de códigos* y pulsando el botón Aceptar.

² Si cambiamos el orden de los códigos, únicamente estaremos cambiando el orden en que se dibuja cada uno de los códigos de la entidad, nada más. Una entidad con el código "a" seguida del código "b" se considerará idéntica (a nivel de codificación) que otra que tenga el código "b" seguido del código "a". En este caso, el orden de los factores no altera el producto.

Para terminar, se pueden añadir a una entidad tantos códigos como se deseé.

Digi3D 2011 no impone ningún límite al número de códigos, el límite lo impone el formato del archivo de dibujo en el que se almacenan las geometrías. El formato .bin clásico tiene un límite de 65535 códigos por entidad.

Digi3D 2011 permite almacenar codificación múltiple en archivos .bin, .dwg y .dgn.

Laboratorio 1(Familiarizándonos con las órdenes)

El objetivo de este laboratorio es familiarizarnos con las órdenes de codificación múltiple de Digi3D 2011. Cargaremos un archivo de dibujo con una tabla de códigos reducida y añadiremos códigos a entidades existentes, dibujaremos entidades con codificación múltiple y por último eliminaremos entidades mediante el editor de códigos.

1. Localizamos el archivo de [Laboratorio de codificación múltiple]\Laboratorio 1\Laboratorio.tab.xml y hacemos doble clic sobre él para que se abra o con *Internet Explorer* o con el *bloc de notas* para comprobar los códigos que tiene:

Código	Descripción
0	Marco de hoja
Parcela	Límite de parcela
Edificio	Edificio privado
Piscina	Piscina

1. Cerramos *Internet Explorer* o el *bloc de notas*.
2. Ejecutamos Digi3D 2011, abrimos el cuadro de diálogo de *Nuevo Proyecto* y en la pestaña *Archivo de dibujo* seleccionamos el archivo [Laboratorio de codificación múltiple]\Laboratorio 1\Modelo.bin.
3. Nos aseguramos de que la opción seleccionada en el campo *Modelo de datos* es *Sin conexión con base de datos*.
4. Abrimos el archivo de dibujo.
5. Comprobamos que tenemos una parcela con un edificio y una piscina.
6. Ocultamos la visualización de todos los códigos ejecutando la orden OFF=*
7. Activamos la visualización del código de parcela ejecutando la orden ON=Parcela.
8. Podemos comprobar que la parcela está abierta, pues el tramo derecho del edificio no tenía codificación múltiple y por lo tanto estaba dibujado únicamente con el código de Edificio.
9. Volvemos a activar la visualización de todo el modelo ejecutando la orden ON=*
10. Activamos la *Ventana de tentativos* si no está visible mediante la opción del menú *Ver/Ventana de Tentativos*.
11. Hacemos *tentativo* sobre la parte derecha del edificio para comprobar que esa línea únicamente tiene un código: *Edificio*.
12. Ejecutamos la orden EDITAR_COD y seleccionamos la línea derecha del edificio. Al confirmar, Digi3D nos mostrará el cuadro de diálogo *Editor de códigos* mostrándonos únicamente el código que tiene esa línea, que es *Edificio*.

13. Pulsamos el botón titulado *Añadir nuevo*, seleccionamos *Parcela* en la lista de códigos que nos muestra el cuadro de diálogo *Selecciona los códigos a añadir*, aceptamos pulsando el botón *Seleccionar* y comprobamos que ahora el cuadro de diálogo *Editor de códigos* nos está mostrando dos códigos: *Edificio* y *Parcela*. Aceptamos el cuadro de diálogo.
14. Ahora la parte derecha del edificio tiene dos códigos: *Edificio* y *Parcela*. Podemos comprobarlo haciendo *tentativo* sobre la línea en cuestión y comprobando que en la *Ventana de Tentativos* se están mostrando los dos códigos.
15. Para realizar la comprobación gráfica, ejecutamos OFF=* para ocultar la visualización de todas las entidades y ON=Parcela para comprobar que ahora la parcela está cerrada.
16. Ahora vamos a dibujar una línea con dos códigos: *Parcela* y *Piscina*. Para ello, ejecuta la orden COD=Parcela,Piscina
17. Comprueba en la *Ventana de códigos* que los códigos activos son *Parcela* y *Piscina*.
18. Dibuja una línea.
19. Tentatívala para comprobar que tiene los dos códigos.
20. Edítala con la orden EDITAR_COD, elimínale los dos códigos y acepta el cuadro de diálogo *Editor de códigos* para comprobar que Digi3D ha eliminado la línea porque no tiene ningún código.
21. Ejecutamos ON=* y UNDO para asegurarnos de que el archivo de dibujo ha quedado como al principio del ejercicio para el siguiente usuario.
22. Sal de Digi3D.

Etiquetas en los códigos

Digi3D 2011 incorpora una novedad que facilita mucho trabajar con codificación múltiple, y es que ahora se pueden asignar etiquetas a los códigos en la tabla de códigos.

Las etiquetas no son más que nombres, como por ejemplo *Altimetría* o *Usos de Suelo*.

Se pueden asignar tantas etiquetas como se desee a un código, y como las etiquetas pueden tener espacios en blanco (en el ejemplo anterior acabamos de ver que una etiqueta podría ser *Usos de Suelo*), utilizaremos como separador de etiquetas el carácter coma (,).

Si ejecutamos el programa externo Digi.Tab lo podemos hacer en la sección Etiquetas del código, y si editamos manualmente el archivo Digi.tab.xml, tenemos un nuevo atributo denominado *flags*.

Veamos un ejemplo con tres códigos, *curva de nivel maestra*, *curva de nivel fina* y *cota*, todos ellos con la etiqueta *Altimetría* y las curvas con la etiqueta *Curvas*:

```
<code
    name="020400"
    description="Curva de nivel maestra"
    tags="Altimetría,Curvas"
    type ="1">
```

```

</code>
...
<code
    name="020200"
    description="Curva de nivel fina"
    tags="Altimetría,Curvas"
    type ="1">
...
</code>
<code
    name="020401"
    description ="Cota"
    tags="Altimetría"
    type ="1">
...
</code>

```

Todas las órdenes que admiten códigos en sus líneas de comando, ahora admiten también *etiquetas*, y para indicarle a la orden que el parámetro que le estamos pasando es una etiqueta y no un código, ponemos el carácter almohadilla (#) delante del nombre de la etiqueta, de modo que si queremos apagar todos los códigos del archivo de dibujo y encender únicamente los de altimetría podemos ejecutar la siguiente secuencia de comandos:

OFF=*
ON=#Altimetría

Si quisiéramos eliminar todas las entidades con la etiqueta *Curvas* ejecutaríamos la orden

BORRA_COD=#Curvas *

Si quisiéramos seleccionar como código activo todos los códigos que tengan la etiqueta *Altimetría* lo haríamos con la orden

COD=#Altimetría

Laboratorio 2(Familiarizándonos con etiquetas en los códigos)

En este laboratorio vamos a comprobar cómo trabajar con etiquetas en Digi3D 2011. Tenemos un archivo con entidades representables a escalas 1:1000 y 1:5000 y básicamente vamos a activar/desactivar la visualización de códigos ejecutando las órdenes ON y OFF pasándoles como parámetro etiquetas.

La tabla de códigos de este laboratorio incluye los siguientes códigos:

Código	Descripción	Etiquetas
0	Marco de hoja	1000, 5000
020400	Curva de nivel maestra	Altimetría, Curvas, 5000, 1000

020200	Curva de nivel fina	Altimetría, Curvas, 5000, 1000
020401	Cota	Altimetría, 5000, 1000
050146	Edificio privado	Edificaciones, 5000, 1000
050324	División de alturas	Edificaciones, 1000

1. Localizamos el archivo [Laboratorio de codificación múltiple]\Laboratorio 2\Laboratorio.tab.xml para abrirlo haciendo doble clic sobre él y abrirlo con Internet Explorer o el bloc de notas. Desplazamos hasta el final del todo para comprobar que los códigos tienen como etiquetas las indicadas en la tabla anterior.
2. Cerramos Internet Explorer o el bloc de notas.
3. Ejecutamos Digi3D 2011, abrimos el cuadro de diálogo de *Nuevo Proyecto* y en la pestaña *Archivo de dibujo* seleccionamos el archivo [Laboratorio de codificación múltiple]\Laboratorio 2\Modelo.bin.
4. Abrimos el archivo de dibujo.
5. Digi3D 2011 muestra el archivo de dibujo completo y representa un modelo en el que se han dibujado entidades representables a escala 1:1000 y 1:5000. La división de alturas no es representable a escala 1:5000, por eso no tiene la etiqueta 5000 asociada a su código en la tabla de códigos.
6. Vamos a visualizar únicamente las entidades representables a escala 1:5000
7. Ejecutamos la orden OFF=* para ocultar la visualización de todas las entidades del modelo.
8. Ejecutamos la orden ON=#5000 para activar únicamente las entidades con la etiqueta 5000 asociada.
9. Comprobamos que Digi3D nos muestra únicamente las entidades representables a escala 1:5000 que son las curvas de nivel, las cotas y el edificio. Han desaparecido las divisiones de alturas.
10. Ahora nos interesa ocultar las curvas de nivel, por lo tanto ejecutamos la orden OFF=#Curvas, por lo tanto únicamente vemos el contorno del edificio y las cotas.
11. Ahora nos interesa ver todas las líneas de edificación, por lo tanto ejecutaremos la orden OFF=* y ON=#Edificaciones y comprobamos que únicamente se ver el edificio y las líneas de división de alturas.
12. Cerramos Digi3D.

Traducción de códigos incluso a nivel de atributo

Un factor a tener en cuenta es que Digi3D 2011 considera todos los códigos de una entidad como códigos y no como atributos.

Los importadores de archivos de dibujo (los módulos de Digi3D que almacenan las entidades en archivos .bin o .dwg, o .dwg, o...) no almacenan las entidades con el código que ha seleccionado el usuario, sino con el *código de traducción* para ese código.

Ese proceso se realiza de forma transparente al usuario y en un principio estaba pensado para que los empleados de una empresa que trabajase para varios clientes memorizasen únicamente un conjunto de códigos independientemente de la codificación de cada uno de sus clientes, de modo que para registrar un edificio tuvieran que seleccionar siempre el código *Edificio* independientemente de si están trabajando para el cliente A o el B.

Lo que sí que se tendrían son dos tablas de códigos, ClienteA.tab.xml y ClienteB.tab.xml, las dos con exactamente los mismos códigos (entre los cuales estaría el código *Edificio*), pero con configuraciones de transformación a .bin diferentes obviamente, en una por ejemplo se podría indicar que *Edificio* se almacenaba en el archivo .bin con código 500001 y en la otra que el mismo código *Edificio* ésta vez se almacenaba en el archivo .bin con código 021234.

Además esta característica de traducción simultánea de códigos permite que el usuario se olvide de las limitaciones de codificación intrínsecas al formato del archivo de dibujo utilizado. Los archivos .bin por ejemplo tienen una limitación en la longitud de los códigos, que no pueden superar los 6 caracteres, de modo que no se puede almacenar en un archivo .bin el código: CARRETERA_EN_CONSTRUCCION_AÑO_2010. Este código se debe almacenar en un archivo .bin con algún identificador de como máximo 6 caracteres. Pero gracias a esta característica de traducción simultánea, sí que podemos crear una tabla de códigos que tenga el código CARRETERA_EN_CONSTRUCCION_AÑO_2010 (configurado para que se almacene en el archivo .bin con algún identificador de cómo máximo 6 caractéres), pero bajo el punto de vista del operador, el código se llama CARRETERA_EN_CONSTRUCCION_AÑO_2010, ya que Digi3D realiza la traducción de forma transparente al almacenar/cargar entidades en el archivo .bin.

Todo este proceso de traducción del código que ve el usuario al código en el que realmente hay que almacenar en el archivo de dibujo, en versiones anteriores se realizaba únicamente con los códigos principales de las entidades. Los códigos secundarios (aquellos añadidos como atributos) no pasaban por este proceso de traducción.

Digi3D 2011 traduce todos los códigos, de modo que si volvemos a estudiar la tabla de códigos del Laboratorio 1, esta vez comprobando los códigos de transformación para formato .bin:

Código	Descripción	Código en el archivo .bin
0	Marco de hoja	0
Parcela	Límite de parcela	1
Edificio	Edificio privado	2
Piscina	Piscina	3

Podemos comprobar que cuando editamos la línea de edificio y le añadimos el código de Parcela, en realidad no se almacenó una línea con los códigos “Edificio” y “Parcela” sino que se almacenó una línea con códigos “2” y “1”.

Al volver a cargar ese archivo en memoria, Digi3D se encarga de volver a convertir los códigos “2” y “1” en “Edificio” y “Parcela”.

De hecho, el **Laboratorio 1 no se puede imitar con versiones anteriores de Digi3D** ya que si ejecutamos con una versión anterior la orden ALTA, tecleamos en el campo Resto el código *Parcela*, al almacenar la entidad en el archivo .bin, como el campo Resto de los atributos tiene únicamente un espacio para 6 caracteres, el exportador a .bin tiene que truncar el código para que este tenga como máximo 6 caracteres, de modo que la palabra que se almacena no es *Parcela* sino *Parcel*, por lo que al regenerar el modelo Digi3D mostraría la entidad con color desconocido porque desconoce qué código es el código *Parcel* ya que en la tabla de códigos sencillamente no existe.

Vamos a verlo en acción en el siguiente laboratorio...

Laboratorio 3 (Exportando a .asc)

En este laboratorio vamos a exportar un archivo con una entidad con codificación múltiple al formato ASCII clásico de Digi para comprobar que Digi3D traduce los códigos según lo indicado en la tabla de códigos.

Como no se puede modificar el formato de un archivo, ya que se rompería con la compatibilidad de versiones anteriores, Digi3D 2011 se las tiene que *apañar* para almacenar todos los códigos de una entidad con codificación múltiple en un formato que las versiones anteriores puedan entender, y como en versiones anteriores una entidad podía tener un código principal y una serie de códigos secundarios (almacenados como atributos), los exportadores a formato .bin y .asc se van a comportar de esa manera, van a asumir que el primer código de la lista de códigos de una entidad es el principal y el resto se van a almacenar como atributos de la entidad.

1. Ejecuta Digi3D 2011 y carga el archivo Modelo.bin del directorio [Laboratorio de codificación múltiple]\Laboratorio 3.
2. Tentativa la línea dibujada en el centro de la superficie de trabajo para comprobar que es una línea con codificación múltiple (si no tienes activa la *Ventana de Tentativos*), actívala para poder ver todos los códigos de la entidad: *Parcela*, *Edificio*, *Piscina*.
3. Ejecuta la orden *Exportar* y selecciona como formato de archivo de salida *Archivos Ascii de Digi (*.asc)*.
4. Guarda el archivo con el nombre por defecto *Modelo.asc*.
5. Cierra Digi3D 2011.
6. Abre el archivo .asc creado con un bloc de notas y comprueba que los códigos que se han almacenado se han traducido. Si se fijas en la segunda entidad, ésta tiene como código principal el código “1” (Parcela en la tabla de códigos). Dispone de dos atributos, el primero en su campo Tabla tiene un 1, que indica que ese código no está borrado de la entidad (además de indicarnos que no está enlazado con ninguna tabla de base de datos, pues si no, su valor sería mayor que 1), y en el campo Resto se ha almacenado el código “2” (Edificio). El último atributo almacenado es el del código “3” (Piscina).

Laboratorio 4(*Importando de un .asc*)

En este laboratorio vamos a importar un archivo .asc para comprobar que Digi3D 2011 también realiza la traducción en modo inverso.

El archivo Modelo.bin de este laboratorio únicamente tiene el marco que define el límite de trabajo y el archivo Modelo.asc es una modificación del generado en el laboratorio 3 al que se ha eliminado la primera entidad.

1. Ejecuta Digi3D 2011 y carga el archivo Modelo.bin del directorio [Laboratorio de codificación múltiple]\Laboratorio 4.
2. Ejecuta la orden *Importar* y selecciona el archivo *Modelo.asc*.
3. Tentativa la línea importada para comprobar que sus códigos no son 1, 2 y 3 sino *Parcela, Edificio, Piscina*.

Acabas de comprobar el comportamiento del importador/exportador de archivos .ASC y este importador/exportador se comporta exactamente igual que el importador/exportador de archivos .bin.

Codificación múltiple en archivos de dibujo de terceros

Digi3D 2011 es capaz de almacenar entidades con codificación múltiple en formatos de archivo de aplicaciones que no tienen esa funcionalidad.

Si cargamos estos archivos con terceras aplicaciones no podremos trabajar con codificación múltiple, pero si los cargamos con Digi3D 2011 podremos trabajar con éstas como si estuvieran almacenadas en un archivo .bin.

Archivos AutoCAD .dwg

El formato DWG permite que cualquier aplicación se registre en la cabecera del archivo de dibujo. Si se registra una aplicación en un archivo de dibujo, ésta podrá almacenar atributos en las entidades. Estos atributos pueden tener el formato que desee la aplicación con la condición de que se debe almacenar como primer nodo del atributo un nodo de tipo cadena de caracteres con el símbolo llave abierta ({}) y como último atributo uno con una llave cerrada (}). Entre medias se puede almacenar lo que sea y en el formato que sea.

Si abrimos un archivo .DWG con Digi3D, una de las primeras cosas que hace el importador/exportador de DWG es registrar la aplicación “DIGING”.

Al almacenar entidades, si estas tienen codificación múltiple, se almacenará en el archivo .DWG junto con la entidad tantos atributos como códigos tenga la entidad.

El formato de estos atributos es el siguiente:

Tipo de nodo DWG	Valor almacenado
ControlString	{
Integer16	1
Integer32	0
AsciiString	Código
ControlString	}

Vamos a verlo en acción:

Laboratorio 5 (Exportando a .dxf)

En este laboratorio vamos a transformar un modelo con una entidad con codificación múltiple a formato .DXF.

El formato .DXF refleja fielmente la información almacenada en un archivo .DWG, y como este es ASCII, vamos a poder ver cómo se registra Digi3D 2011 y cómo se almacenan los atributos.

1. Haz doble clic sobre la tabla de códigos Labotatorio.tab.xml en el directorio [Laboratorio de codificación múltiple]\Laboratorio 5 para abrirlo con *Internet Explorer* o con el bloc de notas.
2. Desplázate a la sección de códigos y comprueba que ahora la tabla de códigos tiene parámetros de traducción a DWG a parte de los parámetros de traducción a .bin que ya tenía en laboratorios anteriores. Si te fijas, ahora los nombres de las capas a las que traducir coinciden con los nombres de los códigos en Digi, ya que AutoCAD no tiene limitación en el tamaño de los nombres de las capas.
3. Ejecuta Digi3D 2011 y carga el archivo Modelo.bin del directorio [Laboratorio de codificación múltiple]\Laboratorio 5.
4. Tentativa la línea dibujada en el centro de la superficie de trabajo para comprobar que es una línea con codificación múltiple (si no tienes activa la *Ventana de Tentativos*), actívala para poder ver todos los códigos de la entidad: *Parcela, Edificio, Piscina*.
5. Ejecuta la orden *Exportar* y selecciona *Archivos AutoCAD DXF (*.dxf)* y por último almacena el archivo con el nombre propuesto, *Modelo.dxf*
6. Cierra Digi3D.
7. Abre el archivo *Modelo.dxf* con el bloc de notas o con tu editor favorito. Ahora vete a la línea 1370 (puedes activar la visualización de línea/columna en el bloc de notas en el menú Ver/Barra de estado).
8. Los archivos DWG y DXF son en realidad bases de datos con muchas tablas. Hay una tabla llamada *RegAppTable* que es la tabla donde se almacenan las aplicaciones registradas.

- La líneas 1370 a 1380 representan una entrada en la tabla de aplicaciones del archivo .dxf en la que se está registrando la aplicación DigiING.
9. Vete ahora a la línea 1790. Aquí comienza la línea con codificación múltiple. En la línea 1802 se almacena la capa de la entidad. Como AutoCAD no sabe de múltiples capas, la entidad se tiene que almacenar con una única capa. El exportador de DWG de Digi ha seleccionado el primer código de la entidad como capa de ésta.
 10. Ahora vete a la línea 1815, ahí comienza el primer atributo para la aplicación DIGING. Podrás comprobar que la entidad tiene tres atributos para la aplicación DIGING y todos ellos están entre llaves. En las líneas 1824, 1834 y 1844 tienes los tres códigos que tenía la entidad en Digi3D.
 11. Cierra el bloc de notas

Laboratorio 6 (Importando de .dxf)

En este laboratorio vamos a importar el archivo .dxf creado en el laboratorio anterior para comprobar que Digi3D 2011 importa correctamente las entidades con múltiple codificación.

1. Desplázate al directorio [Laboratorio de codificación múltiple]\Laboratorio 6 y comprueba si existe el archivo Modelo.bin. Si existe, elimínalo.
2. Ejecuta Digi3D 2011 y crea archivo Modelo.bin en el directorio [Laboratorio de codificación múltiple]\Laboratorio 6.
3. Ejecuta la orden Importar e importa el archivo Modelo.dxf del mismo directorio. Termina haciendo un zoom extendido para ver el modelo importado.
4. Tentativa la línea con codificación múltiple y comprueba que esta tiene los códigos *Parcela, Edificio y Piscina*.
5. Cierra Digi3D.
6. Abre el archivo Modelo.dxf con el bloc de notas o con tu editor favorito. Ahora vete a la línea 1370 (puedes activar la visualización de línea/columna en el bloc de notas en el menú Ver/Barra de estado).

Archivos MicroStation DGNv8

Las cosas en DGN no son tan sencillas como en BIN o en DWG. El importador de archivos DGNv8 tiene más complicado decidir el código de una entidad por un problema heredado de los archivos DGNv7.

En DGNv7 el nivel de una entidad se definía como un número entre el 0 y el 64.

64 niveles era un número muy pequeño, por lo que no se podría hacer una clasificación de todos los elementos representables en una cartografía basándose únicamente en el número

de nivel. Cualquiera que haya visto una tabla de códigos Digi.tab.xml puede comprobar que el número de códigos para cualquier escala supera claramente este valor.

En DGNv7 se solucionaba el problema agrupando entidades similares en un mismo nivel, utilizando como elemento diferenciador otro atributo como el color de la entidad, su estilo o su grosor, de modo que se podían agrupar todos los tipos de *camino* en una capa, digamos la 10, y para diferenciar entre senda, camino en construcción, camino carretero,... se utilizaba el color, cada una de ellas con un estilo lineal adecuado y con su grosor correspondiente.

Otra manera de clasificar entidades era mediante un atributo denominado Grupo Gráfico que se podía asignar a cada entidad.

Otra manera consistía en seguir el estándar de Geographics (estándar pensado para codificación múltiple), que básicamente consistía en que los códigos de la entidad se almacenan en el campo mslink de los enlaces a base de datos de la entidad.

Y para terminar, llegó DGNv8³ en el que los niveles han pasado a ser alfanuméricos, y por lo tanto ya sí que se puede utilizar únicamente el nivel como criterio diferenciador.

Así que tenemos muchas formas de decidir el código que tendrá una entidad cargada de un archivo DGN de modo que debería de haber alguna forma de indicarle al importador el criterio que queremos que siga para decidir qué códigos tendrán las entidades que vemos en pantalla.

Y eso es lo que nos pregunta precisamente el importador de DGNv8 cuando creamos o cargamos un archivo de este tipo en la sección Parámetros del importador/exportador del cuadro de diálogo Nuevo Proyecto (pestaña Archivo de dibujo) o en los cuadros de diálogo comunes de Abrir archivo o Guardar Como: El criterio de codificación.

Si desplegamos la lista, podemos seleccionar una de las siguientes opciones que explico en la siguiente tabla:

Valor	Descripción	Multi codificación
Nivel	Al cargar una entidad, se buscará el código de la tabla de códigos que tiene configurado ese nivel en sus parámetros de traducción de DGN. Se seleccionará el primer código de la tabla de códigos que cumpla esta condición.	No
Nivel/Célula	Igual que el caso anterior para todos los tipos de entidad excepto	No

³ Los párrafos anteriores se refieren a DGNv7, pero son totalmente aplicables a DGNv8, ya que las empresas cuando migraron de DGNv7 a DGNv8 no cambiaron sus criterios de clasificación, así que siguieron trabajando y siguen trabajando de esa manera.

	para las células.	
	Si se localiza una célula, el código con el que la veremos en Digi3D será el del primer código en la tabla que tenga en sus parámetros de traducción a DGN esa célula en el campo <i>célula</i> .	
Nivel,Color,Estilo,Grosor	Al cargar una entidad, se buscará el código en la tabla de códigos que tenga almacenada la combinación de nivel, color, estilo y grosor de la entidad que se está cargando.	No
Nivel,Color,Estilo,Grosor/Célula	Igual que el caso anterior para todos los tipos de entidad excepto para las células. Si se localiza una célula, el código con el que la veremos en Digi3D será el del primer código en la tabla que tenga en sus parámetros de traducción a DGN esa célula en el campo <i>célula</i> .	No
Grupo Gráfico	Al cargar una entidad, se buscará el código de la tabla de códigos que tiene configurado ese grupo gráfico en sus parámetros de traducción de DGN. Se seleccionará el primer código de la tabla de códigos que cumpla esta condición.	No
Geographics	Al cargar una entidad, se localizarán sus enlaces de base de datos y se extraerá de dichos enlaces el conjunto de códigos de la entidad. Luego se buscará el código de la tabla de códigos que tiene configurado ese nivel en sus parámetros de traducción de DGN.	Si

Como puedes comprobar en la tabla anterior, el único criterio que nos va a permitir trabajar con codificación múltiple es *Geographics*, de modo que si seleccionamos cualquiera de los otros criterios y cargamos un archivo DGN con entidades con enlaces de base de datos tipo *Geographics*, no se interpretarán como códigos.

En la siguiente tabla puedes ver el comportamiento del importador/exportador según el criterio seleccionado al cargar/almacenar entidades con o codificación múltiple:

Criterio	Cargar entidad sin enlace a base de datos	Cargar entidad con enlaces a base de datos	Almacenar entidad con un único código	Almacenar entidad con múltiples códigos
Nivel Nivel/Célula Nivel, color, estilo, grosor Nivel,color,estilo,grosor/Célula Grupo gráfico	Carga la entidad asignándole el código en función del criterio.	Carga la entidad asignándole el código en función del criterio pero hará caso omiso de los enlaces de base de datos.	Almacena la entidad con el nivel, color, estilo, grosor y grupo indicados en los parámetros de traducción a DGN en la tabla de códigos para el código de la entidad.	Almacena la entidad en el nivel indicado en el campo Nivel de la tabla de códigos para el primer código de la entidad.
Geographics	Muestra un cuadro de diálogo indicando que no se puede cargar el archivo de dibujo pues no cumple el estándar de Geographics.	Carga la entidad asignándole como códigos los de los enlaces a base de datos de la entidad, localizando en la tabla de códigos a qué códigos debe convertir los mslinks de esos enlaces.	Almacena la entidad en el nivel indicado en los parámetros de traducción a DGN en la tabla de códigos para el primer código de la entidad y almacena un enlace a base de datos de tipo ODBC, en la tabla 2 y con mmlink=Nivel	Almacena la entidad en el nivel indicado en los parámetros de traducción a DGN en la tabla de códigos para el primer código de la entidad y almacena un enlace a base de datos de tipo ODBC, en la tabla 2 y con mmlink=Nivel localizado en

	localizado en la tabla de códigos por cada código de la entidad.	la tabla de códigos por cada código de la entidad.
--	--	--

Si quieres profundizar un poco más en el tema, a continuación te explico cómo son los enlaces a base de datos en un archivo DGN, si no te interesa, salta al laboratorio 7.

Un enlace a base de datos no es más que una tripla de valores que se adjunta a una determinada entidad en el archivo DGN. Se pueden adjuntar tantos enlaces como quieras y estos tienen este aspecto:

Tipo	Tabla	Mslink
BSI, FRAMME, Informix, Ingres, ODBC, OLEDB, Oracle, RIS, Sybase y Xbase	Número de tabla si hay enlace a base de datos.	Valor numérico donde del atributo.

El estándar de Geographics define que los códigos de una entidad se almacenarán como enlaces de base de datos de tipo ODBC, con el valor de tabla = 2 y con el campo mslink igual al código de la entidad.

Veamos un ejemplo: Si tenemos la línea de los laboratorios anteriores con códigos *Parcela*, *Edificio* y *Piscina* (suponiendo que en la tabla de códigos la Parcela se almacena en el nivel 1, el edificio en el nivel 2 y la piscina en el nivel 3) al almacenar esa entidad en un archivo DGN siguiendo el criterio de Geographics nos encontraríamos con algo parecido a esto:

Tipo de entidad	Línea		
Nivel	1		
Color	[el especificado en la sección DGN para el código Parcela en la tabla de códigos]		
Estilo	[el especificado en la sección DGN para el código Parcela en la tabla de códigos]		
Grosor	[el especificado en la sección DGN para el código Parcela en la tabla de códigos]		
Vértices	(conjunto de coordenadas de los vértices de la línea)		
Enlace de base de datos	Tipo	Tabla	Mslink
	ODBC	2	1
Enlace de base de datos	Tpo	Tabla	Mslink
	ODBC	2	2
Enlace de base de datos	Tipo	Tabla	Mslink
	ODBC	2	3

Laboratorio 7 (Exportando a .dgn)

En este laboratorio vamos a crear un archivo .dgn indicando como criterio Nivel, luego vamos a almacenar una entidad con tres códigos para comprobar que el exportador nos va a avisar de que no se han podido almacenar todos los códigos de la entidad porque el modo en el que se ha abierto el archivo de dibujo no admite codificación múltiple.

A continuación cargaremos el archivo con un *dumper* de Dgn denominado OdaDgnApp que puedes localizar en el directorio del laboratorio para comprobar que la entidad se ha almacenado sin enlaces a base de datos.

1. Abre un explorador de archivos y localiza la carpeta [Laboratorio de codificación múltiple]\Laboratorio 7.
2. Si existe en esa carpeta algún archivo con extensión .dgn, elimínalo.
3. Haz doble clic sobre el archivo *Laboratorio.tab.xml* ubicada para comprobar que tiene cuatro códigos y todos ellos tienen asignada la etiqueta “Todos”.
4. Cierra *Internet Explorer* o el bloc de notas.
5. Ejecuta Digi3D 2011 y crea el archivo *Modelo.dgn* en el directorio directorio [Laboratorio de codificación múltiple]\Laboratorio 7 seleccionando en *Criterio para códigos*: Nivel.
6. Ejecuta la orden COD=#Todos para seleccionar como códigos activos todos los códigos de la tabla de códigos.
7. Dibuja una línea y confírmala.
8. Comprueba que el exportador de DGN nos muestra una ventana indicando que no se van a almacenar todos los códigos de esa línea porque no estamos trabajando con un DGN abierto en modo de codificación múltiple.
9. Abre la *Ventana de tareas* si la tienes cerrada, activa la visualización de errores y comprueba que Digi3D 2011 ha añadido una tarea indicando el error.
10. Haz doble clic sobre la tarea para comprobar que Digi3D 2011 desplaza la vista para centrar la línea en pantalla.
11. Dibuja una segunda línea comprobar que el exportador ya no vuelve a mostrar el cuadro de diálogo pero sin embargo sí se añade una nueva tarea en la *Ventana de tareas*.
12. Cierra el modelo y vuelve a cargarlo.
13. Tentativa las líneas y comprueba que únicamente tienen un código, el código 0 que era el primero de la lista de códigos cuando almacenamos las entidades.
14. Cierra Digi3D 2011.
15. Ejecuta la aplicación OdaDgnApp que nos va a permitir visualizar el archivo DGN por dentro.
16. Pulsa Ctrl+O para indicarle al programa que queremos abrir un archivo de dibujo.
17. Localiza y abre el archivo *Modelo.dgn* que acabas de crear en la carpeta Laboratorio 7.
18. El programa OdaDgnApp nos muestra una ventana con un árbol en su parte izquierda y una ventana donde se muestran valores en la parte derecha.

19. Abre la rama titulada OdDgModelTable, luego abre su rama hija, Model y comprueba que tiene dos hijos de tipo OdDgLineString. Esas son las dos líneas que has registrado en el archivo de dibujo.
20. Haz clic en la primera y comprueba que en la parte derecha de la ventana en la penúltima línea nos está indicando que la entidad tiene 0 enlaces a base de datos (Linkages Count = 0).
21. Cierra el programa.

Laboratorio 8(Analizando un archivo .dgn)

En este laboratorio crearemos un archivo DGN con entidades con múltiples códigos y vamos a utilizar el programa OdDgnApp para comprobar que el exportador de DGN añade a las entidades tantos enlaces a base de datos como códigos tiene la entidad.

1. Abre un explorador de archivos y localiza la carpeta [Laboratorio de codificación múltiple]\Laboratorio 8.
2. Si existe en esa carpeta algún archivo con extensión .dgn, elimínalo.
3. Ejecuta Digi3D 2011 y crea el archivo Modelo.dgn en el directorio [Laboratorio de codificación múltiple]\Laboratorio 8 seleccionando en *Criterio para códigos:* Geographics.
4. Ejecuta la orden COD=#Todos para seleccionar como códigos activos todos los códigos de la tabla de códigos.
5. Dibuja una línea y confírmala.
6. Cierra Digi3D 2011.
7. Ejecuta la aplicación OdaDgnApp que nos va a permitir visualizar el archivo DGN por dentro.
8. Pulsa Ctrl+O para indicarle al programa que queremos abrir un archivo de dibujo.
9. Localiza y abre el archivo Modelo.dgn que acabas de crear en la carpeta Laboratorio 8.
10. El programa OdaDgnApp nos muestra una ventana con un árbol en su parte izquierda y una ventana donde se muestran valores en la parte derecha.
11. Abre la rama titulada OdDgModelTable, luego abre su rama hija, Model y comprueba que tiene un nodo hijo de tipo OdDgLineString.
12. Haz clic en ese nodo y comprueba que en la parte derecha de la ventana el campo *Linkages count* tiene asignado el valor 4 (que corresponden con los cuatro códigos de la entidad) y a continuación aparecen los cuatro enlaces a base de datos de tipo OdDgDBLinkage.
13. Cierra el programa.

En este capítulo has descubierto las nuevas facilidades disponibles con Digi3D 2011 para permitirnos crear modelos con geometrías con múltiples códigos.

Ha cambiado el interfaz de usuario de Digi3D porque ha desaparecido la barra clásica de códigos y ha aparecido la ventana de códigos, se han modificado las órdenes de selección de códigos para que admitan muchos códigos, se han creado órdenes nuevas relacionadas con códigos, se han añadido etiquetas a las tablas de códigos y has podido comprobar lo que sucede *detrás del telón* en los distintos formatos de archivo de dibujo cuando almacenamos entidades con múltiples códigos.

Pero no hemos terminado aún con los códigos, ya que estos están muy relacionados con los enlaces de base de datos. En el capítulo Acceso a Base de datos básico aprenderás cómo enlazar geometrías en Digi3D 2011 con bases de datos y comprobarás cómo se almacena la información de enlace a bases de datos en los distintos formatos de archivo de dibujo que soportan enlaces a bases de datos.

2. Codificación múltiple mediante programación .NET

Si no tienes conocimientos de programación .NET, salta al siguiente capítulo.

El entorno de desarrollo de Digi3D 2011 dispone de varios tipos y métodos que permiten seleccionar códigos activos, consultar los códigos activos, localizar los códigos que tengan asociada una determinada etiqueta, consultar las etiquetas de un código...

En este capítulo nos familiarizaremos con los tipos *Code*, *DigiTab* y con varios métodos de la instancia *Digi*.

Realizaremos alguna aplicación de consola y alguna orden para ejecutar dentro de DigiNG.

Es necesario que tengas instalado Microsoft Visual C# 2010 Express para poder realizar los ejercicios de este capítulo.

El tipo valor *Code*

Los códigos se representan mediante el tipo valor *Digi21.DigiNG.Entities.Code* ubicado en el ensamblado *Digi21.DigiNG*.

El tipo *Code* dispone de dos constructores, uno que admite únicamente el nombre del código y otro que nos permite indicar la *Tabla* e *Id* asociados con el código.

Además define tres propiedades *Name*, *Table* e *Id* (todas ellas de solo lectura) para recuperar el nombre del código, la tabla asignada al código y el identificador del código.

Las propiedades *Table* e *Id* se utilizan para el enlace a base de datos y se explicarán en el capítulo de Acceso a base de datos básico.

El tipo *Code* sobrecarga el método *Object.ToString()* que devuelve una cadena de caracteres con el estado del código.

El tipo *Code* sobrecarga el operador igualdad para comprobar si dos códigos son idénticos.

Laboratorio 1(Familiarizándonos con el tipo Code)

En este laboratorio vamos a hacer una aplicación de consola que creará códigos y realizará operaciones con éstos.

1. Ejecutamos Visual C# 2010 Express
2. En el menú Archivo seleccionamos *Nuevo Proyecto*.
3. En *Plantillas instaladas* nos aseguramos de que está seleccionado *Visual C#*.
4. Seleccionamos *Aplicación de Consola*.
5. En *Nombre* tecleamos *Códigos*.
6. Pulsamos el botón *Aceptar*.
7. En el *Explorador de soluciones* cambiamos el nombre del archivo *Program.cs* (pulsando con el botón derecho del ratón y *Cambiar nombre* en el menú contextual) por *Códigos.cs*.
8. Aceptamos el cuadro de diálogo que nos pregunta si queremos renombrar también la clase.
9. Añadimos una referencia al ensamblado *Digi21.DigiNG* (pulsando con el botón derecho la rama *References* del proyecto en la ventana *Explorador de soluciones* y seleccionando *Agregar referencia* en el menú contextual).
10. El ensamblado *Digi21.DigiNG* se puede localizar en la pestaña *.NET* del cuadro de diálogo *Agregar Referencia*.
11. Aceptamos el cuadro de diálogo *Agregar Referencia*. Declaramos una variable *finas* de la siguiente manera dentro del cuerpo del método *Main*:

```
Code finas = new Code("020200");
```

12. Comprobamos que Visual C# 2010 Express marca el nombre del tipo con una línea roja indicando que no reconoce el tipo *Code* (porque está ubicado en un espacio de nombres al que no se tiene acceso desde nuestro código).

Hacemos clic con el botón derecho del ratón sobre el nombre del tipo y seleccionamos *Resolver/using Digi21.DigiNG.Entities* en el menú contextual para comprobar que el compilador ha añadido al comienzo del programa la declaración:

```
using Digi21.DigiNG.Entities;
```

Y que ahora la declaración de la variable *finas* aparece en color verde (que indica que el tipo es conocido):

```
Code finas = new Code("020200");
```

13. Declaramos el código *maestras* y *códigoDibujo*

```
Code maestras = new Code("020400");
Code códigoDibujo = new Code("020400");
```

14. Imprimimos el nombre asignado a la variable *finas*.

```
Console.WriteLine(
    string.Format(
        "El código de finas tiene el nombre: {0}",
        finas.Name));
```

15. Compilamos (pulsando la tecla F6), corregimos los errores y ejecutamos el programa (pulsando la combinación de teclas Ctrl+F5).

16. La propiedad *Name* es de solo lectura. Añadimos la siguiente línea y compilamos para comprobar que el compilador nos indica que la propiedad es de solo lectura:

```
códigoDibujo.Name = "112233";
```

17. Comentamos la línea anterior.

18. Podemos utilizar el operador igualdad para comprobar si dos códigos son idénticos:

```
if (códigoDibujo == finas)
    Console.WriteLine("El código de dibujo es el de curvas de nivel
finas.");
else if( códigoDibujo == maestras )
    Console.WriteLine("El código de dibujo es el de curvas de nivel
maestras.");
else
    Console.WriteLine("El código de dibujo no es ni el de curvas de
nivel finas ni maestras.");
```

19. Compilamos y ejecutamos para comprobar que el programa nos indica que el código de dibujo es el de curvas de nivel *maestras*.

20. Muchos de los métodos de Digi3D reciben como parámetros una enumeración de códigos.

Vamos a crear ahora un método que recibe como parámetro un `IEnumerable<Code>` e imprime el contenido de la enumeración.

```
static void ImprimeNombresCódigos(IEnumerable<Code> códigos)
{
    Console.WriteLine("Imprimiendo un listado de códigos");
    foreach (Code código in códigos)
        Console.WriteLine(código.Name);
}
```

21. Y a continuación vamos a ver varias maneras de crear un enumerador.

Como sabrás, `IEnumerable` e `IEnumerable<T>` son dos interfaces que implementan **todos** los contenedores de .NET, (listas, arrays, diccionarios,...) así que aquí vamos a declarar una variable de tipo array que denominaremos *altimetria1* para añadir los códigos de fina y maestra.

```
Code[] altimetria1 = new Code[2];
altimetria1[0] = finas;
altimetria1[1] = maestras;
```

22. Esta sería la forma más clásica de llenar el array, pero C# es un lenguaje inteligente, y nos permite simplificar la creación/asignación del array en una única línea:

```
Code[] altimetria2 = new Code[] {
    new Code("020200"),
    new Code("020400") };
```

23. Y es más inteligente aún porque no es necesario que declaremos el tipo `Code[]` a la derecha de la asignación:

```
Code[] altimetria3 = new[] {
    new Code("020200"),
    new Code("020400") };
```

24. E incluso nos permite simplificar aún más ya que realmente no es necesario el `new[]` en la parte derecha de la asignación:

```
Code[] altimetria4 = { new Code("020200"), new Code("020400") };
```

Bien, pues esta forma de declarar e instanciar los arrays es la que se seguirá en el resto del libro.

25. A continuación vamos a imprimir los cuatro arrays que acabamos de declarar además de una llamada al método con una declaración del array en línea:

```
ImprimeNombresCódigos(altimetria1);
ImprimeNombresCódigos(altimetria2);
ImprimeNombresCódigos(altimetria3);
ImprimeNombresCódigos(altimetria4);
ImprimeNombresCódigos(new[] { new Code("020200"), new Code("020400") });
```

26. Compilamos, corregimos fallos y ejecutamos para ver que el listado de códigos se imprime cinco veces.

27. Por último, muchas veces será necesario comprobar si una determinada enumeración de códigos contiene un determinado código (por ejemplo comprobar si una entidad tiene el código 020400). Para ello podemos utilizar el método de extensión *Contains* definido en el espacio de nombres *System.Linq*.

```
Code[] códigosDeEntidad = altimetria4;
if( códigosDeEntidad.Contains(new Code("020400")))
    Console.WriteLine("La entidad tiene el código 020400 en su lista
de códigos");
```

28. Compilamos, corregimos fallos y ejecutamos para comprobar que el programa indica que la entidad tiene el código 020400 entre sus códigos.

El tipo referencia *DigiTab*

Sabemos que podemos asociar etiquetas a códigos y esta asociación se realiza en la tabla de códigos *digi.tab.xml*.

Podemos cargar una tabla de códigos en .NET mediante el tipo referencia *DigiTab* en el espacio de nombres *Digi21.DigiNG.DigiTab* en el ensamblado *Digi21.DigiNG*.

Este tipo no dispone de constructor. Para construir un objeto de tipo *DigiTab* tenemos que ejecutar el método estático *DigiTab.Load(rutaArchivo)* que devuelve una instancia de *DigiTab*. Este método lanzará excepciones de tipo *System.IO.FileNotFoundException* si no se localiza el archivo pasado por parámetros o de tipo *System.Expcion* si el archivo está mal formado o no es un archivo *digi.tab.xml* válido.

El tipo *DigiTab* dispone de tres métodos relacionados con etiquetas:

1. *DigiTab.TagsOfCode(código)* que devuelve un enumerador con las etiquetas asociadas a un código.
2. *DigiTab.CodesWithTag(etiqueta)* que devuelve un enumerador con todos los códigos que tienen asociada la etiqueta pasada por parámetros.
3. *DigiTab.CodeHasTag(código, etiqueta)* que devuelve verdadero si el código tiene asociada la etiqueta.

Laboratorio 2(Familiarizándonos con el tipo DigiTab)

En este laboratorio vamos a cargar una tabla de códigos y comprobaremos y nos familiarizaremos con los métodos relacionados con etiquetas del tipo *DigiTab*.

1. Ejecutamos Visual C# 2010 Express
2. En el menú *Archivo* seleccionamos *Nuevo Proyecto*.
3. En *Plantillas instaladas* nos aseguramos de que está seleccionado *Visual C#*.
4. Seleccionamos *Aplicación de Consola*.
5. En *Nombre* tecleamos *TablasDeCódigos*.
6. Pulsamos el botón *Aceptar*.
7. En el *Explorador de soluciones* cambiamos el nombre del archivo *Program.cs* (pulsando con el botón derecho del ratón y *Cambiar nombre* en el menú contextual) por *TablasDeCódigos.cs*.
8. Aceptamos el cuadro de diálogo que nos pregunta si queremos renombrar también la clase.
9. Añadimos una referencia al ensamblado *Digi21.DigiNG* (pulsando con el botón derecho la rama *References* del proyecto en la ventana *Explorador de soluciones* y seleccionando *Agregar referencia* en el menú contextual).
10. El ensamblado *Digi21.DigiNG* se puede localizar en la pestaña *.NET* del cuadro de diálogo *Agregar Referencia*.
11. Aceptamos el cuadro de diálogo *Agregar Referencia*.

12. Añadimos los siguientes espacios de nombres al comienzo del programa:

```
using Digi21.DigiNG.DigiTab;
using Digi21.DigiNG.Entities;
using System.IO;
```

13. Declaramos dentro del cuerpo del método *Main* una variable para la tabla de códigos

```
DigiTab digitab;
```

14. Ahora vamos a cargar la tabla de códigos con el método *DigiTab.Load*, pero provocando fallos. Primero intentaremos cargar un archivo inexistente:

```
try
{
    Console.WriteLine("Probando a cargar el archivo pppppp.xml");
    digiTab = DigiTab.Load(@"z:\pppppp.xml");
}
catch (FileNotFoundException excepción)
{
    Console.WriteLine(excepción.Message);
}
Console.WriteLine();
```

15. Luego intentaremos cargar un archivo existente pero mal formado. Sustituye la ruta por la del laboratorio para que apunte al archivo *incorrecto.tab.xml*

```
try
{
    Console.WriteLine("Probando a cargar el archivo
incorrecto.tab.xml");
    digiTab = DigiTab.Load(@"[pon_aqui_la_ruta]\incorrecto.tab.xml");
}
catch (Exception excepción)
{
    Console.WriteLine(excepción.Message);
}
Console.WriteLine();
```

16. Y por último cargaremos el archivo correcto:

```
try
{
    Console.WriteLine(@"Probando a cargar el archivo
Laboratorio.tab.xml");
    digiTab =
DigiTab.Load(@"[pon_aqui_la_ruta]\Laboratorio.tab.xml");
    Console.WriteLine("Archivo cargado correctamente");
}
catch (Exception excepción)
{
    Console.WriteLine(excepción.Message);
    return;
}
Console.WriteLine();
```

17. Compilamos pulsando *F6* corregimos los fallos y ejecutamos pulsando *Ctrl+F5* para comprobar que en el caso de las dos primeras cargas, la excepción está informando exactamente del error y comprobar que a la tercera hemos conseguido cargar el archivo correctamente.
18. Continuamos realizando consultas relacionadas con etiquetas a la tabla de códigos:

```
// Imprimimos las etiquetas del código 020400
Console.WriteLine("Etiquetas del código 020400:");
foreach (string etiqueta in digiTab.TagsOfCode(new Code("020400")))
    Console.WriteLine(etiqueta);

// El método DigiTab.TagsOfCode devuelve un enumerador vacío si el
// código no existe en la tabla de código.
Console.WriteLine("Etiquetas del código ABCDEFG:");
foreach (string etiqueta in digiTab.TagsOfCode(new Code("ABCDEFG")))
    Console.WriteLine(etiqueta);

Console.WriteLine();

// Imprimimos el nombre de los códigos que tienen asociada la
etiqueta 1000
Console.WriteLine("Códigos con la etiqueta 1000");
foreach (Code código in digiTab.CodesWithTag("1000"))
    Console.WriteLine(código.Name);

// El método DigiTab.CodesWithTag devuelve un enumerador vacío si no
localiza ningún código con la etiqueta buscada
Console.WriteLine("Códigos con la etiqueta QWERTY");
foreach (Code código in digiTab.CodesWithTag("QWERTY"))
    Console.WriteLine(código.Name);

Console.WriteLine();

// Comprobamos si el código 020400 tiene asociada la etiqueta 1000
if (digiTab.CodeHasTag(new Code("020400"), "1000"))
    Console.WriteLine("El código 020400 tiene asociada la etiqueta
1000");

// Comprobamos que el código 020400 no tiene asociada la etiqueta
"Planimetría"
if( !digiTab.CodeHasTag(new Code("020400"), "Planimetría") )
    Console.WriteLine("El código 020400 no tiene asociada la etiqueta
Planimetría");
```

Ventana de códigos activos mediante programación

La ventana de códigos activos se puede controlar mediante la propiedad estática *Digi.ActiveCodes*.

Esta propiedad devuelve una enumeración con los códigos activos en la ventana de códigos activos y permite cambiar el contenido de la ventana al asignar un listado de códigos.

Laboratorio 3 (Interactuando con la ventana de códigos activos)

En este laboratorio vamos a crear un clon de la orden COD que denominaremos MICOD. Para ello crearemos una biblioteca de clases y añadiremos una orden DigiING. Luego ejecutaremos Digi3D y ejecutaremos nuestra orden para comprobar que nos permite cambiar de códigos.

1. Ejecutamos Visual C# 2010 Express
2. En el menú *Archivo* seleccionamos *Nuevo Proyecto*.
3. En *Plantillas instaladas* nos aseguramos de que está seleccionado *Visual C#*.
4. Seleccionamos *Biblioteca de clases*.
5. En *Nombre* tecleamos *LaboratorioCod[tu_nombre]*. Yo lo he llamado *LaboratorioCodJoséÁngel*
6. Pulsamos el botón *Aceptar*.
7. En el *Explorador de soluciones* cambiamos el nombre del archivo *Program.cs* (pulsando con el botón derecho del ratón y *Cambiar nombre* en el menú contextual) por *Cod.cs*.
8. Aceptamos el cuadro de diálogo que nos pregunta si queremos renombrar también la clase.
9. Añadimos una referencia a los ensamblados *Digi21.DigiNG* y *Digi21.DigiNG.Plugin* (pulsando con el botón derecho la rama *References* del proyecto en la ventana *Explorador de soluciones* y seleccionando *Agregar referencia* en el menú contextual).
10. Estos ensamblados se pueden localizar en la pestaña *.NET* del cuadro de diálogo *Agregar Referencia*.
11. Aceptamos el cuadro de diálogo *Agregar Referencia*.
12. Añadimos los siguientes espacios de nombres al comienzo del programa:

```
using Digi21.DigiNG.Plugin;
using Digi21.DigiNG.Entities;
```

13. Si queremos que Digi3D reconozca una clase como una orden, esta debe estar decorada con el atributo *CommandAttribute* y debe heredar de la clase *Command* ambos en el espacio de nombres *Digi21.DigiNG.Plugin* (de ahí que hayamos añadido este espacio de nombres al comienzo de nuestro programa). El tipo *CommandAttribute* tiene una única propiedad *Name* en la que indicamos el nombre que tendrá que teclear el usuario para ejecutar la orden. Vamos a llamarla *[tu_nombre].cod* de modo que la mía se llamará *JoséÁngel.cod*. De esta manera si en el mismo equipo alguien hace otra orden con el mismo nombre, no tendremos

colisiones (a menos que ese alguien se llame José Ángel claro está).

Para decorar una clase con un atributo, tenemos que poner el nombre del atributo entre corchetes justo antes del nombre de la clase. Además en la declaración del atributo podemos especificar entre paréntesis los valores de sus atributos.

Mi clase tiene este aspecto:

```
[CommandAttribute(Name = "JoséÁngel.cod")]
public class Cod
{}
```

14. Pero C# permite eliminar el postfijo Attribute de los atributos, de hecho nadie los pone (así lo dictan las reglas de estilo de C#), así que voy a quitárselo. Ahora la clase tiene este aspecto:

```
[Command(Name = "JoséÁngel.cod")]
public class Cod
{}
```

15. Hemos que dado que para que Digi3D 2011 reconozca una clase como una orden, es necesario que esta herede de la clase *Command*, así que únicamente nos queda hacer que nuestra clase herede de esta clase:

```
[Command(Name = "JoséÁngel.cod")]
    public class Cod : Command
{}
```

16. Enhorabuena acabas de crear una orden en Digi3D 2011.
17. La clase base *Command* define un evento denominado *Initialize* que es el evento que se lanza cuando la orden puede comenzar a interactuar con Digi3D 2011. No podemos llamar a ningún método relacionado con Digi3D 2011 hasta que no se ejecuta este evento, así que lo primero que haremos **siempre** al programar una orden en .NET será añadir un manejador de eventos a ese evento en el constructor de la clase.

```
public Cod()
{
    this.Initialize += new Action(Cod_Initialize);
}
```

18. Al escribir el operador `+=`, Visual C# 2010 Express me ha propuesto un nombre por defecto. He pulsado dos veces el tabulador y él solo ha continuado la línea (añadiendo por mí `new Action(Cod_Initialize);` y además ha añadido el siguiente método:

```
void Cod_Initialize()
{
    throw new NotImplementedException();
```

}

19. Digi3D 2011 lanzará el evento *Initialize* de nuestra orden cuando haya inicializado todo lo necesario para que esta se ejecute con seguridad.
20. La clase base *Command* dispone de una propiedad *Args* que es un array de cadenas de caracteres con los parámetros pasados por el usuario, de modo que si el usuario ejecuta la orden JoséÁngel.cod=020200 020400 Hola “¿cómo estás?” la propiedad *Args* tendrá cuatro valores.
21. Vamos a analizar la propiedad *Args*, y en caso de que el usuario haya pasado parámetros a la orden, los seleccionaremos como códigos activos en la *Ventana de códigos activos* de Digi3D 2011 mediante la propiedad *Digi.ActiveCodes*

```
if (this.Args.Length != 0)
{
    List<Code> códigos = new List<Code>();
    foreach (string parámetro in this.Args)
        códigos.Add(new Code(parámetro));
    Digi.ActiveCodes = códigos;
}
```

22. Por último destruiremos la orden llamando al método *Dispose*. La clase debe quedar así:

```
namespace LaboratorioCodJoséÁngel
{
    [Command(Name = "JoséÁngel.cod")]
    public class Cod : Command
    {
        Cod()
        {
            this.Initialize += new Action(Cod_Initialize);
        }

        void Cod_Initialize()
        {
            if (this.Args.Length != 0)
            {
                List<Code> códigos = new List<Code>();
                foreach (string parámetro in this.Args)
                    códigos.Add(new Code(parámetro));
                Digi.ActiveCodes = códigos;
            }
            Dispose();
        }
    }
}
```

23. Como sabrás, si llamas al método *Dispose* de cualquier objeto .NET, éste se destruye, por lo tanto esta debe ser a última operación que se realice con el objeto. La llamada a

Dispose() desapila la orden de la pila de órdenes de DigiNG y además destruye la orden.

Visualizando/Ocultando códigos

Podemos controlar los códigos visibles/ocultos mediante el método *Digi.ShowCode(código, mostrar)* ó *Digi.ShowCode(lista_de_códigos, mostrar)*

Este método admite como parámetro el nombre de un código o un enumerador de nombres de códigos y como segundo parámetro un booleano indicando si el código es o no visible.

Podemos acceder al tipo *Digi21.DigiNG.DigiTab.DigiTab* cargado mediante la propiedad de solo lectura *Digi.DigiTab*.

El tipo *Digi21.DigiNG.DigiTab.DigiTab* tiene una propiedad por defecto que recibe como parámetro el nombre de un código y devuelve o una excepción indicando que el código que intentamos recuperar no existe o un tipo *Digi21.DigiNG.DigiTab.NodeDigiTab* con información de ese código (descripción, colores, grosores, visibilidad...).

Laboratorio 4 (Mejorando las órdenes ON y OFF de DigiNG)

Las órdenes ON y OFF no son muy intuitivas pues no muestran cuadros de diálogo al usuario.

En este laboratorio vamos a crear una orden nueva que denominaremos OnOff que mostrará un cuadro de diálogo con los códigos de la tabla de códigos activa. Cada código se mostrará con un *checkbox* a su izquierda que indicará si el código es visible o invisible. El usuario podrá activar/desactivar la visualización de un determinado código activando o desactivando el *checkbox*.

Crearemos un formulario con un control *CheckedListBox* que es un *ListBox* que muestra un *checkbox* en cada ítem añadido. Además dispone de un evento *ItemCheck* al que nos podemos conectar para ser notificados cada vez que el usuario haga clic en el *checkbox* asociado a cada ítem añadido.

Podemos añadir ítems a nuestro *CheckedListBox* con las distintas sobrecargas del método *Add*. Una de ellas recibe como parámetros un *System.Object*, es decir, cualquier tipo de .NET porque todos los tipos .NET heredan directa o indirectamente de *System.Object*. y un booleano que indica el estado inicial de este ítem, activado o desactivado.

Si añadimos ítems con esta sobrecarga (pasando como primer parámetro un objeto), el texto que se mostrará en el *CheckedListBox* será el texto que devuelva el método *ToString* del objeto.

Crearemos una clase que denominaremos *DatoEnListBox* que contendrá dos propiedades *Nombre* y *Descripción* con el nombre y la descripción de cada código añadido al control y sobrecribiremos su método *ToString* para que el *CheckedListBox* muestre el nombre del código, un tabulador y la descripción del código por cada código añadido.

Si el usuario hace clic sobre el *checkbox* de algún código, el control *CheckedListBox* nos lo notificará ejecutando nuestro manejador de eventos para ese evento y pasándonos como parámetro (de tipo *System.Object*) con el ítem que ha sido pulsado. Como sabemos que todos los items del control *CheckedListBox* son de tipo *DatoEnListBox*, podremos hacer un casting de ese ítem pasado por parámetro al manejador de eventos a un tipo *DatoEnListBox* y allí tendremos el nombre del código que ha sido pulsado.

Tan solo nos quedará informarle a *DigiNG* que active o desactive el código y por último forzarle a renderizar la escena.

1. Ejecutamos Microsoft Visual C# 2010 Express.
2. Seleccionamos *Archivo/Nuevo proyecto*.
3. En el cuadro de diálogo Nuevo proyecto seleccionamos *Biblioteca de clases*.
4. En Nombre tecleamos *OnOff* y aceptamos el cuadro de diálogo.
5. Añadimos referencia a los ensamblados *Digi21.DigiNG* y *Digi21.DigiNG.Plugin*.
6. Cambiamos el nombre del archivo Class1.cs por *OrdenOnOff.cs*.
7. Anadimos la siguiente cláusula using al comienzo del archivo OrdenOnOff.cs

- ```
using Digi21.DigiNG.Plugin;
```
8. Decoramos la clase con el atributo `[Command(Name="tu_nombre.OnOff")]`
  9. Hacemos que la orden herede de la clase *Command*
  10. Hacemos un constructor público para nuestra clase.
  11. Añadimos un manejador de eventos para el evento *Initialize*. Tecleando `this.Initialize +=` y pulsando dos veces tabulador.
  12. Pulsamos con el botón derecho del ratón en nuestro proyecto en la ventana Explorador de soluciones y seleccionamos *Agregar/Clase...*
  13. En el cuadro de diálogo *Agregar nuevo elemento* tecleamos *DatoEnListBox.cs*.
  14. Declaramos en la clase *DatoEnListBox* las propiedades de lectura/escritura *Nombre* y *Descripción* (de tipo *string*) y sobrecargamos el método *ToString* para que se muestre el nombre, tabulador, descripción.

A mí me ha quedado así:

```
class DatoEnListBox
{
 public string Nombre { get; set; }
 public string Descripción { get; set; }

 public override string ToString()
 {
 return string.Format("{0}\t{1}",
 Nombre,
 Descripción);
 }
}
```

15. Pulsamos con el botón derecho del ratón en nuestro proyecto en la ventana Explorador de soluciones y seleccionamos *Agregar/Nuevo* elemento del menú contextual.
16. En el cuadro de diálogo *Agregar nuevo elemento - OnOff* seleccionamos *Windows Forms*.
17. En Nombre, tecleamos *FormularioOnOff* y aceptamos el cuadro de diálogo pulsando el botón Agregar.
18. En el panel Propiedades, cambiamos las siguientes propiedades de nuestro formulario:

| Propiedad          | Valor                      |
|--------------------|----------------------------|
| <b>Text</b>        | Visualizar/Ocultar códigos |
| <b>MaximizeBox</b> | False                      |
| <b>MinimizeBox</b> | False                      |
| <b>ShowIcon</b>    | False                      |

19. Arrastramos del cuadro de herramientas un *CheckedListBox* a nuestro formulario y le cambiamos el tamaño para que ocupe toda la superficie del formulario.
20. En el panel de propiedades, cambiamos las siguientes propiedades al control *CheckedListBox*:

| Propiedad     | Valor                    |
|---------------|--------------------------|
| <b>(Name)</b> | IbCódigos                |
| <b>Anchor</b> | Top, Bottom, Left, Right |

21. Pulsamos la tecla F7 para visualizar el código de nuestro formulario.
22. Añadimos las siguientes cláusulas using al principio del archivo:

```
using Digi21.DigiNG.Plugin;
using Digi21.DigiNG.DigiTab;
```

23. Añadimos un campo de tipo Digi21 que denominaremos Digi al comienzo del formulario.
24. Añadimos un parámetro de tipo DigiNG al constructor del formulario, asignamos el campo local Digi con el valor pasado al constructor del formulario y justo después de la llamada a *InitializeComponent* (que es cuando podemos comenzar a utilizar los controles de nuestro formulario) añadimos el siguiente código para rellenar el control con todos los códigos de la tabla de códigos activa.

```
foreach (NodeDigiTab código in Digi.DigiTab.Codes)
{
 DatoEnListBox dato = new DatoEnListBox();
 dato.Nombre = código.Name;
```

```

 dato.Descripción = código.Description;
 lbCódigos.Items.Add(dato, código.Visibility);
 }
}

```

25. Tan solo nos queda conectarnos con el evento *ItemCheck* del control *CheckedListBox*. Podríamos haberlo hecho en la vista diseño del formulario, en la ventana *Propiedades*, pero no lo hemos hecho así porque si lo hubiéramos hecho así se habría añadido el código de conexión con el evento como parte del código del método *InitializeComponent* lo que quiere decir que cada vez que añadimos un ítem con el método *Add* se nos hubiera llamado a nuestro manejador de eventos, y no nos interesa. Queremos conectarnos al manejador de eventos una vez lleno el control *CheckedListBox*.
26. Teclea justo después de rellenar la lista *lbCódigos.ItemCheck +=* y verás que justo al teclear el carácter igual, Visual Studio te muestra un *tooltip* proponiéndote un método. Pulsa el tabulador dos veces y se habrá añadido el código para conectarte con el evento mediante el manejador de eventos *lbCódigos\_ItemCheck*.
27. El método *lbCódigos\_ItemCheck* será el método al que se llame cada vez que el usuario cambie el estado de un código. El parámetro *e* de tipo *ItemCheckEventArgs* dispone de dos propiedades que nos interesan: *Index* que nos indica el índice del código al que el usuario le ha cambiado el estado de visibilidad y *NewValue*. Si la propiedad *NewValue* es *CheckState.Checked* significa que el usuario ha activado la casilla de verificación, si no, pues que la ha desactivado.
- Como se ha explicado al comienzo del laboratorio, vamos a obtener el objeto *DatoEnListBox* de la posición *e.Index*, llamaremos al método *Digi.ShowCode* para ordenarle a Digi que active o desactive la visualización del código y por último forzaremos a que Digi renderice la escena. Veamos el código del manejador de eventos:

```

private void lbCódigos_ItemCheck(object sender, ItemCheckEventArgs e)
{
 DatoEnListBox código = (DatoEnListBox)lbCódigos.Items[e.Index];
 Digi.ShowCode(código.Nombre, e.NewValue == CheckState.Checked);
 Digi.RenderScene();
}

```

28. Y ya casi hemos terminado. Volvamos a nuestra clase *OrdenOnOff*. ¿Te acuerdas que habíamos añadido un manejador de eventos para el evento *Initialize*? Modifica el código del manejador de eventos para que instancie nuestro formulario, que lo muestre de forma modal (llamando al método *ShowDialog*) y una vez finalizado, llama al método *Dispose()* para auto-destruir la orden *OnOff*.
29. Compila el proyecto y corrige los errores.

30. Si estás utilizando Visual C# 2010 Professional pasa al siguiente punto, si estás utilizando la versión Express, el proyecto está almacenado en un directorio desconocido. Guarda el proyecto pulsando el botón *Guardar Todo* de la barra de herramientas *Estándar* o pulsando la combinación de teclas *Ctrl+Mayúsculas+S*. En el cuadro de diálogo *Guardas proyecto* pulsa el botón *Guardar*. Ahora tu proyecto está en una ubicación conocida (si no has cambiado nada será en *Mis documentos\Visual Studio 2010\Projects*).
31. Ejecuta Digi3D 2011 y registra el proyecto.
32. Ejecuta la orden (en mi caso se llama *joseangel.onoff*) y se mostrará el formulario.

#### *Laboratorio 5 (Orden OnOff no modal)*

En el laboratorio anterior hicimos que nuestro cuadro de diálogo fuera un cuadro de diálogo modal (es decir, bloqueante). El usuario no podía interactuar con el resto del programa mientras se estuviera mostrando el cuadro de diálogo OnOff.

En este laboratorio vamos a modificar el proyecto del laboratorio anterior para hacer que el cuadro de diálogo no sea modal y que permanezca visible hasta que o el usuario lo cierre o el usuario cierre el archivo de dibujo.

Para mostrar un formulario de una manera no modal, tendremos que llamar al método *Show()* en vez de llamar al método *ShowDialog()*. Si modificamos el manejador de eventos *OrdenOnOff\_Initialize()* de nuestro proyecto anterior podríamos dejarlo así:

```
void OrdenOnOff_Initialize()
{
 FormularioOnOff formulario = new FormularioOnOff(this.Digi);
 formulario.Show();
 Dispose();
}
```

Pero no funcionará, porque inmediatamente después de la llamada al método *Show* del formulario estamos llamando al método *Dispose()* que destruye la orden.

Si queremos convertir nuestro formulario en un formulario no modal tendremos que aprender qué es la pila de órdenes de DigiNG.

¿Sabes que es una pila?

Es un contenedor de tipo LIFO (Last Input First Output), lo que significa que el último que ha entrado es el primero que sale.

Imagínate que eres un pinche en un restaurante y te toca lavar platos.

Tienes una pila de platos que lavar. Mientras lavas un plato un camarero añade más platos a tu pila de platos.

El siguiente plato que lavarás será el que está arriba del todo de la pila y el último que laves será el primero que se añadió a la pila.

DigiNG almacena las órdenes ejecutadas por el usuario en una pila. Cada vez que el usuario le ordena a DigiNG que se ejecute una orden, DigiNG crea la orden y la pone inmediatamente en la parte superior de la pila de órdenes.

La orden se puede auto-desapilar o puede permanecer allí hasta que el usuario pulse la tecla *Escape* lo que implicará que la orden se desapilará y además se destruirá.

DigiNG envía eventos únicamente a la primera orden de la pila de órdenes, las órdenes anteriores no recibirán eventos de DigiNG hasta que vuelvan a ser las primeras de la pila.

Queremos que nuestra orden muestre un formulario con los códigos de la tabla de códigos y que el cuadro de diálogo permanezca visible hasta que el usuario lo cierre o cierre el modelo. Podríamos simplemente no llamar al método *Dispose*, dejando nuestro manejador así:

```
void OrdenOnOff_Initialize()
{
 FormularioOnOff formulario = new FormularioOnOff(this.Digi);
 formulario.Show();
}
```

Pero no lo estaríamos haciendo bien, pues la orden seguiría siendo la primera orden de la pila de órdenes, es decir, que si el usuario hace clic con el ratón, DigiNG se lo notificará a nuestra orden *OnOff*, y a nuestra orden *OnOff* no le interesa ningún evento de DigiNG.

La solución consiste en desapilar la orden de la pila de órdenes. Para ello podemos utilizar el método *Digi.PopCommand()*. Si ejecutamos este método, desapilamos la orden que esté en el tope de la pila de órdenes.

Este método devuelve un booleano indicando si quedan órdenes en la pila de órdenes, de modo que si nos interesa simular el comportamiento de la orden de DigiNG:

#### ANULA\_ORDENES

Podríamos hacerlo así:

```
while (Digi.PopCommand())
{
}
```

Si desapilamos una orden, ésta ya no volverá a recibir eventos por parte de DigiNG hasta que la volvamos a apilar.

Aquí tienes la modificación que tienes que hacer al proyecto del laboratorio anterior para que el cuadro de diálogo sea no modal:

```
void OrdenOnOff_Initialize()
{
 FormularioOnOff formulario = new FormularioOnOff(this.Digi);
 formulario.Show();
 Digi.PopCommand();
}
```

## Laboratorio 6 (Panel acoplable OnOff)

En este laboratorio vamos a mejorar el interfaz de usuario de nuestra orden OnOff no modal.

Si te fijas, la orden que hemos hecho en el laboratorio anterior rompe la estética de Digi3D ya que el cuadro de diálogo tiene un marco con un tamaño distinto que el resto de ventanas de Digi3D y además la ventana desaparece cuando el usuario hace clic en DigiNG.

El objetivo de este laboratorio es modificar nuestra orden OnOff para que se muestre como un panel acoplable dentro de la interfaz de usuario de Digi3D.

El objeto App dispone de un método (con varias sobrecargas) que nos permite añadir un panel acoplable.

La sobrecarga más sencilla es la siguiente:

`App.AddPane(Form formulario)`

Si llamamos a este método Digi3D se encargará de crear un panel acoplable cuyo contenido será nuestro formulario, pero tendremos que hacer unos pequeños cambios previamente a nuestro formulario.

1. Carga el proyecto anterior.
2. En el explorador de soluciones, haz doble clic en el archivo *FormularioOnOff.cs* para que se muestre la vista diseño de nuestro formulario.
3. Realiza los siguientes cambios en la ventana de propiedades del formulario:

| Propiedad              | Valor | Causa                                                                                                                                                           |
|------------------------|-------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>FormBorderStyle</b> | False | No queremos que se muestre la ventana de título ni el borde de la ventana en el panel acoplable. Ya se encargará Digi3D de mostrar su propia ventana de título. |
| <b>ShowInTaskBar</b>   | False | Ya no nos hace falta que la ventana aparezca en la barra de tareas de Windows, pues Digi3D no puede ocultar la visualización de un panel acoplable.             |

4. Por último, sustituye el manejador de eventos Initialize de la orden OnOff para que quede de la siguiente manera:

```
void OrdenOnOff_Initialize()
{
 App.AddPane(new FormularioOnOff(this.Digi));
 Digi.PopCommand();
}
```

5. Compila el proyecto, corrige los errores.
6. Ejecuta Digi3D 2011, y ejecuta la orden para comprobar que ahora está perfectamente integrada con Digi3D 2011.

### 3. Acceso a Bases de datos básico

En este capítulo se explican las nuevas características de Digi3D para trabajar con bases de datos. Descubrirás lo que hace Digi3D 2011 “detrás del telón”, cómo establecer una conexión con la base de datos y te familiarizarás con el nuevo interfaz de usuario.

Los ejercicios de este laboratorio se realizarán sobre bases de datos Microsoft Access, pero no es necesario que tengas una licencia de Microsoft Access instalada en el ordenador.

Los laboratorios de este capítulo están ubicados en las carpetas Laboratorio 1, Laboratorio 2, Laboratorio 3, y así sucesivamente. Cada laboratorio dispone de su propia tabla de códigos *Laboratorio.tab.xml* y ese patrón se repite en todos los laboratorios, todas las carpetas tienen un archivo *Laboratorio.tab.xml*.

Podemos aprovecharnos de la característica *sustituidores* para evitar tener que seleccionar la tabla de códigos correspondiente a cada laboratorio, de modo que en el cuadro de diálogo de *Nuevo Proyecto*, en la pestaña *Archivo de dibujo*, en el parámetro *Tabla de códigos* en la sección *Entorno* yo personalmente pondría para todos los laboratorios de este capítulo la siguiente cadena:

`$(PathOfDrawingFile)Laboratorio.tab.xml`

Muchos de los laboratorios tienen un menú personalizado para realizar las tareas del propio laboratorio y en todos coincide que el menú está en el propio directorio del laboratorio y su nombre siempre es: *Laboratorio.menu.xml*.

Podemos aprovecharnos aquí también de los sustituidores y poner en el campo de menú la siguiente cadena:

`$(PathOfDrawingFile)Laboratorio.menu.xml`

## Qué hace Digi3D 2011 por nosotros

Si recuerdas del capítulo de codificación múltiple, añadir múltiples códigos a una entidad en versiones de Digi3D anteriores a 2011 era complicado porque implicaba memorizar el código a añadir, ejecutar la orden ALTA, añadir un atributo en memoria...

Si añadir codificación múltiple con versiones anteriores era complicado, añadir atributos de base de datos lo era aún más, porque implicaba tener que cambiar de aplicación: de Digi3D a (usualmente) *Microsoft Access*.

El usuario normalmente tenía que:

- a) Añadir en la base de datos un nuevo registro (esto implicaba cambiar de aplicación, típicamente a Microsoft Access) y memorizar el valor asignado en la clave primaria de ese registro (típicamente denominado mslink).
- b) Averiguar el número interno de la tabla en la que se había añadido esa información, consultando el número en la tabla CATDBS o MSCatalog (en función del modelo de Base de Datos utilizado).
- c) Cambiar a Digi3D y ejecutar la orden ALTA para crear un nuevo registro.
- d) Asignar en el campo Tabla el número de tabla.
- e) Asignar en el campo Registro el valor de la clave primaria (mslink).

Estos pasos son difíciles y es muy fácil equivocarse.

Digi3D 2011 realiza todos estos pasos de forma automática y transparente al usuario gracias a los siguientes cambios que se han añadido al programa:

1. Ahora se puede indicar en la tabla de códigos Digi.tab.xml si un determinado código enlaza con una tabla de la base de datos.
2. Han desaparecido los atributos como tales. Han desaparecido las órdenes ALTA, PONER\_ATR, ... Ahora los atributos se almacenan junto con los códigos de las entidades.
3. Se han incorporado ventanas nuevas que muestran al usuario los campos de una determinada tabla de la base de datos. Cuando se selecciona un campo con enlace a una tabla de la base de datos, Digi3D 2011 le muestra al usuario los campos de la tabla permitiendo al usuario concentrarse únicamente en rellenar la información.
4. Se ha incorporado un motor capaz de crear registros en la base de datos y asociárselos a las entidades de forma automática y transparente para el usuario.

## Códigos desde el punto de vista de Digi3D 2011

En Digi3D 2011 todas las entidades deben tener al menos un código. Estos códigos pueden estar o no enlazados con la base de datos. Por lo tanto una entidad podrá tener como máximo tantos enlaces con la base de datos como códigos tenga.

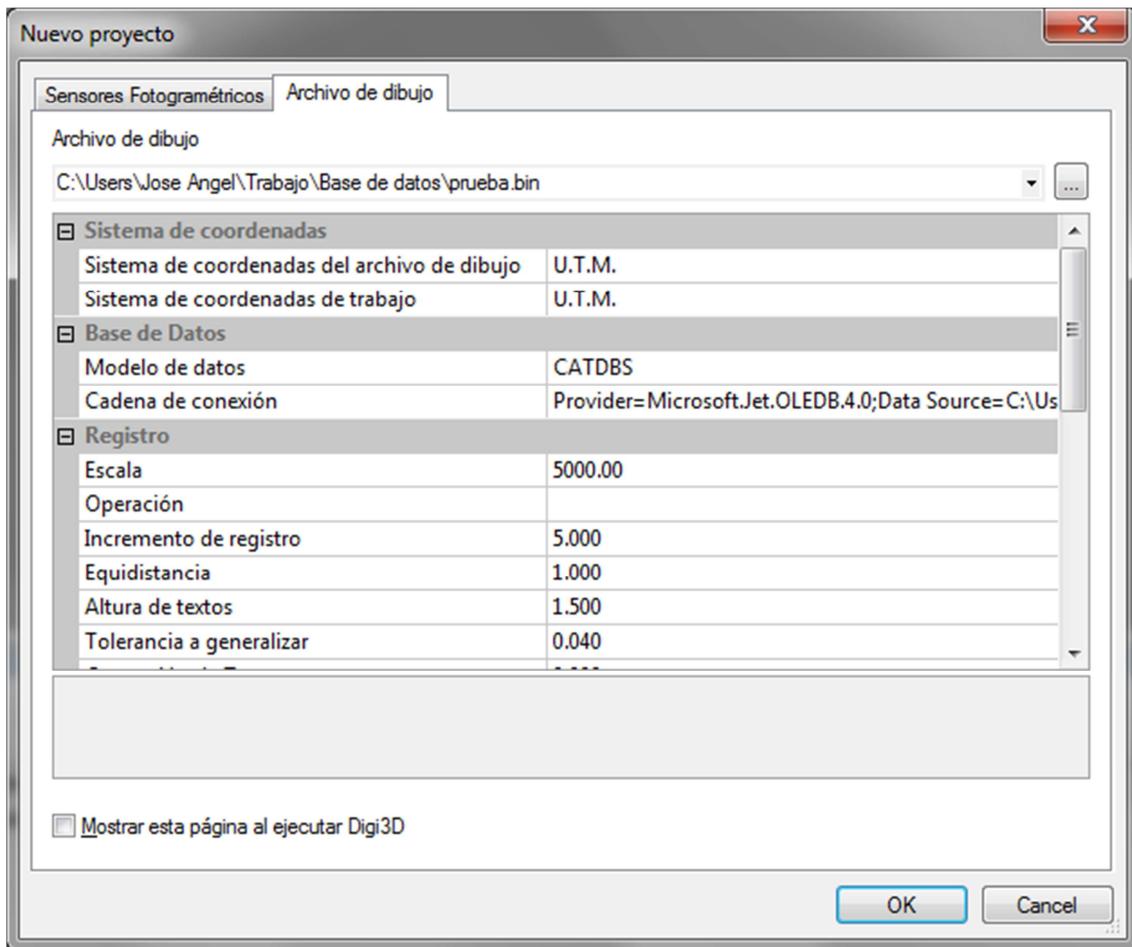
Los códigos están compuestos por los siguientes campos:

| Código (alfanumérico)                                                                                                                                                                              | Tabla (numéricico)                                                                                                                                                                                                                                                                                                                                                                                                                            | Id (numéricico)                                                                                                                                                                                           |
|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <ul style="list-style-type: none"> <li>Nombre del código.</li> </ul> <p>Aquí se almacena el nombre del código, como por ejemplo "AAA010". Se pueden almacenar códigos de hasta 256 caractéres.</p> | <ul style="list-style-type: none"> <li>1: <i>SIN_ENLACE_BBDD</i> Para indicar que este código no tiene enlace con base de datos ni se desea. Cuando se almacene la entidad, no se creará un alta nueva en la base de datos.</li> <li>&gt;1 Para indicar que el código ya está enlazado con un registro de la base de datos, por lo tanto si se almacena una entidad no se creará ningún alta en la base de datos porque ya existe.</li> </ul> | <ul style="list-style-type: none"> <li>0: ID_SIN_ASIGNAR o el número de registro (en la tabla apuntada por el campo Tabla) donde está almacenada la información de base de datos de el código.</li> </ul> |

Si una entidad tiene algún código con valor *Tabla* mayor que 1 podemos localizar la información en la base de datos para ese código de la entidad en el registro con el mismo identificador que el campo *Id* del código.

## Conexión con la base de datos

Si se desea indicar a Digi3D 2011 que debe establecer una conexión con una determinada base de datos al cargar un archivo de dibujo, se deben llenar los parámetros *Modelo de datos* y *Cadena de conexión* que aparecen en la categoría *Base de datos* en la pestaña *Archivo de dibujo* del cuadro de diálogo *Nuevo proyecto*.

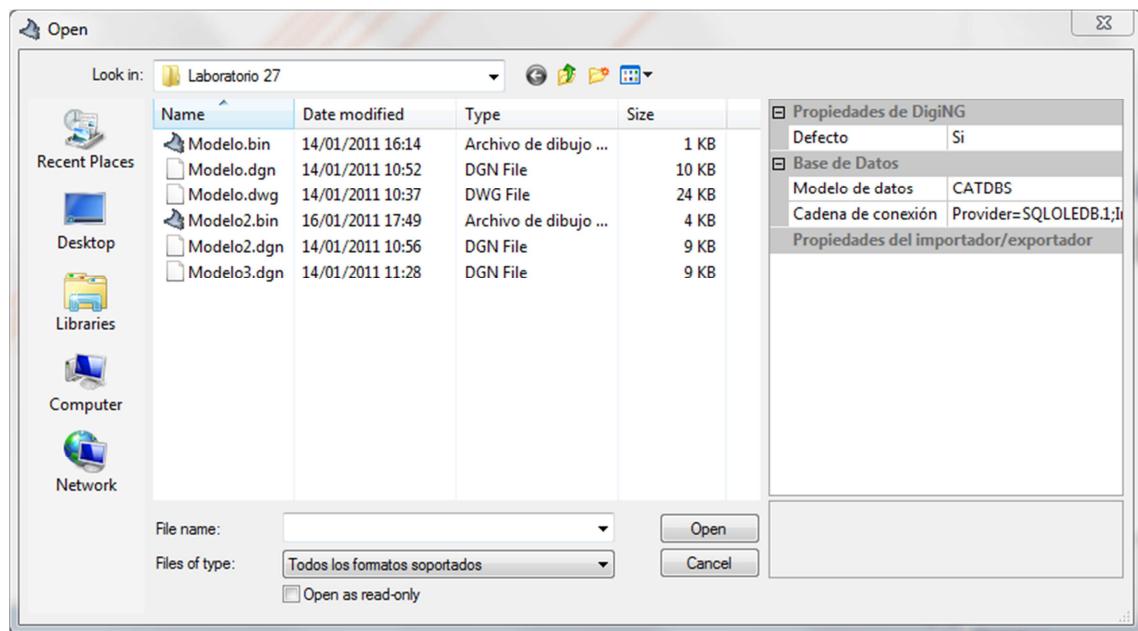


En *Modelo de datos* podemos seleccionar: *Sin conexión con base de datos*, *CATDBS*, *Geographics*,... (Más adelante aprenderás lo que es un modelo de datos para Digi3D 2011).

Si en *Modelo de datos* seleccionamos *Sin conexión con base de datos*, Digi3D 2011 no se conectará con la base de datos, si elegimos cualquier otra opción, intentará utilizar el modelo de datos seleccionado para conectarse con la base de datos.

El campo “Cadena de conexión” es el parámetro que se pasa al modelo de datos para que se conecte con la base de datos.

Estos dos campos también aparecen en el cuadro de diálogo común de abrir archivo cuando el usuario ejecuta la orden CARGA\_F:



## Ventana Campos de la base de datos

Digi3D 2011 incorpora una nueva ventana *Campos de la base de datos* donde podemos indicar la información que queremos almacenar en la base de datos la próxima vez que se almacene una entidad nueva.



Esta ventana está íntimamente relacionada con la ventana de *Códigos Activos*, ya que únicamente muestra los atributos del código seleccionado en la de *Códigos activos*.

Si el código seleccionado en la ventana de *Códigos activos* no está asociado con ninguna tabla, la ventana de *Atributos de base de datos* no mostrará ninguna información.

En la siguiente captura de pantalla podemos ver que el código seleccionado en la ventana de códigos activos es el código 0 ‘Código por defecto’. Este código no está enlazado con ninguna tabla en la base de datos, por lo tanto, la ventana *Campos de la base de datos* no nos muestra ninguna información.

The screenshot shows two windows side-by-side. The top window is titled 'Códigos activos' and contains a table with columns: Código, Tabla, ID, Color, and Descripción. It lists three items: '0' (Color 41, Descripción 'Código por defecto'), '041279 2' (Color 41, Descripción 'Carretera'), and '062242 3' (Color 41, Descripción 'Recinto deportivo'). The bottom window is titled 'Campos de la base de datos' and is currently empty.

Si seleccionamos el código 1 ‘Vías de comunicación’ (que sí que está enlazado con una tabla de la base de datos) la ventana *Campos de la base de datos* nos va a mostrar los campos de la tabla asociada con ese código:

The screenshot shows the same two windows. The 'Códigos activos' window remains the same. The 'Campos de la base de datos' window now displays the fields for the selected code (Vías de comunicación). The table shows the following fields:

| Nombre    | Paseo de la castellana |
|-----------|------------------------|
| Ancho     | 50.000000              |
| Carriles  | 8                      |
| Velocidad | 50                     |

Y si seleccionamos el código 2 ‘Recintos deportivos’ la ventana de *Campos de la base de datos* nos mostrará los atributos de la tabla asociada con ese código:

| Código | Tabla | ID | Color | Descripción        |
|--------|-------|----|-------|--------------------|
| 0      |       |    | 41    | Código por defecto |
| 041279 | 2     |    | 41    | Carretera          |
| 062242 | 3     |    | 41    | Recinto deportivo  |

| [Recintos deportivos] |  |
|-----------------------|--|
| Nombre                |  |
| Tipo de actividad     |  |

La ventana de *Campos de la base de datos* recuerda la última información que hemos almacenado en cada tabla, de modo que si ahora ejecutásemos la orden COD=0

| Código | Tabla | ID | Color | Descripción        |
|--------|-------|----|-------|--------------------|
| 0      |       |    | 41    | Código por defecto |

| [Recintos deportivos] |  |
|-----------------------|--|
| Nombre                |  |
| Tipo de actividad     |  |

... y luego seleccionásemos el código 2 con la orden COD=041279

The screenshot shows two overlapping windows. The top window is titled 'Códigos activos' and contains a table with columns: Código, Tabla, ID, Color, and Descripción. A row is selected with the values: 041279, 2, 41, and Carretera. The bottom window is titled 'Campos de la base de datos' and shows a table with fields: Nombre (Paseo de la castellana), Ancho (50.000000), Carriles (8), and Velocidad (50). Both windows have standard Windows-style toolbars at the top.

Comprobamos que el programa ha memorizado los últimos valores (Nombre = Paseo de la castellana, ancho=50, Carriles=8 y Velocidad=50).

Todas las entidades que registrásemos a partir de este momento tendrían asociada en la base de datos esta combinación de valores.

Si queremos que el programa muestre la información por defecto (que más adelante aprenderás se puede asignar en la tabla de códigos Digi.tab.xml), podemos pulsar el primer botón de la barra de herramientas de la ventana *Campos de la base de datos* “Asignar los valores por defecto de Digi.tab”.

## Editor de códigos

Si se selecciona en el editor de códigos un código con enlace a base de datos, este mostrará en la parte inferior del cuadro de diálogo la información almacenada en la base de datos para ese código.

En la siguiente captura de pantalla podemos ver el cuadro de diálogo EDITAR\_COD en acción:



El cuadro de diálogo aporta mucha información:

1. La entidad que se está editando tiene un único código.
2. El código de la entidad es el 041279(Carretera).
3. Además el código tiene enlace con base de datos, en el registro 6 de la tabla 2.
4. La tabla número 2 en la tabla: *Vías de comunicación*.
5. El registro 6 de la tabla *Vías de comunicación* en la base de datos asociada con el archivo de dibujo tiene la siguiente información:

| IDVial | Nombre                | Ancho | Carreles | Velocidad |
|--------|-----------------------|-------|----------|-----------|
| 6      | Carretera de Canillas | 10.0  | 4        | 50        |

## *Laboratorio 1 (Familiarizándonos con el nuevo interfaz de usuario)*

En este laboratorio vamos a familiarizarnos con el nuevo interfaz de usuario de Digi3D 2011.

1. Arrancamos Digi3D 2011.
2. Cargamos el archivo [ruta al laboratorio de base de datos básico]\Laboratorio 1\Modelo.bin
3. En la sección titulada *Base de Datos*, en la opción *Modelo de Datos* seleccionamos CATDBS.
4. Si estás ejecutando Digi3D 2011 de 32 bits, en el campo titulado *Cadena de Conexión* tecleamos lo siguiente:  

```
Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=$(PathOfDrawingFile)Laboratorio.mdb;Persist Security Info=False
```
5. Si estás ejecutando Digi3D 2011 de 64 Bits, no puedes conectarte con una base de datos Access, así que tu cadena de conexión deberá ser:  

```
Provider=SQLOLEDB.1;Integrated Security=SSPI;Persist Security
Info=False;Initial Catalog=LaboratorioBBDBBasic01;Data
Source=.\\sqlexpress
```
6. Pulsamos el botón OK.
7. Comprobamos que Digi3D 2011 carga el archivo de dibujo correctamente.
8. Activamos el cuadro de herramientas si no está activo *Mayúsculas+Alt+H* (o en el menú *Ver*)
9. Activamos la ventana de Campos de la base de datos si no está activa *Mayúsculas+Alt+B* (o menú *Ver*).
10. Activamos la ventana de tentativos sino está activa *Mayúsculas+Alt+S* (o en el menú *Ver*).
11. Seleccionamos los códigos 0, 041279 y 062242 (podemos hacerlo con la opción del menú del laboratorio en el cuadro de herramientas).
12. Comprobamos que en la ventana de Códigos activos aparecen los tres códigos.
13. El código seleccionado por defecto en la ventana de Códigos activos será el primero 0 (Código por defecto). Como este código no tiene enlace con ninguna tabla de la base de datos, la ventana de Campos de la base de datos se muestra vacía.
14. Seleccionamos el código 041279 (Carretera) y comprobamos que la ventana de Campos de la base de datos muestra los campos: Nombre, Ancho, Carriles y Velocidad además de indicarnos en el título de la propia ventana el nombre de la tabla donde se va a almacenar la información: "Vías de comunicación".
15. Rellenamos todos los campos.
16. Seleccionamos el código 062242 (Recinto deportivo) y comprobamos que la ventana de Campos de la base de datos muestra los campos: Nombre y Tipo de Actividad además de indicarnos en el título de la propia ventana el nombre de la tabla donde se va a almacenar la información: "Recintos deportivos".
17. Rellenamos todos los campos.
18. Dibujamos una línea.
19. Tentativamos la línea para verla en la ventana de Tentativos y comprobar que sus códigos son:

| Código | Tabla | Id <sup>4</sup> |
|--------|-------|-----------------|
| 0      |       |                 |
| 041279 | 1     | 1               |
| 062242 | 2     | 1               |

20. Cerramos Digi3D.

---

<sup>4</sup> Los valores del campo Id no tienen por qué coincidir con los de esta tabla (si la base de datos ya contiene información previa).

## Cada entidad genera su propio conjunto de registros en la base de datos

Si estamos **conectados** a una base de datos, **cada vez que se almacena** una entidad en el archivo de dibujo, Digi3D 2011 **creará** la base de datos tantos **registros** como códigos de la entidad tengan enlace a la base de datos.

El párrafo anterior se cumple **siempre**, independientemente de si la entidad almacenada es una entidad nueva, una copia de una entidad existente, una modificación de una entidad existente.

Hay órdenes en Digi3D 2011 que no crean entidades como por ejemplo la orden ZOOM, que simplemente cambia el factor de zoom.

Y hay otras órdenes que crean entidades nuevas, como las órdenes LINEA, PUNTO, TEXTO, CIR2P...

Y por último, un tercer tipo de órdenes, que son las que modifican entidades existentes, como MOVER, EXT, CAMB\_AT, CAMT\_IT,...

Las órdenes que crean entidades nuevas siguen el siguiente patrón:

1. Crean en memoria una entidad con los códigos activos en la ventana de códigos activos.
2. Configurar la entidad en memoria (le añaden vértices, ....)
3. Le ordenan a Digi3D 2001 que guarde la entidad llamando al procedimiento *GuardarEntidad*.

Las órdenes que modifican entidades existentes siguen siempre el siguiente patrón:

1. Crear una copia en memoria de la entidad a modificar.
2. Modificar la copia en memoria (cambiarle el código en el caso de la orden CAMB\_COD, desplazar sus vértices en el caso de la orden MOVER, añadir vértices en caso de la orden MOD, ...)
3. Le ordenan a Digi3D 2001 que guarde la entidad llamando al procedimiento *GuardarEntidad*.
4. Eliminar la entidad original del archivo de dibujo.

A continuación tenemos el pseudocódigo del procedimiento *GuardarEntidad*:

```
procedimiento GuardarEntidad(entidad)
 si ConectadosconBBDD entonces
 para cada código ε entidad (1)
```

```

 si código.Tabla es mayor que SIN_ENLACE_BBDD entonces(1)
 si código.ID es ID_SIN_ASIGNAR entonces (2)
 código.ID ← CreaRegistroBBDDConDatosVentanaCamposDDBB(3)
 si no
 código.ID ← CreaCopiaRegistro(código.ID)(4)
 fin si
 fin si
 fin para cada
fin si
GuardarEntidadEnArchivo(entidad) (5)
fin procedimiento

```

Vamos a simular el comportamiento de una orden que crea una entidad nueva:

Cuando seleccionamos códigos activos con las órdenes COD, COD+, CLONAR, o mediante programación, los códigos que tienen enlace con base de datos se crean con el valor TABLA=[numero de tabla en función de la base de datos] y con valor ID=[ID\_SIN\_ASIGNAR].

Veamos una captura de pantalla en la que se han seleccionado como códigos activos los códigos 0, 041279 y 062242.

| Códigos activos |       |    |                    |             |
|-----------------|-------|----|--------------------|-------------|
| Código          | Tabla | ID | Color              | Descripción |
| 0               |       | 41 | Código por defecto |             |
| 041279          | 2     | 41 | Carretera          |             |
| 062242          | 3     | 41 | Recinto deportivo  |             |

Los códigos 041279 y 062242 enlazan con las tablas 2 y 3 respectivamente. Los valores ID aparecen en blanco porque no están asignados.

Si ahora dibujamos una entidad nueva, ésta entidad tendrá los siguientes códigos:

| Código | Tabla           | ID             |
|--------|-----------------|----------------|
| 0      | SIN_ENLACE_BBDD | ID_SIN_ASIGNAR |
| 041279 | 2               | ID_SIN_ASIGNAR |
| 062242 | 3               | ID_SIN_ASIGNAR |

Cuando se ordena a Digi3D 2011 que almacene esa entidad el bucle (1) se ejecutará tres veces, la condición (2) no se cumplirá para el primer código, pero sí para los códigos 041279 y 062242.

Como el valor ID de los dos códigos es ID\_SIN\_ASIGNAR, la condición (3) se cumplirá en ambos

casos y Digi3D 2011 creará un registro en las tablas 2 y 3 con la información que ha rellenado el usuario en la ventana *Campos de la base de datos* para esos dos códigos y asignará los valores ID de esos dos códigos, de modo que los códigos de la entidad en el punto (5) son los siguientes:

| Código | Tabla           | ID             |
|--------|-----------------|----------------|
| 0      | SIN_ENLACE_BBDD | ID_SIN_ASIGNAR |
| 041279 | 2               | 4000           |
| 062242 | 3               | 12345          |

Simulemos ahora una orden que modifica una entidad existente:

Si el usuario ejecuta la orden MOVER sobre la entidad anterior, la orden MOVER creará una copia exacta de la entidad en memoria por lo tanto esta copia exacta tendrá los siguientes códigos:

| Código | Tabla           | ID             |
|--------|-----------------|----------------|
| 0      | SIN_ENLACE_BBDD | ID_SIN_ASIGNAR |
| 041279 | 2               | 4000           |
| 062242 | 3               | 12345          |

Luego modificará los vértices de la entidad y llamará al procedimiento *GuardarEntidad* para que Digi3D 2011 almacene la entidad en el archivo de dibujo.

El bucle (1) se ejecutará tres veces, la condición (2) no se cumplirá para el primer código, pero sí para los códigos 041279 y 062242.

Como el valor ID de los dos códigos es mayor que ID\_SIN\_ASIGNAR, la condición (3) no se cumplirá y se ejecutará el código (4), por lo tanto se crearán copias en la base datos para estos dos códigos, por lo tanto se añadirán registros en las tablas 2 y 3 y se actualizarán los valores ID de estos códigos, de modo que los códigos de la entidad en el punto (5) son los siguientes:

| Código | Tabla           | ID <sup>5</sup> |
|--------|-----------------|-----------------|
| 0      | SIN_ENLACE_BBDD | ID_SIN_ASIGNAR  |
| 041279 | 2               | 4344            |
| 062242 | 3               | 12444           |

¿Por qué se crean registros nuevos en vez de aprovechar los existentes?

---

<sup>5</sup> Estos valores son inventados.

Es evidente que si el usuario dibuja una entidad nueva, y ésta tiene un código con un enlace a base de datos, Digi3D 2011 debe almacenar un registro en la base de datos asociado con la entidad.

Lo que no resulta tan evidente es que si el usuario ejecuta la orden MOVER sobre una entidad existente, se cree un nuevo registro en la base de datos para esa entidad, pero existen dos escenarios que justifican que Digi3D 2011 cree registros nuevos siempre:

1. Los cambios en un registro en la base de datos afectan a todas las entidades que enlazan con el registro.  
Si el usuario copia una entidad existente, por ejemplo un edificio que es idéntico geométricamente que otro, pero con diferente uso, uno militar por ejemplo y otro civil, una vez copiado modificará el uso del edificio copiado con la orden EDITAR\_COD para cambiarlo de Militar a Civil. Si se hubiera asignado el mismo registro al edificio nuevo, al cambiar su uso en la base de datos, se estarían modificando el uso de los dos edificios porque los dos enlazan al mismo registro.
2. La información que ve el usuario en la ventana de *Campos de la base de datos* no tiene por qué coincidir con la que realmente se está almacenando en la base de datos.

El posible que Digi3D 2011 esté mostrando al usuario los siguientes campos de la tabla *Vías de comunicación*:

| Nombre | Carriles | Ancho | Velocidad |
|--------|----------|-------|-----------|
|--------|----------|-------|-----------|

Pero la tabla seguro que no es así, al menos tiene un campo más que es el identificador del vial (el campo principal de la tabla) y es posible que ésta además tenga otros campos en los que se almacena información variable:

| IDVial | Nombre | Carriles | Ancho | Velocidad | XMin | Usuario | Fecha/Hora |
|--------|--------|----------|-------|-----------|------|---------|------------|
|--------|--------|----------|-------|-----------|------|---------|------------|

Lo que quiere decir que en el ejemplo anterior, la tabla *Vías de comunicación* posiblemente tiene la siguiente información:

| IDVial | Nombre                | Carriles | Ancho | Velocidad | XMin     | Usuario   | Fecha/Hora       |
|--------|-----------------------|----------|-------|-----------|----------|-----------|------------------|
| 4000   | Carretera de Canillas | 4        | 20    | 50        | 327200.0 | JoseAngel | 18/12/2011 18:22 |
| ...    |                       |          |       |           |          |           |                  |
| 4344   | Carretera de Canillas | 4        | 20    | 50        | 425894.0 | JoseAngel | 16/01/2011 20:00 |

El parámetro *XMin* es variable y almacena la coordenada X mínima de la entidad, por lo tanto el registro 4000 ya no es válido para la entidad 4344 (la entidad que se generó cuando ejecutamos la orden MOVER).

Además es posible que en la base de datos se esté almacenando información de rendimiento, como el nombre del usuario que ha generado el alta y la fecha/hora del alta.

#### *Laboratorio 2 (Comprobando que Digi3D 2011 crea un nuevo registro por cada entidad)*

En este laboratorio vamos a comprobar cómo Digi3D 2011 crea registros nuevos cada vez que se genera una entidad.

1. Arrancamos Digi3D 2011.
2. Cargamos el archivo [ruta al laboratorio de base de datos]\Laboratorio 2\Modelo.bin
3. En la sección titulada *Base de Datos*, en la opción *Modelo de Datos* seleccionamos CATDBS.
4. Si estás ejecutando Digi3D 2011 de 32 bits, en el campo titulado *Cadena de Conexión* tecleamos lo siguiente:  
`Provider=Microsoft.Jet.OLEDB.4.0;Data Source=$(PathOfFile)Laboratorio.mdb;Persist Security Info=False`
5. Si estás ejecutando Digi3D 2011 de 64 Bits, no puedes conectarte con una base de datos Access, así que tu cadena de conexión deberá ser:  
`Provider=SQLOLEDB.1;Integrated Security=SSPI;Persist Security Info=False;Initial Catalog=LaboratorioBBDDBasico2;Data Source=.\sqlexpress`
6. Pulsamos el botón OK.
7. Comprobamos que Digi3D 2011 carga el archivo de dibujo correctamente.
8. Tentativamos la línea dibujada dentro del cuadro y comprobamos el Id del código 041279.
9. Ejecutamos la orden MOVER y movemos la línea. Tentativamos la línea para comprobar que el ID ha cambiado.

## Identidad de geometrías

Acabamos de aprender que si digitalizamos alguna geometría dividida en múltiples partes, se crearán múltiples registros en la base de datos, por lo tanto el valor ID del código de cada una de las partes será distinto, sin embargo, todas estas partes representan la misma geometría.

¿Cómo afecta esto a las órdenes que por su naturaleza unen geometrías, como las órdenes EXT2X, UNIR, y BINTRAM?

El comportamiento de estas órdenes varía en función de si estamos o no conectados con la base de datos.

Veámoslo con un ejemplo:

Supongamos que tenemos la siguiente información en la tabla *Vías de comunicación*:

| Id | Nombre | Borde     |
|----|--------|-----------|
| 11 | N-II   | Izquierdo |
| 12 | N-II   | Izquierdo |
| 13 | N-II   | Derecho   |

Y que disponemos de un archivo de dibujo con tres líneas con el mismo código (que enlaza con esta tabla). Las tres líneas se han digitalizado con continuidad geométrica, de modo que el vértice final de la primera coincide con el primer vértice de la segunda y lo mismo con la tercera.

Las líneas tienen los siguientes códigos:

| Código | Tabla | Id |
|--------|-------|----|
| 041279 | 2     | 11 |
| 041279 | 2     | 12 |
| 041279 | 2     | 13 |

¿Cómo se comportará la orden UNIR si intentamos unir la primera línea con la segunda?

- Si no estamos conectados con la base de datos, la orden UNIR nos mostrará un mensaje de error, pues al comparar los códigos de éstas comprobará que sus IDs son distintos. Como no sabe la información que hay almacenada en la base de datos, no le queda más remedio que mostrarnos un mensaje de error indicando que los códigos no coinciden.
- Si estamos conectados con la base de datos, la orden UNIR nos permitirá unir los dos tramos, ya que aunque tienen distintos IDs, la información almacenada en la base de datos es idéntica: El nombre de la vía en ambos casos es N-II y el borde es el izquierdo.

¿Cómo se comportará la orden UNIR si intentamos unir la segunda línea con la tercera?

1. Si no estamos conectados con la base de datos, no unirá por los mismos motivos en el el caso anterior.
2. Si estamos conectados con la base de datos, no unirá porque el atributo Borde es distinto.

### *Laboratorio 3(Identidad de geometrías)*

En este laboratorio vamos a ver en acción cómo se comporta la orden UNIR con el ejemplo anterior.

1. Arrancamos Digi3D 2011.
2. Cargamos el archivo [ruta al laboratorio de base de datos]\Laboratorio 3\Modelo.bin
3. En la sección titulada *Base de Datos*, en la opción *Modelo de Datos* seleccionamos *Sin conexión con base de datos*.
4. Pulsamos el botón OK.
5. Tentativamos los tres segmentos de línea para comprobar en la ventana Tentativos (si no está visible podemos hacerla visible pulsando Mayúsculas+Alt+S o con la opción del menú *Ver/Tentativos*. Podemos comprobar que las tres líneas enlazan con la tabla 2 con IDs 11, 12 y 13 respectivamente.
6. Ejecutamos la orden UNIR e intentamos unir dos tramos continuos para comprobar que el programa no nos lo permite mostrando un globo indicando que los códigos de la entidad no coinciden.
7. Cerramos el archivo de dibujo.
8. Volvemos a cargar el archivo de dibujo, pero esta vez conectándonos con la base de datos.
9. En la sección titulada *Base de Datos*, en la opción *Modelo de Datos* seleccionamos CATDBS.
10. Si estás ejecutando Digi3D 2011 de 32 bits, en el campo titulado *Cadena de Conexión* tecleamos lo siguiente:  

```
Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=$(PathOfDrawingFile)Laboratorio.mdb;Persist Security Info=False
```
11. Si estás ejecutando Digi3D 2011 de 64 Bits, no puedes conectarte con una base de datos Access, así que tu cadena de conexión deberá ser:  

```
Provider=SQLOLEDB.1;Integrated Security=SSPI;Persist Security
Info=False;Initial Catalog=LaboratorioBBDDBasico3;Data
Source=.\\sqlexpress
```
12. Pulsamos el botón OK.
13. Comprobamos que Digi3D 2011 carga el archivo de dibujo correctamente.
14. Ejecutamos la orden UNIR y unimos el primer tramo con el segundo. Comprobamos que el programa nos ha unido los dos tramos sin problemas.
15. Tentativamos el tramo recién creado para comprobar que su ID es distinto.
16. Intentamos unir este tramo con el último tramo para comprobar que no nos lo permite pues el contenido en la base de datos es distinto.

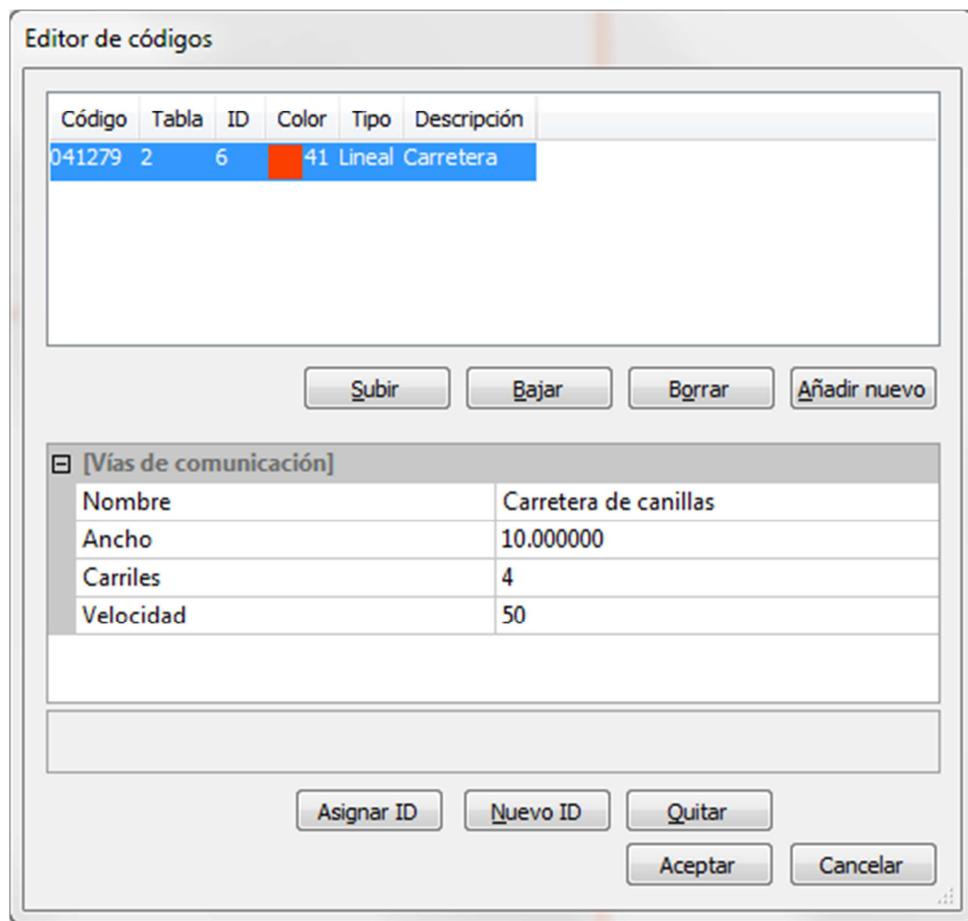


## Editando información existente 1: Editor de códigos

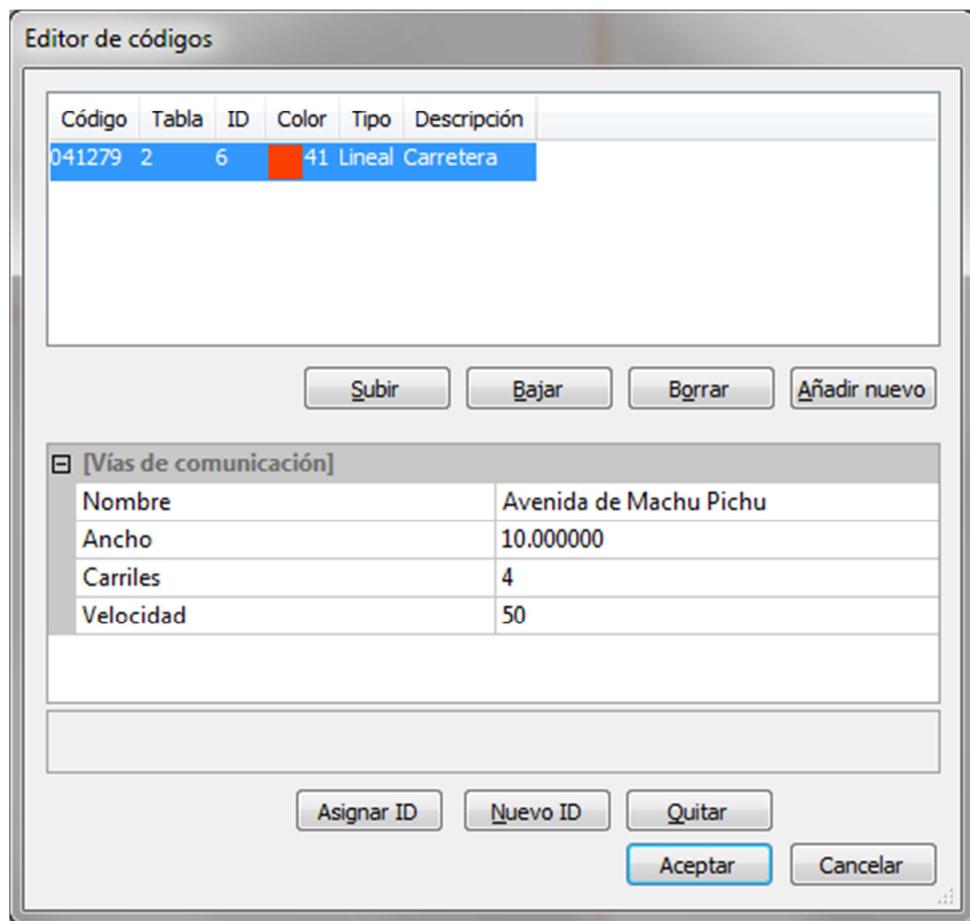
Ya sabemos que Digi3D 2011 crea un registro nuevo en la base de datos por cada entidad nueva que registremos. Ahora vamos a aprender cómo se comporta cuando editamos un registro de la base de datos.

La orden EDITAR\_COD cumple una doble función: Permite añadir/eliminar códigos de una entidad, y si esa entidad está enlazada con una base de datos, permite modificar la base de datos.

Si al editar los códigos de una entidad el cuadro de diálogo tiene el siguiente aspecto:



...si editamos el nombre de la calle, al aceptar el cuadro de diálogo **estamos modificando el registro 6** de la tabla *Vias de comunicación*. Esto es fácilmente comprobable: Si cambiamos el nombre *Carretera de Canillas* por *Avenida de Machu Pichu* y aceptamos el cuadro de diálogo, al volver a editarlo con la orden EDITAR\_COD nos encontramos con lo siguiente:



El campo Nombre del registro 9 de la tabla 6 ahora es *Avenida de Machu Pichu*.

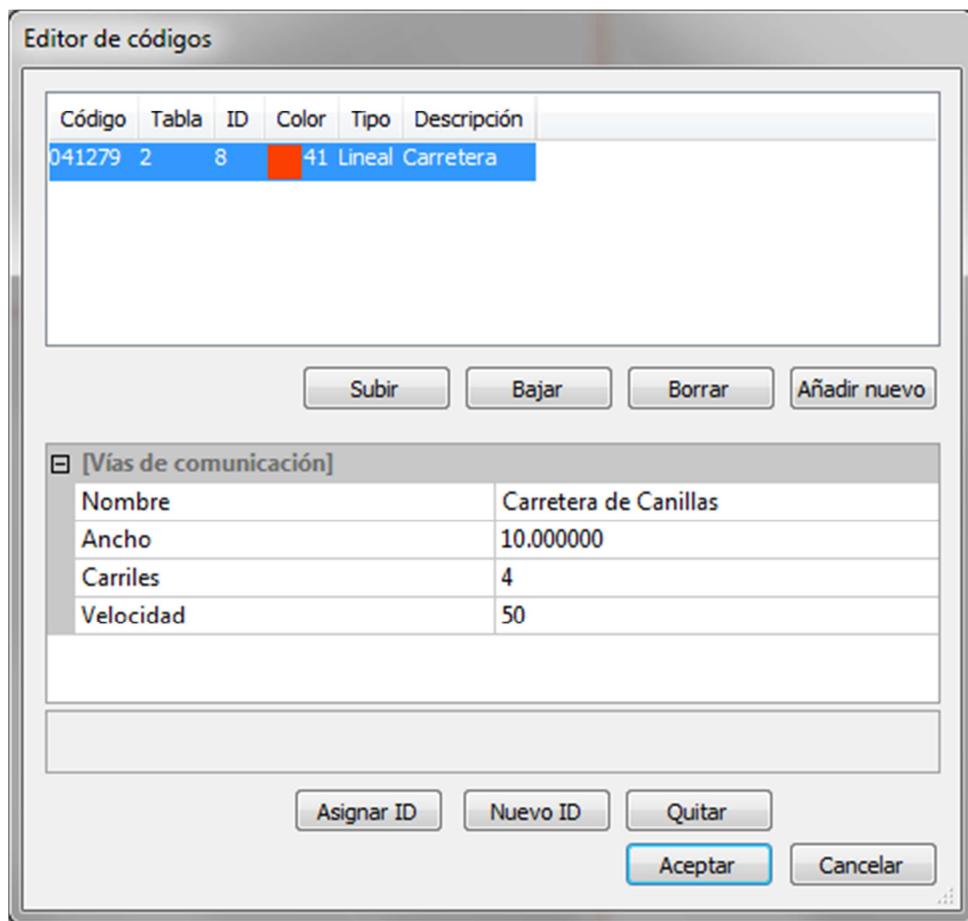
**Tenemos que tener mucho cuidado** a la hora de modificar la base de datos, ya que si alguna otra entidad estaba enlazando con el mismo registro, lógicamente le habremos cambiado el nombre *Carretera de Canillas* por *Avenida de Machu Pichu*.

Si pulsamos el botón *Nuevo ID*, el programa mantendrá la información actual (Nombre=Avenida de Machu Pichi, Ancho=8,...) pero cambiando el valor *ID* del código a *SIN\_ASIGNAR*:



Esto quiere decir que si aceptamos el cuadro de diálogo se creará una nueva entrada en la base de datos la información en los campos.

Si volvemos a poner en el campo *Nombre* el valor *Carretera de Canillas*, aceptamos el cuadro de diálogo y volvemos a editar la misma entidad el resultado será:



Como puedes comprobar, ahora la entidad tiene el ID=8<sup>6</sup>, luego en la tabla tendrá estos dos registros:

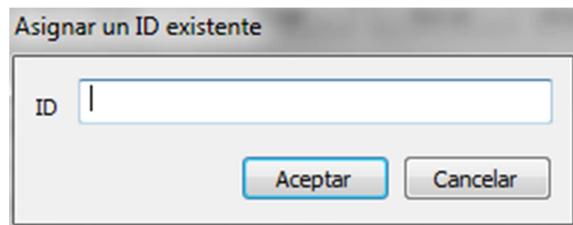
| IDVial | Nombre                 | Ancho | Carriles | Velocidad |
|--------|------------------------|-------|----------|-----------|
| 6      | Avenida de Machu Pichu | 8     | 2        | 50        |
| ...    |                        |       |          |           |
| 8      | Carretera de Canillas  | 8     | 2        | 50        |

Podemos forzar a que un código apunte a un determinado registro de la base de datos. Para ello podemos pulsar el botón Asignar ID.

Si pulsamos el botón *Asignar ID* el programa pregunta por el ID existente:

---

<sup>6</sup> El resultado no tiene por qué ser el inmediatamente superior, lo importante es que sea uno nuevo e inexistente en la base de datos hasta ese momento.



Si tecleo 6 y acepto, el cuadro de diálogo ahora volverá a mostrar los datos del registro 6 (y de hecho, el ID del código será otra vez):

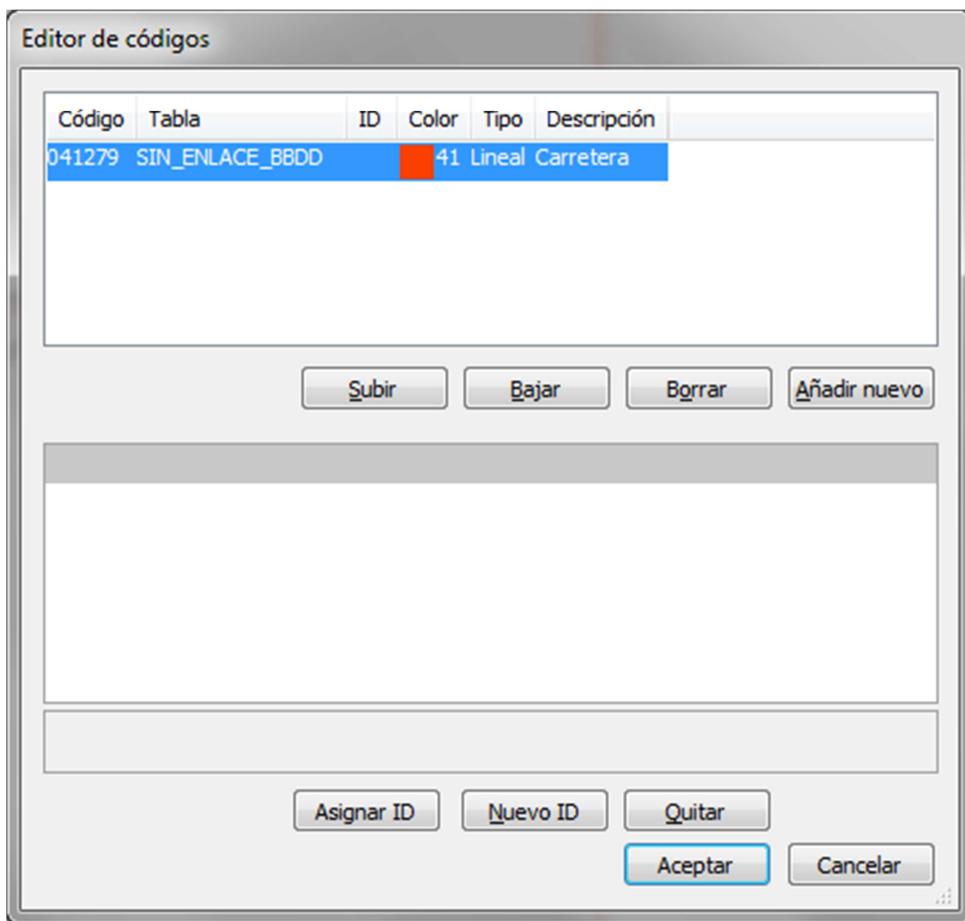
| Código | Tabla | ID | Color  | Tipo | Descripción      |
|--------|-------|----|--------|------|------------------|
| 041279 | 2     | 6  | orange | 41   | Lineal Carretera |

[Vías de comunicación]

|           |                        |
|-----------|------------------------|
| Nombre    | Avenida de Machu Pichu |
| Ancho     | 10.000000              |
| Carriles  | 4                      |
| Velocidad | 50                     |

Si aceptamos el cuadro de diálogo y volvemos a editar la misma entidad, el ID seguirá siendo 6.

A continuación podemos pulsar el botón *Quitar* que marca el código como un código sin enlace a base de datos:



#### *Laboratorio 4 (Editando códigos de una entidad existente)*

En este laboratorio vamos a familiarizarnos con la orden EDITAR\_COD.

1. Arrancamos Digi3D 2011.
2. Cargamos el archivo [ruta al laboratorio de base de datos]\Laboratorio 4\Modelo.bin
3. En la sección titulada *Base de Datos*, en la opción *Modelo de Datos* seleccionamos CATDBS.
4. Si estás ejecutando Digi3D 2011 de 32 bits, en el campo titulado *Cadena de Conexión* tecleamos lo siguiente:  

```
Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=$(PathOfDrawingFile)Laboratorio.mdb;Persist Security Info=False
```
5. Si estás ejecutando Digi3D 2011 de 64 Bits, no puedes conectarte con una base de datos Access, así que tu cadena de conexión deberá ser:  

```
Provider=SQLOLEDB.1;Integrated Security=SSPI;Persist Security
Info=False;Initial Catalog=LaboratorioBBDBBasico4;Data
Source=.\\sqlexpress
```
6. Pulsamos el botón OK.
7. Comprobamos que Digi3D 2011 carga el archivo de dibujo correctamente.
8. Ejecutamos la orden EDITAR\_COD y seleccionamos la línea de carretera para comprobar sus atributos. Comprobamos que es una línea que representa el borde

izquierdo de la Nacional II. El número de registro de base de datos para esta línea es el 5.

9. Cancelamos el cuadro de diálogo *Editor de códigos*.
10. Ejecutamos la orden COPIAR y copiamos la línea con un desplazamiento.
11. Ejecutamos la orden EDITAR\_COD y seleccionamos la línea que acabamos de crear.
12. En el cuadro de diálogo *Editor de Códigos* cambiamos el parámetro Borde de Izquierda a Derecha y aceptamos el cuadro de diálogo.
13. Volvemos a editar la línea. Pulsamos el botón *Seleccionar ID* del cuadro de diálogo *Editor de códigos*.
14. En el cuadro de diálogo *Asignar un ID existente* tecleamos 5 y aceptamos.
15. Aceptamos el cuadro de diálogo *Editor de códigos*.
16. Volvemos a editar la última entidad para comprobar que ahora tiene el ID 5, por lo tanto las dos líneas están apuntando al mismo registro de base de datos.
17. Cambiamos el borde de izquierdo a derecho y aceptamos el cuadro de diálogo.
18. Ejecutamos la orden EDITAR\_COD esta vez sobre la primera entidad y comprobamos que se ha cambiado el borde que ahora tiene el valor Derecho.  
Vamos a arreglarlo haciendo que sea izquierdo y aceptando el cuadro de diálogo.
19. Ejecutamos la orden EDITAR\_COD sobre la línea nueva y pulsamos el botón *Nuevo ID*. Comprobamos que en la parte superior del cuadro de diálogo *Editor de códigos* el campo ID del códigos ya no es 5, sino SIN\_ASIGNAR.
20. Cambiamos el campo Borde de Izquierdo a Derecho.
21. Aceptamos el cuadro de diálogo.
22. Volvemos a editar la última línea para comprobar que el ID que tiene asignado no es 5.
23. Por último eliminamos la última entidad.

## Editando información existente: 2: Editor de la base de datos

Digi3D 2011 incorpora una nueva ventana denominada *Editor de la base de datos*, que podemos hacer visible en el menú *Base de datos/Ventanas/Editor de la base de datos* o con la combinación de teclas *Mayúsculas+Alt+E*.

Esta ventana en un principio aparece con sus controles deshabilitados, y no se habilitan hasta que el archivo de dibujo activo esté conectado con la base de datos.

| Id | Nombre                 | Borde |
|----|------------------------|-------|
| 42 | N-I                    | 1     |
| 43 | N-II                   | 1     |
| 44 | M-30                   | 2     |
| 46 | SE-30                  | 1     |
| 47 | N-I                    | 5     |
| 48 | CR-420                 | 0     |
| 49 | N-II                   | 2     |
| 51 | Carretera de canillas  | 9     |
| 53 | Avenida de Machu Pichu | 8     |
| 54 | N-IV                   | 1     |
| 56 | N-I                    | 9     |
| 57 | N-III                  | 1     |

Si el archivo de dibujo activo está conectado con la base de datos, podremos seleccionar en el cuadro de lista *Mostrar registros de la tabla* una tabla. Si seleccionamos alguna tabla, la ventana nos mostrará **una instantánea** de dicha tabla en el momento de la selección.

He marcado en negrita las palabras *una instantánea* porque eso es precisamente lo que vemos. La ventana le solicita al servidor de base de datos que le envíe la información en la tabla y la base de datos se la devuelve. La ventana se la muestra al usuario y a partir de ese momento se pierde toda conexión entre esta ventana y la base de datos, de modo que si algún usuario modifica algún registro de la base de datos, esta ventana no refleja esos cambios ya que el servidor no lo comunica.

Si estamos trabajando en un entorno multiusuario podemos forzar a regenerar la vista pulsando el botón *Regenera* de la barra de herramientas de la ventana *Editor de la base de datos*.

En la captura de pantalla anterior se puede ver la ventana *Editor de la base de datos* en acción.

Podemos ordenar la vista haciendo clic en las cabeceras de los datos de la parte inferior, de modo que si queremos ordenar por nombre, tan solo tenemos que hacer clic en el campo *Nombre*:

Editor de la base de datos

Mostrar registros de la tabla Vías de comunicación

Arrastre la cabecera de una columna aquí para agrupar por esa columna.

| Id | Nombre                 | Borde |
|----|------------------------|-------|
| 53 | Avenida de Machu Pichu | 8     |
| 51 | Carretera de canillas  | 9     |
| 48 | CR-420                 | 0     |
| 44 | M-30                   | 2     |
| 61 | M-30                   | 10    |
| 62 | M-45                   | 1     |
| 42 | N-I                    | 1     |
| 47 | N-I                    | 5     |
| 56 | N-I                    | 9     |
| 43 | N-II                   | 1     |
| 49 | N-II                   | 2     |
| 57 | N-III                  | 1     |

Además podemos cambiar a nuestro antojo el orden de los campos que queremos ver, tan sólo tenemos que arrastrar el campo a la posición que nos interese:

Editor de la base de datos

Mostrar registros de la tabla Vías de comunicación

Arrastre la cabecera de una columna aquí para agrupar por esa columna.

| Id | Nombre                 | Borde |
|----|------------------------|-------|
| 53 | Avenida de Machu Pichu | 8     |
| 51 | Carretera de canillas  | 9     |
| 48 | CR-420                 | 0     |
| 44 | M-30                   | 2     |
| 61 | M-30                   | 10    |
| 62 | M-45                   | 1     |
| 42 | N-I                    | 1     |
| 47 | N-I                    | 5     |
| 56 | N-I                    | 9     |
| 43 | N-II                   | 1     |
| 49 | N-II                   | 2     |
| 57 | N-III                  | 1     |

En esta captura se puede ver cómo estamos haciendo clic en la cabecera del campo Nombre, y estamos arrastrando ese campo hacia la izquierda. En el momento en que hemos desplazado el campo suficiente, aparecen unas flechas rojas que indican la posición donde se va a quedar el campo si soltamos. Una vez soltado, la ventana tiene el siguiente aspecto:

Editor de la base de datos

Mostrar registros de la tabla Vías de comunicación

Arrastre la cabecera de una columna aquí para agrupar por esa columna.

| Nombre                 | Id | Borde |
|------------------------|----|-------|
| Avenida de Machu Pichu | 53 | 8     |
| Carretera de canillas  | 51 | 9     |
| CR-420                 | 48 | 0     |
| M-30                   | 44 | 2     |
| M-30                   | 61 | 10    |
| M-45                   | 62 | 1     |
| N-I                    | 42 | 1     |
| N-I                    | 47 | 5     |
| N-I                    | 56 | 9     |
| N-II                   | 43 | 1     |
| N-II                   | 49 | 2     |
| N-III                  | 57 | 1     |

Podemos además quitar campos que no nos interesen arrastrando la cabecera del campo fuera de la ventana:

Editor de la base de datos

Mostrar registros de la tabla Vías de comunicación

Arrastre la cabecera de una columna aquí para agrupar por esa columna.

| Nombre                 | Id | Borde |
|------------------------|----|-------|
| Avenida de Machu Pichu | 53 | 8     |
| Carretera de canillas  | 51 | 9     |
| CR-420                 | 48 | 0     |
| M-30                   | 44 | 2     |
| M-30                   | 61 | 10    |
| M-45                   | 62 | 1     |
| N-I                    | 42 | 1     |
| N-I                    | 47 | 5     |
| N-I                    | 56 | 9     |
| N-II                   | 43 | 1     |
| N-II                   | 49 | 2     |
| N-III                  | 57 | 1     |

Aparecerá una cruz negra en la zona en la que si soltamos se elimina el campo, y si soltamos...

| Editor de la base de datos                                             |    |
|------------------------------------------------------------------------|----|
| Mostrar registros de la tabla Vías de comunicación                     |    |
| Arrastre la cabecera de una columna aquí para agrupar por esa columna. |    |
| Nombre                                                                 | Id |
| Avenida de Machu Pichu                                                 | 53 |
| Carretera de canillas                                                  | 51 |
| CR-420                                                                 | 48 |
| M-30                                                                   | 44 |
| M-30                                                                   | 61 |
| M-45                                                                   | 62 |
| N-I                                                                    | 42 |
| N-I                                                                    | 47 |
| N-I                                                                    | 56 |
| N-II                                                                   | 43 |
| N-II                                                                   | 49 |
| N-III                                                                  | 57 |

La ventana ya no muestra ese campo.

Si volvemos a seleccionar la misma tabla en el cuadro de lista *Mostrar registros de la tabla*, volverán a aparecer todos los campos.

Entre la barra de herramientas de la ventana y la parte inferior que es la que muestra los datos, disponemos de un área en la que podemos arrastrar las cabeceras de los campos para agrupar por ese campo.

Veamos qué pasa si arrastramos el campo Nombre a la zona *Arrastre la cabecera de una columna aquí para agrupar por esa columna...*

| Editor de la base de datos                         |       |
|----------------------------------------------------|-------|
| Mostrar registros de la tabla Vías de comunicación |       |
| Nombre                                             |       |
| Id                                                 | Borde |
| [ ] Nombre: N-I                                    |       |
| 42                                                 | 1     |
| 47                                                 | 5     |
| 56                                                 | 9     |
| [ ] Nombre: N-II                                   |       |
| 43                                                 | 1     |
| 49                                                 | 2     |
| [ ] Nombre: N-III                                  |       |
| 57                                                 | 1     |
| 58                                                 | 1     |

Como se puede observar, la ventana crea grupos en los que se muestran los registros que pertenecen a cada grupo. Así que podemos comprobar que existen tres geometrías que definen la Nacional I: la que tiene Id=42, 47, 56 con bordes 1, 5 y 9 respectivamente.

La ventana Editor de la base de datos cumple una doble función:

1. Nos permite editar la base de datos.

Los cambios que realicemos en esta ventana se verán reflejados inmediatamente en la base de datos.

2. Nos permite enviar las entidades enlazadas con los registros seleccionados a la orden que se esté ejecutando y que esté esperando que seleccionemos una entidad o grupo de entidades.

Veámoslo con un ejemplo: Si ejecutamos la orden ZOOM\_ENTIDAD, ésta orden nos solicita que seleccionemos la entidad a centrar en pantalla. Podemos seleccionarla manualmente o ejecutando cualquiera de las órdenes de selección (selección por índice, selecciona último, ...)

Si estamos conectados con la base de datos, y estamos visualizando en la ventana *Editor de la base de datos* una tabla y además tenemos seleccionado en esta ventana algún registro, podremos ejecutar la orden SELECCIONA\_EDITOR\_BBDD, que localizará las entidades que apuntan al registro/s seleccionado/s y se las enviarán a la orden ZOOM\_ENTIDAD, que realizará un zoom extendido que comprenderá el conjunto de entidades que cumplan la condición.

## 4. Acceso a base de datos en profundidad

En este capítulo se explica en profundidad el concepto de Modelo de datos, los nuevos parámetros de configuración en las tablas de códigos Digi.tab.xml, el proceso de conexión con la base de datos y cómo programar Microsoft SQL Server para enviar errores a Digi3D 2011 desde el servidor.

La inmensa mayoría de los laboratorios de este capítulo requieren que tengamos instalado el servidor de bases de datos *Microsoft SQL Server*.

Los laboratorios de este capítulo están ubicados en las carpetas Laboratorio 1, Laboratorio 2, Laboratorio 3, y así sucesivamente. Cada laboratorio dispone de su propia tabla de códigos *Laboratorio.tab.xml* y ese patrón se repite en todos los laboratorios, todas las carpetas tienen un archivo *Laboratorio.tab.xml*.

Podemos aprovecharnos de la característica *sustituidores* para evitar tener que seleccionar la tabla de códigos correspondiente a cada laboratorio, de modo que en el cuadro de diálogo de *Nuevo Proyecto*, en la pestaña *Archivo de dibujo*, en el parámetro *Tabla de códigos* en la sección *Entorno* yo personalmente pondría para todos los laboratorios de este capítulo la siguiente cadena:

`$(PathOfDrawingFile)Laboratorio.tab.xml`

Muchos de los laboratorios tienen un menú personalizado para realizar las tareas del propio laboratorio y en todos coincide que el menú está en el propio directorio del laboratorio y su nombre siempre es: *Laboratorio.menu.xml*.

Podemos aprovecharnos aquí también de los sustituidores y poner en el campo de menú la siguiente cadena:

`$(PathOfDrawingFile)Laboratorio.menu.xml`

## Modelos de Datos

Las versiones de Digi3D incluida Digi3D 2007 y Digi3D 2007 MGCP (o Digi3D 2010) estaban pensadas para almacenar datos en bases de datos *Microsoft Access*.

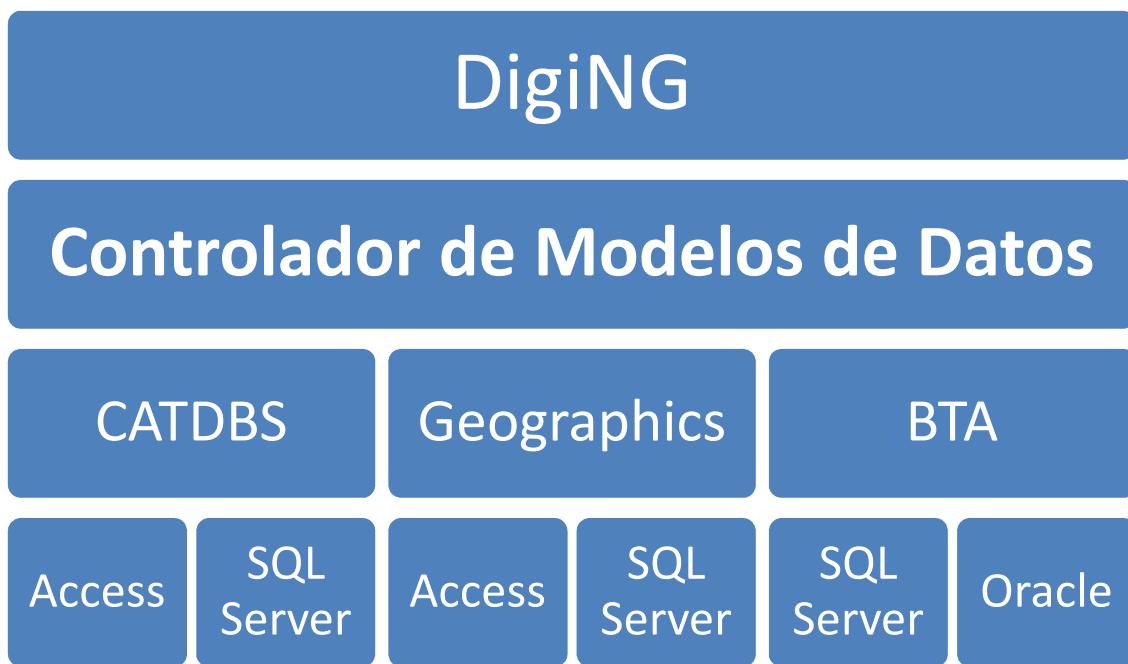
La funcionalidad de almacenar atributos en bases de datos estaba implementada en el propio programa de modo que era el propio Digi3D el encargado de conectarse con la base de datos, localizar tablas a partir de números de tabla consultando las tablas *CATDBS* o *MSCATALOG* y por último almacenar o cargar la información de las distintas tablas.

No se podía hacer mucho más sencillamente porque ni Digi3D ni *Microsoft Access* permitían hacer mucho más. Tenía que existir forzosamente una tabla *CATDBS* o *MSCATALOG* y además Digi3D asumía que tenía siempre control sobre la base de datos y que nadie más estaba conectado a esa base de datos.

En Digi3D 2011 se ha optado por separar la capa de comunicación con la base de datos y se ha creado el concepto de *Modelo de Datos*.

Digi3D 2011 no sabe ni conectarse con bases de datos ni almacenar ni recuperar información de una base de datos, delega esta responsabilidad en el *Modelo de datos* seleccionado para establecer la conexión con la base de datos.

El siguiente gráfico muestra la relación entre DigiNG y los distintos *modelos de datos*:



Para Digi3D 2011, un *Modelo de Datos* no es más que una extensión (que puede estar programada en cualquier lenguaje de programación .NET) que básicamente cumple cinco condiciones:

1. Es capaz de conectarse con un repositorio de datos (usualmente una base de datos, aunque podría ser perfectamente un archivo .txt o un servicio en la nube) mediante una cadena de conexión.

2. Es capaz de devolver el nombre de una tabla a partir de un número de tabla.
3. Es capaz de devolver el número de tabla a partir de su nombre.
4. Es capaz de recuperar información a partir de la dupla (número de tabla/número de registro)
5. Es capaz de almacenar información en una tabla, devolviendo el ID del registro creado.

Como Digi3D no implementa los modelos de datos, sino que esta implementación la realizan módulos cargables, se pueden añadir en un futuro nuevos *modelos de datos* sin necesidad de crear una nueva versión de Digi3D.

Digi3D 2011 viene de fábrica con los siguientes *Modelos de datos*:

| Nombre             | S.G.B.D. apropiado                                                                               | Descripción                                                                                             |
|--------------------|--------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------|
| <b>CATDBS</b>      | Microsoft SQL Server Express, Microsoft SQL Server, Oracle Express, Oracle, mySQL, Postgree, ... | Implementa el método clásico de las versiones Digi21-Digi3D 2007 con archivo de tabla índice CATDBS.    |
| <b>Geographics</b> | Microsoft SQL Server Express, Microsoft SQL Server, Oracle Express, Oracle, mySQL, Postgree, ... | Implementa el método clásico de las versiones Digi21-Digi3D 2007 con archivo de tabla índice MSCATALOG. |

## Cambios en Digi.tab.xml

Ya sabemos que en Digi3D 2011 podemos vincular un código con una tabla de base de datos, y esta vinculación se indica en la tabla de códigos.

Para poder realizar esto, se ha añadido un nuevo atributo *table* a los nodos *code* que indica con qué tabla está asociado un determinado código.

Este atributo no es obligatorio, por lo tanto en una tabla de códigos podemos tener códigos sin enlace a base de datos y códigos con enlace de datos. Los códigos que sí que tengan enlace a base de datos deberán estar asociados con una única tabla y esa tabla deberá existir en la base de datos lógicamente.

Veamos un ejemplo de un código con enlace a base de datos:

```
<code
 name="CAA010"
 description="Centroide de Mina de extracción Entidad de área"
 type="1"
 table="AAA010" >
 ...
</code>
```

Si la base de datos estuviera optimizada únicamente para la captura con Digi3D 2011, quizás con este atributo añadido a los códigos en las tablas de códigos hubiera sido suficiente para que Digi3D 2011 mostrase al usuario en la *Ventana de atributos de base de datos* los campos a llenar en la base de datos en el momento de finalizar una entidad.

Pero el mundo real no es tan sencillo, y es muy posible que se den alguna de las siguientes situaciones:

1. Quizás es posible que alguna de las tablas de la base datos tenga campos que el usuario no puede llenar sencillamente porque no se dispone de información en el momento de la captura. Un ejemplo sería una tabla con un campo denominado **FECHA\_DE\_FINALIZACIÓN**. En este tipo de situaciones habría que ocultar la visualización de este tipo de campos e indicarle a Digi3D que no almacene ningún valor en ese campo al crear el registro en la base de datos, de modo que se quedará con valor nulo.

Simularemos esta situación en el laboratorio 14.

2. También es posible para un determinado trabajo, todos los registros de un campo de la tabla deban tener un valor constante, como por ejemplo una tabla con un campo denominado *Contratista* en el que hay que almacenar el nombre de la compañía que está digitalizando la cartografía. Si la empresa en cuestión es *Acme, S.A.*, debería de haber alguna manera de indicarle a Digi3D 2011 que no muestre ese campo al operador (ya que no le aporta ninguna información importante) y que se almacene el valor constante *Acme, S.A.* en ese campo en todos los registros.

Simularemos esta situación en los laboratorios 15, 16 y 17.

3. También es posible que en la base de datos se almacene información que se deba obtener de forma dinámica a la hora de almacenar una entidad, como por ejemplo la longitud de la línea, su área, el número de vértices que tiene, o el nombre del ordenador desde el que se ha almacenado la entidad, o el nombre del operador, o la fecha y hora en la que se ha almacenado, ... En este tipo de situaciones habría que ocultar la visualización de este tipo de campos al usuario y ordenarle a Digi3D 2011 que calcule dinámicamente el valor del campo en el momento de crear el registro en la base de datos.

Simularemos esta situación en los laboratorios 18, 19 y 20.

4. Quizás tengamos una tabla con un campo numérico en el que únicamente se puedan almacenar una serie de valores constantes como por ejemplo un campo que indique el tipo de actividad deportiva en un recinto deportivo:

| Valor a almacenar | Significado |
|-------------------|-------------|
| 24537             | Fútbol      |
| 642               | Baloncesto  |
| 5                 | Voleibol    |
| 2247              | Vóley playa |
| 9087765           | Tenis       |

Esto es realmente muy difícil de memorizar. En este tipo de situaciones sería necesario poder indicarle a Digi3D 2011 que a pesar de que el campo es numérico, que oculte al usuario los números y que en su lugar le muestre una lista de posibles valores (Fútbol, Baloncesto...) y que de forma automática, si el usuario ha seleccionado Tenis en la lista, que se almacene el valor 9087765.

Simularemos esta situación en los laboratorios 21 y 22.

5. Exagerando un poco más el punto anterior, es posible (y a día de hoy al menos una empresa que yo conozca que tiene este tipo de situación) que por un mal diseño en la base de datos sea necesaria una determinada combinación de valores para indicar que una entidad pertenece a un determinado tipo.

Supongamos que tenemos una tabla con los siguientes campos y sus posibles valores

| Tipo de edificio | Subtipo de edificio         | SubSubtipo de edificio        |
|------------------|-----------------------------|-------------------------------|
| 1. Público       | 1. Para edificios públicos: | ...                           |
| 2. Privado       | 1.                          | ...                           |
|                  | Administració               | 20. Para Museos públicos:     |
|                  | n                           | 1. Historia                   |
|                  | 2. Policía                  | 2. Pintura                    |
|                  | 3. Museo                    | 3. ...                        |
|                  | 4. Religioso                | 21. Para Edificios religiosos |
|                  | 2. Para edificios           | 1. Iglesia                    |

|            |                      |
|------------|----------------------|
| privados:  | 2. Mezquita          |
| 1. Banco   | 3. Templo budista... |
| 2. Oficina |                      |

Con esta configuración, si el operador registra un museo de historia tendrá que almacenar en esta tabla un registro con los valores:

| Tipo de edificio | Subtipo | SubSubtipo de edificio |
|------------------|---------|------------------------|
| 1                | 3       | 1                      |

Y si con posterioridad registra un templo budista, tendrá que almacenar en la base de datos un registro con los valores:

| Tipo de edificio | Subtipo | SubSubtipo de edificio |
|------------------|---------|------------------------|
| 1                | 4       | 3                      |

Esto es realmente complicado para el usuario, y existen muchas posibilidades de que se equivoque a la hora de seleccionar la combinación correcta.

Debería de haber alguna manera de que se asignaran estos valores de forma automática, y que se indicase que se le mostraran esos valores al operador, pero de solo lectura, mediante algún botón en una barra de herramientas, o alguna opción en un menú o mediante un archivo de macroinstrucciones o mediante orden personalizada, de manera que el usuario indicase “Quiero registrar un templo budista” y se realizara este trabajo de forma automática y sin posibilidad de error.

Simularemos esta situación en el laboratorio 23.

6. Es posible que los campos en la base de datos tengan nombres tan raros que el operador se encuentre con la situación de *no saber qué poner ahí*. En este tipo de situaciones debería de haber alguna manera de mostrarle al usuario una pequeña ayuda cuando esté llenando ese campo.

Simularemos esta situación en el laboratorio 24.

7. Quizás nos interese que Digi3D 2011 cambie el nombre de los campos mostrados al usuario en la ventana de *Campos de la base de datos*. Podría suceder que los nombres de los campos en la base de datos estén en español y nos interese mostrárselos al usuario en inglés.

Simularemos esta situación en el laboratorio 25.

Existe una manera de hacer todo lo que acabo de enumerar en los puntos anteriores, y la manera es almacenar esa información en la tabla de códigos.

Las tablas de códigos de Digi3D 2011 incorporan una sección nueva<sup>7</sup> denominada: *databaseSchema*.

A continuación tienes el esquema XML de esta nueva sección en la tabla de códigos:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<xsschema attributeFormDefault="unqualified" elementFormDefault="qualified"
xmlns:xss="http://www.w3.org/2001/XMLSchema">
<xselement name="databaseSchema">
<xsccomplexType>
<xsssequence>
<xselement maxOccurs="unbounded" name="table">
<xsccomplexType>
<xsssequence>
<xselement maxOccurs="unbounded" name="field">
<xsccomplexType>
<xssattribute name="name" type="xs:string" use="required" />
<xssattribute name="title" type="xs:string" use="required" />
<xssattribute name="type" type="xs:string" use="required" />
<xssattribute name="length" type="xs:unsignedByte" use="required" />
<xssattribute name="bgColor" type="xs:string" use="required" />
<xssattribute name="allowZeroLength" type="xs:unsignedByte" use="optional" />
<xssattribute name="defaultValue" type="xs:string" use="optional" />
<xssattribute name="forceDefaultValue" type="xs:unsignedByte" use="required" />
<xssattribute name="description" type="xs:string" use="required" />
<xssattribute name="category" type="xs:string" use="optional" />
<xssattribute name="primaryKey" type="xs:unsignedByte" use="optional" />
<xssattribute name="readOnly" type="xs:unsignedByte" use="optional" />
<xssattribute name="valueList" type="xs:string" use="optional" />
<xssattribute name="visibility" type="xs:unsignedByte" use="optional" />
</xsccomplexType>
</xselement>
</xsssequence>
<xssattribute name="name" type="xs:string" use="required" />
</xsccomplexType>
</xselement>
</xsssequence>
</xsccomplexType>
</xselement>
</xsschema>
```

Si no entiendes de esquemas XML, lo anterior quiere decir que la sección *databaseSchema* está formada por una serie de nodos *table* que a su vez tienen una serie de nodos *field* que

---

<sup>7</sup> Esta es una de las grandes de XML, podemos crear una tabla de códigos con la sección *databaseSechema* y cargarla en Digi3D 2007 sin problemas porque el que tenga más o menos nodos no afecta al a carga del XML. La versión 2007 sencillamente ignorará la sección *databaseSechema*.

representan los campos de la tabla. El nodo *table* tiene un atributo denominado *name* donde se almacena el nombre de la tabla cuyo esquema se está especificando y los nodos *field* tienen una serie de atributos como *name*, *title*, *type*, *length*, ... que nos van a servir para solucionar todos los problemas que he enumerado anteriormente.

Aquí tienes un recorte de una tabla de códigos que tiene en su esquema de base de datos el de una tabla denominada *Viales* en la que se enumeran dos campos, *IDVial* que es el campo de tipo *clave principal* e invisible para el usuario y un campo *Nombre* donde el usuario tendrá que teclear el nombre del vial.

```
...
<databaseSchema>
 <table name="Viales">
 <field name="IDVial" type="N" primaryKey="1" visibility="0" />
 <field name=" Nombre" type="N"/>
 </table>
</databaseSchema>
...
```

No vamos a entrar en detalle ahora sobre el significado de cada uno de estos atributos. Por ahora tienes que saber que nuestra tabla de códigos debe tener almacenado el esquema de la base de datos con la que pretendes conectar, pero no te asustes, existen herramientas como el programa Digi.Tab que realizan esta tarea de forma automática.

Además, no se tienen por qué enumerar todos los campos de la base de datos, únicamente tendremos que enumerar los que queramos que sean visibles para el usuario, los que requieran valores por defecto (constantes o dinámicos) y el que hace de clave primaria.

Los que no estén enumerados en el esquema o los que estén marcados como ocultos, sencillamente no aparecerán en la *Ventana Campos de la base de datos*.

## Conexión con la base de datos en profundidad

Si el usuario selecciona un *Modelo de datos* en la pestaña *Archivo de dibujo* del cuadro de diálogo de *Nuevo Proyecto* o en el cuadro de diálogo común que se muestra al cargar un archivo de referencia, cuando se finaliza con la carga del archivo de dibujo, Digi3D 2011 comienza con el proceso de conexión con la base de datos.

Este proceso está dividido en varias fases como se puede ver en la tabla de a continuación:

### Creación del objeto Modelo de Datos

### Conexión con la Base de Datos

### Comprobación de compatibilidad Base de Datos <-> Modelo de Datos

### Comprobación de compatibilidad Digi.Tab <-> Modelo de Datos

#### Creación del objeto Modelo de Datos

Digi3D 2011 intenta localizar la extensión que implementa el *modelo de datos* que ha seleccionado el usuario.

Si no se consigue cargar la extensión, Digi3D 2011 mostrará en la ventana de resultados una descripción del error y en la ventana de tareas una tarea que al hacer doble clic muestra información detallada del error.

#### Laboratorio 1 (Creo que voy a eliminar este laboratorio)

~~En este laboratorio vamos a forzar a que Digi3D nos muestre un error porque el usuario ha seleccionado un *Modelo de Datos* con problemas.~~

Para ello, instalaremos un componente denominado *Modelo de datos Incorrecto* que nos permitirá seleccionar un modelo de datos que no se puede crear, para comprobar cómo comunica el problema Digi3D.

Este componente básicamente le indica a Digi3D que le permita al usuario seleccionar como modelo de datos uno denominado *Modelo de datos incorrecto* que es un modelo de datos que no se puede cargar. De esta manera, simulamos errores que se podrían dar en una situación real como que el componente ha sido desinstalado, o que le falta algún archivo de configuración, o que no se puede ejecutar porque el usuario no tiene permisos, o porque está ubicado en un servidor remoto al cual no se tiene acceso temporalmente...

1. Comprobamos si está instalado en el equipo el componente *Modelo de Datos Incorrecto Laboratorio BBDD Digi3D 2011*. Para ello, pulsamos el botón de Windows/Panel de control/Desinstalar un programa. En caso de que ya esté instalado, pasamos al punto 4.
2. Descargamos el componente de la dirección  
<http://download.dig3d.net/ModeloDatosIncorrectoLaboratorioBBDDDig3D2011.msi>
3. Instalamos el componente haciendo doble clic.
4. Arrancamos Digi3D 2011 y seleccionamos el menú Ayuda/Acerca de Digi3D...
5. Comprobamos que en el cuadro de diálogo de Acerca de Digi3D aparece el componente titulado *Modelo de Datos Incorrecto* en la ventana *Productos Instalados*. Si aparece en esta lista quiere decir que Digi3D lo ha cargado correctamente en su proceso de inicio y que lo podemos utilizar.
6. Cerramos el cuadro de diálogo de Acerca de Digi3D y cargamos el archivo [ruta al laboratorio de base de datos]\Laboratorio 1\Modelo.bin
7. En la sección titulada *Base de Datos*, en la opción *Modelo de Datos* seleccionamos el modelo titulado *Modelo de datos incorrecto*.
8. En el campo *Tabla de códigos* seleccionamos el archivo [ruta al laboratorio1]\Laboratorio1.tab.xml
9. Pulsamos el botón OK para que Digi3D cargue el archivo de dibujo y que intente establecer conexión con el *Modelo de Datos de Base de Datos* seleccionado.
10. Comprobamos que Digi3D 2011 ha comunicado el error en la ventana de resultados. Si esta está cerrada, podemos abrirla mediante el menú Ver/Ventana de resultados.
11. Comprobamos que Digi3D 2011 ha comunicado el error en la ventana de tareas. Si esta está cerrada, podemos abrirla mediante el menú Ver/Ventana de Tareas.
12. Hacemos doble clic en el error mostrado en la tarea y comprobamos que Digi3D muestra un cuadro de diálogo indicando el error.
13. Desplegamos el botón titulado Más información para comprobar la información extendida.
14. Cerramos la tarea.
15. Cerramos Digi3D.

## Conexión con la base de datos

Una vez creado el Modelo de Datos de Base de Datos, Digi3D 2011 le ordena a este que se conecte con la base de datos mediante la cadena de conexión especificada por el usuario.

El formato de cadena de conexión varía en función de la tecnología con la que se ha programado el Modelo de Datos de Base de Datos y en función del servidor de Base de Datos al que se va a conectar. Tendremos que referirnos a la documentación del Modelo de datos para más información.

El formato para los Sistemas Gestores de Base de Datos CATDBS y Geographics para bases de datos locales en formato Microsoft Access es el siguiente:

```
Provider=Microsoft.Jet.OLEDB.4.0;Data Source=[ruta al archivo].mdb
```

Ejemplo:

```
Provider=Microsoft.Jet.OLEDB.4.0;Data Source=c:\Bases de Datos\Proyecto1.mdb
```

El formato para el Modelo de Datos CATDBS y Geographics para bases de datos Microsoft SQL Server es el siguiente:

```
Provider=SQLOLEDB.1;Integrated Security=SSPI;Persist Security Info=False;Initial Catalog=[nombre de la base de datos];Data Source=[nombre de la instancia SQL Server]
```

Por ejemplo:

```
Provider=SQLOLEDB.1;Integrated Security=SSPI;Persist Security Info=False;Initial Catalog=laboratorio4;Data Source=.\SQLEXPRESS
```

Si se produce algún error a la hora de establecer la conexión con la base de datos, el Modelo de Datos de Base de Datos se lo comunicará a Digi3D que a su vez mostrará un cuadro de diálogo indicando de la condición de error, así como una descripción del error en la ventana de resultados y una tarea que al hacer doble clic muestra información detallada del error.

## Laboratorio 2 (Base de datos desconocida)

En este laboratorio vamos a forzar a que Digi3D nos muestre un error porque el usuario ha indicado como cadena de conexión una ruta a una base de datos incorrecta.

1. Arrancamos Digi3D 2011
2. Cargamos el archivo [ruta al laboratorio de base de datos]\Laboratorio 2/Modelo.bin
3. En la sección titulada *Base de Datos*, en la opción *Modelo de Datos* seleccionamos CATDBS.
4. En el campo titulado *Cadena de Conexión* tecleamos como ruta al a base de datos una ruta incorrecta, por ejemplo, suponiendo que el equipo donde se está ejecutando el laboratorio no dispone de unidad R, tecleamos:  

```
Provider=Microsoft.Jet.OLEDB.4.0;Data Source=R:\Proyecto1.mdb
```
5. Pulsamos el botón OK para que Digi3D cargue el archivo de dibujo y que intente establecer conexión con el *modelo de datos*.

6. Comprobamos que Digi3D 2011 nos muestra un cuadro de diálogo indicando que se ha localizado un error al conectar con la base de datos. Si pulsamos el botón titulado *Más información* Digi3D 2011 nos muestra la cadena de error que ha devuelto el proveedor de base de datos.
7. Aceptamos el mensaje de error y comprobamos que Digi3D 2011 nos muestra el archivo de dibujo correctamente, pero no tendremos acceso a base de datos.
8. Si activamos la *Ventana de resultados* podremos comprobar que Digi3D 2011 muestra una descripción del error.
9. Si activamos la *Ventana de tareas* podremos comprobar que Digi3D 2011 muestra una tarea con el error. Si hacemos doble clic sobre la tarea, Digi3D mostrará un cuadro de diálogo como el explicado en el punto 8 de este laboratorio.

### Comprobación de compatibilidad Base de Datos <-> Modelo de Datos

Una vez establecida correctamente la conexión con la base de datos, se comprueba que ésta sea compatible con el modelo de datos seleccionado por el usuario.

Cada modelo de datos define su propio esquema de compatibilidad, de modo que tendremos que estudiar la documentación de cada modelo de datos a la hora de crear bases de datos para dicho modelo de datos.

Cuando Digi3D le ordena al Modelo de Datos que compruebe si la base de datos es compatible, es posible que se produzcan tanto errores como advertencias.

Si se localiza algún error o alguna advertencia, Digi3D 2011 mostrará en la ventana de resultados una descripción del error/advertencia y en la ventana de tareas una tarea que al hacer doble clic muestra información detallada del error/advertencia.

El Modelo de Datos CATDBS impone que la base de datos tenga una tabla denominada CATDBS.

Esta tabla se utiliza para indicarle a Digi3D el nombre de una tabla asociado con un número de tabla.

Si recordamos la estructura de un código en Digi3D 2011:

| Código<br>(alfanumérico) | Tabla (numérico)                                                                                                                                                                                                                                                                | Id (numérico)                                                                                                                                                                                                                              |
|--------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| • Nombre del código.     | <ul style="list-style-type: none"> <li>0 Para indicar que el código está eliminado.</li> <li>1 Para indicar que este código no tiene enlace con una base de datos externa.</li> <li>&gt;1 Para indicar que el enlace de base de datos está en una determinada tabla.</li> </ul> | <ul style="list-style-type: none"> <li>Valor que obtiene la clave principal (campo Primary Key, de una tabla, campo registro, campo mslink, ...) en la tabla indicada en el campo Tabla del código si el campo Tabla es &gt; 1.</li> </ul> |

Comprobamos que si una entidad tiene un enlace de base de datos, dicho enlace estará formado por un número de tabla (valor almacenado en el campo *Tabla* del código) y un número que indica qué entrada de entre todas las que tiene esa tabla contiene la información de base de datos para la entidad (valor almacenado en el campo *Id* del código).

Si Digi3D 2011 localiza por ejemplo una entidad con el siguiente código:

| Código | Tabla | Id    |
|--------|-------|-------|
| 020400 | 12    | 12367 |

entenderá que la información de base de datos para esa entidad está almacenada en el registro cuya clave principal tenga el valor 12367 de la tabla número 12, pero ¿qué tabla es la tabla 12?

Digi3D 2011 no sabe transformar el número 12 en un nombre de tabla, delega esa tarea al Modelo de Datos con el que nos hemos conectado a la base de datos.

Si el modelo de datos es CATDBS, este localizará ese nombre de tabla en la tabla con nombre CATDBS que deberá tener obligatoriamente la base de datos (si la base de datos no dispone de esa tabla, no se considerará válida para conectarse con ella mediante el modelo de datos CATDBS).

La tabla CATDBS debe tener al menos dos campos:

- Campo índice, donde se almacena el número de la tabla, que debe ser obligatoriamente de tipo numérico, sin permitir nulos y que además sea el campo *clave primaria* de la tabla.
- Campo donde se almacenan los nombres de las tablas, que debe ser obligatoriamente de tipo alfanumérico.

El nombre de estos dos campos no es importante, lo importante es que el primero sea numérico y el segundo alfanumérico.

Además se impone que exista en esta tabla un registro que indique que el nombre de tabla para el número de tabla 1 sea DBDIGI.

La tabla CATDBS puede tener más campos, pero el modelo de datos CATDBS no los utiliza. Se puede añadir por ejemplo un campo de tipo descripción para describir la información que se almacena en una determinada tabla.

Ejemplo de una tabla CATDBS:

| ID | Nombre Tabla | Descripción            |
|----|--------------|------------------------|
| 1  | DBDIGI       | Tabla interna de Digi  |
| 2  | AAA010       | Minería de Extracción  |
| 3  | AAA012       | Cantera                |
| 4  | AAA054       | Campo de hidrocarburos |
| 12 | AAH050       | Fortificación          |

Cuando Digi3D 2011 le ordena al modelo de datos CATDBS que compruebe la compatibilidad con una base de datos, este comprueba si la base de datos dispone de esta tabla y cumple las restricciones de que su primer campo debe ser numérico,... Si no se localiza la tabla, los campos no cumplen con los requisitos o no se localiza el registro con la dupla de valores (1, DBDIGI), el modelo de datos le devuelve a Digi3D un error que Digi3D se encarga de comunicar al usuario.

Si se pasa la prueba de la tabla CATDBS, el modelo de datos comprueba si existen las tablas enumeradas en esa tabla. El modelo de datos no exige que existan esas tablas en la base de datos, por lo tanto si no localiza alguna de estas tablas en la base de datos, no creará un error, sino que creará una advertencia.

De modo que para la tabla anterior, el modelo de datos comprobará si existe en la base de datos una tabla denominada AAA010, otra denominada AAA012 y así sucesivamente.

Si no se encuentra la tabla AAA054, Digi3D mostrará una advertencia.

Para las tablas que sí que existen, el modelo de datos CATDBS debe realizar una comprobación más: que todas ellas tengan un campo marcado como *clave primaria* y que este sea de tipo numérico.

Si se localiza alguna tabla sin clave primaria o con clave primaria pero con un tipo no numérico, el modelo de datos CATDBS informará con un mensaje de error que Digi3D se encargará de comunicar al usuario.

Si se pasa esta prueba, se continúa con el siguiente paso, comprobación de compatibilidad entre la tabla de códigos Digi.Tab.xml y el modelo de datos.

### Laboratorio 3 (Creando una base de datos no compatible con el modelo CATDBS)

En este laboratorio vamos a crear con *Microsoft SQL Server Express* una base de datos que no cumple los requisitos para conectarse con el modelo de datos CATDBS para comprobar cómo

comunica Digi3D el error indicando que la base de datos no es compatible con el modelo CATDBS.

1. Ejecutamos el programa *Microsoft SQL Server Management Studio*. Este programa nos sirve de interfaz con el servidor bases de datos Microsoft SQL Server. Los servidores no tienen interfaz de usuario, son servicios, no tienen ventanas, ni cuadros de diálogo. La única forma de comunicarse con ellos es con programas como SQL Server Management Studio.  
SQL Server Management Studio nos permite conectarnos con servidores (locales y remotos) crear tablas, usuarios, consultas, procedimientos almacenados, funciones, y todos los aspectos de un servidor de bases de datos.
2. En el cuadro de diálogo *Connect to Server* seleccionamos en el cuadro de opciones *Server Type* la opción *Database Engine* para indicar que nos queremos conectar con un servidor de bases de datos y no con un archivo local.
3. En el campo *Server Name* tecleamos: `.\SQLEXPRESS`. Con ello hemos indicado al programa SQL Server Management Studio que queremos conectarnos con la instancia denominada SQLEXPRESS de la máquina local. Los nombres de las instancias de servidores se eligen al instalar el servidor de base de datos, y el nombre por defecto para *SQL Server Express* es *SQLEXPRESS*.  
Si nos quisiéramos conectar contra la instancia SQLEXPRESS del ordenador Mercurio, seleccionaríamos `\Mercurio\SQLEXPRESS`.
4. En el campo *Authentication* seleccionamos *Windows Authentication*.
5. Pulsamos el botón *Connect*.
6. El programa *SQL Server Management Studio* se conectará con el servidor de bases de datos y nos mostrará en la ventana acoplable *Object Explorer* (a la izquierda por defecto) un árbol con los diferentes objetos que dispone el servidor de bases de datos.
7. Desplegamos la rama *Databases* para comprobar las bases de datos publicadas en nuestro servidor de bases de datos.
8. Si existe la base de datos titulada *CompatibilidadCATDBS*, la eliminamos haciendo clic sobre su nombre con el botón derecho del ratón, y seleccionando la opción *Delete* en el menú contextual que aparece. Confirmamos pulsando *OK* en el cuadro de diálogo que nos muestra para confirmar la eliminación de la base de datos.
9. Creamos una base de datos nueva haciendo clic con el botón derecho del ratón sobre la rama *Databases* y pulsando la opción *New Database...* del menú contextual.
10. En el cuadro de diálogo *New Database* le damos a la base de datos el nombre *CompatibilidadCATDBS* en el campo *Database Name*.
11. Pulsamos la opción *OK* y comprobamos que el servidor de bases de datos ha creado la base de datos porque ésta aparece en la rama *Databases*.
12. Comprobamos que la base de datos que acabamos de crear no tiene aún ninguna tabla, para ello desplegamos la rama *CompatibilidadCATDBS/Tables* y comprobamos que únicamente tiene *System Tables*, que son las tablas ocultas que necesita el servidor de base de datos por cada base de datos para realizar sus tareas de indexación, mantenimiento, permisos, ...
13. Ejecutamos Digi3D y abrimos el cuadro de diálogo *Nuevo Proyecto* con la combinación de teclas *Ctrl+N* o el menú Archivo/Nuevo, abrir archivo de proyecto..., seleccionamos la pestaña *Archivo de Dibujo*

14. Seleccionamos el archivo [ruta al laboratorio de base de datos]\Laboratorio3\Modelo.bin en el campo *Archivo de dibujo*.
15. Seleccionamos el modelo de datos CATDBS.
16. En cadena de conexión, pulsamos el botón de los tres puntos, en *Provider* seleccionamos *Microsoft OLE DB Provider for SQL Server* y pulsamos *Next.'*
17. En la pestaña *Connection* en el campo *Select or enter a Server Name*, tecleamos .\SQLEXPRESS o desplegamos el cuadro de opciones (si desplegamos el campo de opciones posiblemente tengamos que esperar mucho tiempo, ya que en ese momento se inicia el proceso de localizar instancias de Microsoft SQL Server por toda la red) así que si sabemos el nombre de la instancia a la que nos queremos conectar, en este caso .\SQLEXPRESS, pues será mejor escribirla que esperar a que el cuadro de diálogo localice todas las instancias visibles en nuestra red.
18. En el campo *Enter Information to log on to the server* seleccionamos *Use Windows NT Integrated security* que es el modelo más seguro de seguridad y que es además el que se configura por defecto cuando instalamos Microsoft SQL Server Express.
19. En el campo *Select the database on the server*, desplegamos el cuadro de opciones y seleccionamos la base de datos *CompatibilidadCATDBS*.
20. Pulsamos el botón *Test connection* para comprobar que la conexión se puede establecer correctamente y aceptamos el cuadro de mensajes que nos indica que se ha podido establecer la conexión sin problemas.
21. Por último aceptamos el cuadro de diálogo *Data Link Properties*.
22. Comprobamos que el campo *Cadena de conexión* se ha llenado automáticamente con la cadena: Provider=SQLOLEDB.1;Integrated Security=SSPI;Persist Security Info=False;Initial Catalog=CompatibilidadCATDBS;Data Source=.\SQLEXPRESS
23. Podríamos haber tecleado directamente esta cadena de conexión en vez de pulsar el botón de los tres puntos en el punto 18 de este laboratorio y nos habríamos ahorrado pasar del punto 18 a 24.
24. Pulsamos *OK* en el cuadro de diálogo *Nuevo Proyecto* y comprobamos una vez cargado el archivo de dibujo que Digi3D 2011 nos muestra un cuadro de diálogo indicándonos que no se ha podido conectar correctamente con la base de datos porque ésta no cumple con los requisitos mínimos para el modelo de datos CATDBS ya que no dispone de una tabla denominada CATDBS. Además Digi3D muestra la misma información en la Ventana de resultados, que podemos hacer visible mediante el menú Ver/Ventana de resultados y además añade una tarea con información del error en la ventana de tareas que podemos hacer visible mediante el menú Ver/Ventana de tareas.

#### Laboratorio 4 (Creando una base de datos compatible con el modelo CATDBS)

En este laboratorio vamos a modificar nuestra base de datos *CompatibilidadCATDBS* para que sea compatible con el modelo de datos CATDBS.

1. Ejecutamos el programa *Microsoft SQL Server Management Studio*.

2. En el cuadro de diálogo *Connect to Server* seleccionamos en el cuadro de opciones *Server Type* la opción *Database Engine* para indicar que nos queremos conectar con un servidor de bases de datos.
3. En el campo *Server Name* tecleamos: .\SQLEXPRESS.
4. En el campo *Authentication* seleccionamos *Windows Authentication*.
5. Pulsamos el botón *Connect*.
6. Abrimos la rama *Databases*.
7. Abrimos la rama que representa nuestra base de datos y pulsamos con el botón derecho del ratón sobre su rama *Tables* y seleccionamos la opción *New Table...* en el menú contextual.
8. *Microsoft SQL Server Management Studio* nos mostrará una ventana en la que podemos añadir campos a la tabla que queremos crear.
9. En *Column Name* tecleamos *ID*, pulsamos la tecla flecha derecha del teclado tecleamos *int* (o desplegamos el cuadro de opciones y seleccionamos *int*) y deshabilitamos la opción *Allow Nulls*. Por último, pulsamos el botón derecho del ratón sobre el triángulo negro que aparece a la izquierda del campo donde hemos tecleado *NULL* y en el menú contextual seleccionamos *Set Primary Key*. Comprobaremos que el campo *NULL* es la clave primaria porque ahora tiene asociado un icono con una llave amarilla.
10. Pasamos al siguiente campo y tecleamos como nombre de campo *Tabla*, de tipo *varchar(32)* y deshabilitamos la opción *Allow Nulls*. El tipo *varchar([tamaño])* permite almacenar cadenas con tamaño variable, y eso quiere decir que si almacenamos en un campo de tipo *varchar(32)* la cadena “*hola*” ésta tendrá únicamente cuatro caracteres, sin embargo si almacenamos la cadena “*hola*” en un campo de tipo *char(32)* ésta tendrá 32 caractéres, *hola* + 28 espacios en blanco.
11. Por último añadimos un nuevo campo denominado *Descripción* de tipo *varchar(256)* permitiendo Nulls.
12. En la ventana acoplable *Properties* (que por defecto aparece acoplada en la parte derecha) en el campo *Name* cambiamos el nombre propuesto por el programa (en mi caso *Table\_1*) por *CATDBS*.
13. Para finalizar cerramos la ventana donde hemos tecleado los nombres de los campos que tendrá la tabla *CATDBS*.
14. *Microsoft SQL Server Management Studio* nos mostrará un cuadro de diálogo preguntando si guardar los cambios en la base de datos. Seleccionamos *Yes*.
15. Si ahora desplegamos la rama *Tables* de la base de datos *CompatibilidadCATDBS* comprobaremos que esta base de datos ya sí que tiene la tabla *CATDBS*.
16. Vamos a añadir registros en esta tabla pulsando con el botón derecho del ratón sobre la tabla *CATDBS* y seleccionando la opción *Edit Top 200 Rows* del menú contextual.
17. Comprobaremos que *Microsoft SQL Server Management Studio* muestra una vista editable con el contenido de la tabla. Como nuestra tabla está recién creada no tiene información aún.
18. Añadimos tres registros con la información de la tabla de a continuación:

| ID | ID | Tabla | Descripción |
|----|----|-------|-------------|
|----|----|-------|-------------|

|          |   |        |                                                                                                   |
|----------|---|--------|---------------------------------------------------------------------------------------------------|
| <b>1</b> | 1 | DBDIGI | Número de tabla utilizado por Digi3D para indicar que la entidad no tiene enlace de base de datos |
|----------|---|--------|---------------------------------------------------------------------------------------------------|

19. Nos desconectamos del servidor de base de datos pulsando el botón *Disconnect* en la barra de herramientas del panel *Object Explorer*.
20. Arrancamos Digi3D 2011 y cargamos el archivo de dibujo [ruta al laboratorio de base de datos]\Laboratorio 4\Modelo.bin, seleccionando CATDBS como modelo de datos y como cadena de conexión: Provider=SQLOLEDB.1;Integrated Security=SSPI;Persist Security Info=False;Initial Catalog=CompatibilidadCATDBS;Data Source=.\SQLEXPRES
21. Cargamos el modelo y comprobamos que Digi3D realiza la conexión sin ningún problema.

## Laboratorio 5 (Múltiples usuarios accediendo a un servidos monousuario)

En este laboratorio vamos a comprobar a ver en acción el comportamiento del servidor de bases de datos cuando múltiples usuarios intentan conectarse con una base de datos marcada como monousuario.

Para realizar este laboratorio es imprescindible haber finalizado correctamente el laboratorio número 4.

1. Arrancamos *Microsoft SQL Server Management Studio*, nos conectamos contra la instancia SQLEXPRESS local y desplegamos la rama de Bases de datos localizar la base de datos *CompatibilidadCATDBS*.
2. Si el ícono se la base de datos tiene una persona en miniatura, el IDE nos estará indicando que la base de datos es mono-usuario. Además, a la derecha del nombre de la base de datos pondrá entre paréntesis (*Single User*).
3. Pulsamos sobre el nombre de nuestra base de datos *CompatibilidadCATDBS* con el botón derecho del ratón y seleccionaremos *Properties* en el menú contextual luego *Options* en el cuadro de diálogo *Database Properties* y en *State/Restrict Access* selecciónamos **SINGLE\_USER**, aceptamos y aceptamos. Acabamos de convertir nuestra base de datos en una base de datos monousuario.
4. *Cerramos Microsoft SQL Server Management Studio*.
5. Arrancamos Digi3D 2011 y cargamos el archivo de dibujo [ruta al laboratorio de base de datos]\Laboratorio 5\Modelo1.bin, seleccionando como modelo de datos CATDBS y con la cadena de conexión Provider=SQLOLEDB.1;Integrated Security=SSPI;Persist Security Info=False;Initial Catalog=CompatibilidadCATDBS;Data Source=.\SQLEXPRES
6. Comprobamos que Digi3D 2011 nos muestra el modelo correctamente.

7. Ejecutamos otro Digi3D 2011 (sin cerrar el anterior), y abrimos el archivo [ruta al laboratorio de base de datos]\Laboratorio 5\Modelo2.bin.
8. Cuando esta segunda instancia de Digi3D 2011 intenta conectarse con la base de datos, el servidor de bases de datos le devuelve un error a Digi3D indicándole que no le permite conectarse con ésta porque ya hay una conexión establecida. Digi3D 2011 mostrará un cuadro de diálogo informando del error y añadirá un mensaje en la *Ventana de resultados* y añadirá una tarea en la *Ventana de Tareas*.

#### Laboratorio 6 (Creando una tabla en la base de datos sin clave primaria)

En este laboratorio vamos a añadir una tabla para almacenar información de viales. Nos interesa saber el identificador del vial (que será la clave principal de la tabla) y su nombre, pero lo vamos a hacer mal, no vamos a marcar ningún campo de la tabla Viales como clave principal, por lo tanto el modelo de datos CATDBS generará un error al comprobar la compatibilidad de la base de datos y comprobaremos que Digi3D 2011 mostrará mediante cuadros de diálogo, mensajes en la ventana de resultados y tareas en la ventana de tareas.

1. Ejecutamos *Microsoft SQL Server Management Studio*, y nos conectamos con nuestro servidor de bases de datos.
2. Buscamos la tabla CATDBS (en la base de datos CompatibilidadCATDBS) y añadimos una nueva entrada (pulsando sobre el nombre de la tabla con el botón derecho del ratón y seleccionando la opción *Edit Top 200 Rows*) indicando que la tabla 2 es la tabla Viales y añadimos un comentario. En mis caso, la tabla CATDBS tiene el siguiente aspecto:

| ID | ID | Tabla  | Descripción                                                                                       |
|----|----|--------|---------------------------------------------------------------------------------------------------|
| 1  | 1  | DBDIGI | Número de tabla utilizado por Digi3D para indicar que la entidad no tiene enlace de base de datos |
| 2  | 2  | Viales | Tabla en la que se almacena la información de un vial.                                            |

3. Creamos ahora la tabla Viales pulsando con el botón derecho del ratón sobre la rama *Tables* de nuestra base de datos y seleccionando la opción *New Table...*
4. Añadimos la siguiente información:

| Column name | Data Type    | Allow Nulls | Primary Key |
|-------------|--------------|-------------|-------------|
| IDVial      | int          | No          | No          |
| Nombre      | varchar(256) | No          | No          |

5. Cerramos la ventana donde acabamos de llenar toda esta información.
6. Como no hemos indicado en ningún momento el nombre de la tabla (podríamos haberlo cambiado en el panel acoplable *Properties*, a la derecha, pero esta vez no lo hemos hecho intencionadamente, *Microsoft SQL Server Management Studio* nos preguntará su nombre mediante un cuadro de diálogo. Tecleamos *Viales* y pulsamos *OK*.
7. Comprobamos que nuestra base de datos ya tiene dos tablas, CATDBS y Viales.
9. Arrancamos Digi3D 2011. Cargamos el archivo [ruta al laboratorio de base de datos]\Laboratorio 6\Modelo.bin, seleccionando como modelo de datos CATDBS y con la cadena de conexión Provider=SQLOLEDB.1;Integrated Security=SSPI;Persist Security Info=False;Initial Catalog=CompatibilidadCATDBS;Data Source=.\SQLEXPRESS
8. Aceptamos el cuadro de diálogo de nuevo proyecto y comprobamos que Digi3D 2011 no se puede conectar con la base de datos porque ésta no cumple con el protocolo impuesto por el modelo de datos CATDBS, ya que la tabla Viales no tiene ningún campo marcado como clave principal.

#### Laboratorio 7 (Tabla enumerada en el catálogo pero inexistente en la base de datos)

En este laboratorio vamos a arreglar el problema de la tabla Viales del laboratorio anterior además añadiremos a la tabla CATDBS una entrada adicional indicando que la tabla número 3 es la tabla Propietarios, sin embargo, no vamos a crear la tabla Propietarios para comprobar que el modelo de datos CATDBS no considera esta circunstancia como error, ya que no será ningún error mientras no se localice alguna entidad con un código enlazando con la tabla número 3. Si se localizara alguna entidad enlazada con la tabla 3 se podrá mostrar un mensaje de error, pero mientras no se localice ninguna, no se considera como error una tabla inexistente en la base de datos.

1. Ejecutamos *Microsoft SQL Management Studio*, nos conectamos con nuestro servidor de bases de datos local, editamos la tabla CATDBS para añadirle un registro que indique que la tabla 3 es la tabla Propietarios.

Mi tabla CATDBS tiene ahora el siguiente aspecto:

| ID | ID | Tabla        | Descripción                                                                                       |
|----|----|--------------|---------------------------------------------------------------------------------------------------|
| 1  | 1  | DBDIGI       | Número de tabla utilizado por Digi3D para indicar que la entidad no tiene enlace de base de datos |
| 2  | 2  | Viales       | Tabla en la que se almacena la información de un vial.                                            |
| 3  | 3  | Propietarios | Tabla propietarios creada para generar una advertencia en                                         |

## Digi3D

2. Editamos el diseño de la tabla Viales (pulsando con el botón derecho sobre su nombre y seleccionando *Design* en el menú contextual y hacemos que el campo IDVial sea de tipo clave principal (botón derecho sobre el ícono a la izquierda de su nombre y seleccionando *Set Primary Key* en el menú contextual)).
3. Cerramos *Microsoft SQL Management Studio*.
10. Ejecutamos Digi3D, cargamos el archivo [ruta al laboratorio de base de datos]\Laboratorio 7\Modelo.bin, seleccionando como modelo de datos CATDBS y con la cadena de conexión Provider=SQLOLEDB.1;Integrated Security=SSPI;Persist Security Info=False;Initial Catalog=CompatibilidadCATDBS;Data Source=.\\SQLEXPRES
4. Comprobamos que Digi3D carga correctamente el archivo de dibujo y no nos muestra ningún mensaje de error, sin embargo, mostrará una advertencia en la ventana de resultados y un mensaje de tipo advertencia en la ventana de tareas indicando que no se ha localizado la tabla Propietarios.

**Comprobación de compatibilidad Digi.Tab <-> Modelo de Datos**

Una vez finalizado el proceso de comprobación de compatibilidad Base de datos <-> Modelo de datos, comienza el último proceso de comprobación de compatibilidad, que es el que comprueba que la tabla de códigos utilizada en el momento de la carga del modelo es compatible con la base de datos con la que se pretende conectar.

Tal y como se explicó previamente, en Digi3D 2011 se puede indicar si un código está enlazado o no con una base de datos.

Si un código está enlazado con una base de datos, deberá de existir el esquema de dicha tabla en el propio Digi.tab.xml. Para ello, se ha añadido una nueva sección *DatabaseSchemas* en el Digi.tab.xml.

En esta sección se definen esquemas de tablas y se debe definir al menos todas las tablas referenciadas por los códigos, de modo que si un código está vinculado con la tabla Viales, deberá existir forzosamente la definición de dicha tabla en la sección Esquemas de la tabla de códigos. Digo al menos, porque es posible que otros programas o módulos de Digi3D como por ejemplo el exportador DigiNG.IO.Shp ó DigiNG.IO.Geomedia utilicen sus propias tablas que deberán estar descritas también en esta sección.

El proceso de comprobación de compatibilidad Digi.tab.xml <-> Modelo de datos consiste en varias fases:

Comprobación de existencia en la base de datos de todas las tablas indicadas en la tabla de códigos.

Comprobación de que se define el esquema de la tabla en la tabla de códigos.

Comprobación de que el esquema de la tabla tiene un campo de tipo clave primaria

Comprobación de que el nombre del campo clave primaria en el esquema y en la base de datos coinciden.

Para cada campo indicado en el esquema de base de datos en la tabla de códigos, comprobación de que su tipo coincide con el de la base de datos.

***Comprobación de existencia en la base de datos de todas las tablas indicadas en la tabla de códigos.***

Digi3D localiza los nombres de todas las tablas enlazadas por los códigos de la tabla de códigos. Una vez identificados los nombres de todas las tablas utilizadas, se consulta al modelo de datos el número de cada una de ellas. Si el modelo de datos no sabe el número de alguna de las tablas se genera un error que Digi3D le comunicará al usuario mediante los mecanismos habituales.

***Laboratorio 8 (Tabla enumerada en la tabla de códigos pero inexistente en la base de datos)***

En este laboratorio vamos a conectarnos con una base de datos utilizando una tabla Digi.tab.xml que tiene una tabla desconocida en la base de datos para comprobar como comunica el error Digi3D 2011.

La tabla de códigos Digi.tab.xml de este laboratorio tiene dos códigos “0” y “1” que enlazan con las tablas “UsosDeSuelo” y “Clientes”, tablas que no están definidas en la base de datos *CompatibilidadCATDBS*.

1. Hacemos doble clic en la tabla de códigos [ruta al laboratorio de base de datos]\Laboratorio 8\Laboratorio.tab.xml para que lo abra Internet Explorer o el bloc de notas y comprobamos que el código “0” está asociado con la tabla “UsosDeSuelo” y que el código “1” está asociado con la tabla “Clientes”.
2. Cerramos Internet Explorer o el bloc de notas.
3. Ejecutamos Digi3D, cargamos el archivo [ruta al laboratorio de base de datos]\Laboratorio 8\Modelo.bin, seleccionando como modelo de datos CATDBS y con la cadena de conexión Provider=SQLOLEDB.1;Integrated

- ```
Security=SSPI;Persist Security Info=False;Initial Catalog=CompatibilidadCATDBS;Data Source=.\\SQLEXPRES
```
4. Comprobamos que Digi3D carga correctamente el archivo de dibujo pero nos muestra una ventana indicando que se han localizado errores al comprobar la compatibilidad de la tabla de códigos con el modelo de datos.
 5. En la *Ventana de resultados* tenemos una entrada por cada error localizado y en la *Ventana de tareas* podemos ver las dos tareas que nos indican de los errores localizados.

Comprobación de que se define el esquema de la tabla en la tabla de códigos

La siguiente fase consiste en localizar en la tabla de códigos el esquema de todas las tablas utilizadas por los códigos que tengan enlace a base de datos.

Es obligatorio que existan en la sección de esquemas de la tabla de códigos los esquemas de todas las tablas utilizadas por los códigos de éstas, es decir, que si en la tabla de códigos se indica que el código CAA010 enlaza con la tabla AAA010, deberá de existir en la sección de esquemas de la tabla de códigos para dicha tabla.

Laboratorio 9 (Tabla sin esquema)

En este laboratorio vamos a conectarnos con la base de datos *CompatibilidadCATDBS* con una tabla de códigos con un único código “0” con enlace con la tabla “Viales”, tabla que existe en nuestra base de datos, pero cuyo esquema no está definido en la tabla de códigos.

1. Hacemos doble clic en la tabla de códigos [ruta al laboratorio de base de datos]\Laboratorio 9\Laboratorio.tab.xml para que lo abra Internet Explorer o el bloc de notas. Nos vamos al final del archivo y comprobamos que el código “0” está asociado con la tabla “Viales” pero no existe ninguna sección DatabaseSchemas, por lo tanto no existe la definición de esa tabla en la tabla de códigos.
2. Cerramos Internet Explorer o el bloc de notas.
3. Ejecutamos Digi3D, cargamos el archivo [ruta al laboratorio de base de datos]\Laboratorio 9\Modelo.bin, seleccionando como modelo de datos CATDBS y con la cadena de conexión Provider=SQLOLEDB.1;Integrated Security=SSPI;Persist Security Info=False;Initial Catalog=CompatibilidadCATDBS;Data Source=.\\SQLEXPRES
4. Comprobamos que Digi3D carga correctamente el archivo de dibujo pero nos muestra una ventana indicando que se han localizado errores al comprobar la compatibilidad de la tabla de códigos con el modelo de datos.
5. En la *Ventana de resultados* tenemos una entrada por cada error localizado y en la *Ventana de tareas* podemos ver las dos tareas que nos indican de los errores localizados.

Comprobación de que la tabla de que el esquema de la tabla tiene un campo de tipo clave primaria

El esquema de una tabla en la tabla de códigos consiste en un nodo *table* que contiene nodos de tipo *field*.

En el nodo *table* se define el nombre de la tabla y en los nodos *field* se definen los campos de la tabla que queremos mostrar al usuario.

El nodo *table* tiene un único atributo que se explica a continuación:

Nombre	Descripción
name	Sirve para indicar el nombre de la tabla cuyo esquema se está enumerando en el nodo.

Los nodos de tipo *field* tienen una serie de atributos que se enumeran a continuación:

Nombre	Descripción	Valores
name	Indica el nombre del campo. Debe coincidir con el nombre del campo en la tabla de la base de datos.	Cadena de caracteres
title	Indica el nombre que se mostrará al usuario. Es necesario porque es posible que el nombre del campo en la base de datos tenga un nombre incómodo para el usuario, como por ejemplo [PARAMETRO_1] en vez de “Nombre de la via”.	Cadena de caracteres
	Si se omite este atributo, Digi3D mostrará el nombre del campo.	
description	Texto a mostrar al usuario en la ventana de atributos de base	Texto

	de datos cuando esté rellenando el campo.									
primaryKey	Indica si el campo es el campo índice de la tabla.	Yes/No o 1/0								
type	Tipo de valor a mostrar en el campo. En función del tipo de valor Digi3D permitirá teclear letras o números.	<table border="1"> <thead> <tr> <th>Valor</th><th>Significado</th></tr> </thead> <tbody> <tr> <td>C</td><td>Cadena de caracteres</td></tr> <tr> <td>N</td><td>Número entero</td></tr> <tr> <td>F</td><td>Número real</td></tr> </tbody> </table>	Valor	Significado	C	Cadena de caracteres	N	Número entero	F	Número real
Valor	Significado									
C	Cadena de caracteres									
N	Número entero									
F	Número real									
length	Longitud del campo en la base de datos. Este atributo lo utilizan los exportadores que crean tablas nuevas como por ejemplo el exportador de Shapefiles.	Número								
decimalCount	Número de decimales en caso de que el tipo del campo sea <i>Número real</i> Este atributo lo utilizan los exportadores que crean tablas nuevas como por ejemplo el exportador de Shapefiles.	Número								
defaultValue	Valor que se asignará al campo por defecto. Si se selecciona un valor comodín, este no se calculará hasta el momento en el que se almacene la entidad. Los posibles valores comodín implementados son:	Se pueden poner constantes como "Revisión 2010" o valores comodines.								
		<table border="1"> <thead> <tr> <th>Valor</th><th>Descripción</th></tr> </thead> <tbody> <tr> <td>%UID%</td><td>Se almacenará en el campo un</td></tr> </tbody> </table>	Valor	Descripción	%UID%	Se almacenará en el campo un				
Valor	Descripción									
%UID%	Se almacenará en el campo un									

	número aleatorio único de 128 bits (GUID).
%ENTITY_CODE%	Se almacenará en el campo el código de la entidad.
%ENTITY_LENGTH_2D%	Se almacenará en el campo el perímetro bidimensional de la entidad.
%ENTITY_LENGTH_3D%	Se almacenará en el campo el perímetro tridimensional de la entidad.
%ENTITY_AREA%	Se almacenará en el campo el area de la entidad.
%ENTITY_FIRST_VERTEX_Z%	Se almacenará en el campo la coordenada Z del primer vértice de la entidad.
%ENTITY_FIRST_VERTEX_Z_INT%	Se almacenará en el campo la parte entera de la coordenada Z del primer vértice de la entidad.
%ENTITY_XMIN%	Se almacenará en el campo la coordenada X mínima de la entidad.
%ENTITY_YMIN%	Se almacenará en el campo la coordenada Y mínima de la entidad.
%ENTITY_ZMIN%	Se almacenará en el campo la coordenada Z mínima de la entidad.
%ENTITY_XMAX%	Se almacenará

		en el campo la coordenada X máxima de la entidad.
	%ENTITY_YMAX%	Se almacenará en el campo la coordenada Y máxima de la entidad.
	%ENTITY_ZMAX%	Se almacenará en el campo la coordenada Z máxima de la entidad.
	%ENTITY_WIDTH%	Se almacenará en el campo el ancho (X máxima – X mínima) de la entidad.
	%ENTITY_HEIGHT%	Se almacenará en el campo el alto (Y máxima – YX mínima) de la entidad.
	%ENTITY_LENGTH%	Se almacenará en el campo el largo (Z máxima – ZX mínima) de la entidad.
	%SUBSTITUTING%	Se almacenará la sustitución de tipo <i>sustituidores</i> almacenada a continuación de la palabra %SUSTITUTING%.
forzeDefaultValue	Indica si se utilizará siempre el valor por defecto independientemente del valor que hubiera tecleado el usuario en la ventana Campos de la base de datos.	Yes/No o 1/0

		<p>Este atributo tiene sentido para campos marcados invisibles para el operador, pero que hay que almacenar forzosamente en la base de datos.</p>
valueList	<p>Este campo es opcional, y consiste en una lista de valores separados por el separador que se utilizarán como listado de opciones para el usuario.</p>	<p>El formato es el siguiente:</p> <p>Valor Título Descripción</p> <p>Ej: 107 Iglesia Selecciona esta opción si el edificio es una iglesia 110 Mezquita Selecciona esta opción si el edificio es una mezquita</p> <p>Si en un campo se selecciona este valor, cuando el usuario haga clic en el campo, Digi3D mostrará un cuadro de lista con dos opciones: Iglesia y Mezquita. Si el usuario selecciona Iglesia, el programa mostrará la ayuda “Selecciona esta opción si el edificio es una iglesia” y si almacena la entidad, en la base de datos se almacenará el valor 107.</p>
restrictedToValueList	Indica si se va a permitir únicamente seleccionar valores de entre los enumerados en el campo <i>Lista de Valores</i> o si por el contrario se va a permitir que el usuario teclee un valor manualmente.	Yes/No o 1/0
readOnly	Indica si se le permite al usuario modificar el valor asignado (mediante el atributo defaultValue o mediante programación) a este campo.	Yes/No o 1/0
allowZeroLength	Indica si se va a	Yes/No o 1/0

	permitir que el usuario almacene una entidad sin haber rellenado el campo previamente.	
visibility	Indica si Digi3D mostrará este campo en la ventana Campos de la base de datos. Este campo cobra sentido si se indica un valor por defecto para el campo.	Yes/No o 1/0
bgColor	Color con el que Digi3D mostrará el campo en la ventana Campos de la base de datos. Este campo permite clasificar campos por su color de fondo para ayudar al usuario.	Color estándar web: #[rojo en hexadecimal][verde en hexadecimal][azul en hexadecimal] Ej: #0abf77

Todos los esquemas de tabla localizados en la tabla de códigos deben de tener un campo con el atributo *primaryKey* con valor verdadero, y el resto con valor falso.

Laboratorio 10 (Tabla sin clave primaria)

En este laboratorio vamos a conectarnos con la base de datos *CompatibilidadCATDBS* con una tabla de códigos con un único código “0” con enlace con la tabla “Viales”, cuyo esquema si está definido en la propia tabla de códigos pero no tiene ningún campo de tipo PrimaryKey para comprobar cómo comunica el error Digi3D.

1. Hacemos doble clic en la tabla de códigos Laboratorio.tab.xml para que lo abra Internet Explorer o el bloc de notas. Nos vamos al final del archivo y comprobamos que el código “0” está asociado con la tabla “Viales”, además ahora nuestra tabla de códigos dispone de una sección *databaseSchema* que define la tabla *Viales* con un único campo “A” que no está marcado como el campo de tipo clave primaria (porque no tiene un atributo *primaryKey*=“1”).

2. Cerramos Internet Explorer o el bloc de notas.
3. Ejecutamos Digi3D, cargamos el archivo [ruta al laboratorio de base de datos]\Laboratorio 10\Modelo.bin, seleccionando como modelo de datos CATDBS y con la cadena de conexión Provider=SQLOLEDB.1;Integrated Security=SSPI;Persist Security Info=False;Initial Catalog=CompatibilidadCATDBS;Data Source=.\\SQLEXPRES
4. Comprobamos que Digi3D carga correctamente el archivo de dibujo pero nos muestra una ventana indicando que se han localizado errores al comprobar la compatibilidad de la tabla de códigos con el modelo de datos.
5. En la *Ventana de resultados* tenemos una entrada por cada error localizado y en la *Ventana de tareas* podemos ver las dos tareas que nos indican de los errores localizados.

Comprobación de que el nombre del campo clave primaria en el esquema y en la base de datos coinciden

El nombre del campo marcado como *clave primaria* en la tabla de códigos debe coincidir con el nombre del campo *clave primaria* en la base de datos.

Laboratorio 11 (Distintas claves primarias)

En este laboratorio vamos a conectarnos con la base de datos *CompatibilidadCATDBS* con una tabla de códigos con un único código “0” con enlace con la tabla “Viales”, cuyo esquema si está definido en la propia tabla de códigos con un único campo “A” marcado como el campo *clave primaria* pero que no es el campo *clave primaria* de a base de datos, que como sabemos para la tabla *Viales* es *IDVial*.

1. Hacemos doble clic en la tabla de códigos Laboratorio.tab.xml para que lo abra Internet Explorer o el bloc de notas. Nos vamos al final del archivo y comprobamos que el código “0” está asociado con la tabla “Viales”. En la definición de la tabla Viales podemos comprobar que se define el campo “a” marcado con el atributo de *clave primaria*.
2. Cerramos Internet Explorer o el bloc de notas.
3. Ejecutamos Digi3D, cargamos el archivo [ruta al laboratorio de base de datos]\Laboratorio 11\Modelo.bin, seleccionando como modelo de datos CATDBS y con la cadena de conexión Provider=SQLOLEDB.1;Integrated Security=SSPI;Persist Security Info=False;Initial Catalog=CompatibilidadCATDBS;Data Source=.\\SQLEXPRES
4. Comprobamos que Digi3D carga correctamente el archivo de dibujo pero nos muestra una ventana indicando que se han localizado errores al comprobar la compatibilidad de la tabla de códigos con el modelo de datos.
5. En la *Ventana de resultados* tenemos una entrada por cada error localizado y en la *Ventana de tareas* podemos ver las dos tareas que nos indican de los errores localizados.

Para cada campo indicado en el esquema de base de datos en la tabla de códigos, comprobación de que su tipo coincide con el de la base de datos.

Una vez superadas todas las pruebas anteriores se puede afirmar que la tabla de códigos es compatible con el modelo de datos, pero Digi3D realiza una comprobación adicional que esta vez no va a generar errores, sino que generará advertencias, y consiste en analizar el tipo de dato (cadena de caracteres, número entero o número real) de todos los campos indicados en el esquema de la tabla de códigos con respecto a sus *alter egos* en la base de datos, de modo que si por ejemplo en la tabla de códigos se indica que el tipo del campo *Area* en una determinada tabla es *número entero* y en la base de datos es de tipo *número real*, se genere una advertencia indicando que Digi3D no le va a mostrar al operador un número real (con coma decimal...) sino que le va a mostrar un número entero, por lo tanto el operador nunca podrá teclear decimales.

Laboratorio 12 (Los tipos no coinciden)

En este laboratorio vamos a conectarnos con la base de datos *CompatibilidadCATDBS* con una tabla de códigos con un único código “0” con enlace con la tabla “Viales”, cuyo esquema está definido en la propia tabla de códigos con el campo de tipo *clave* primaria cuyo nombre coincide con el indicado en la base de datos, pero con el campo *Nombre* con tipo *número* entero en vez de tipo *cadena de caracteres*

1. Hacemos doble clic en la tabla de códigos Laboratorio.tab.xml para que lo abra Internet Explorer o el bloc de notas. Nos vamos al final del archivo y comprobamos que el código “0” está asociado con la tabla “Viales”. En la definición de la tabla Viales podemos comprobar que se define el campo “Nombre” marcado con tipo *número* entero en vez de tipo *cadena de caracteres*.
2. Cerramos Internet Explorer o el bloc de notas.
3. Ejecutamos Digi3D, cargamos el archivo [ruta al laboratorio de base de datos]\Laboratorio 12\Modelo.bin, seleccionando como modelo de datos CATDBS y con la cadena de conexión Provider=SQLOLEDB.1;Integrated Security=SSPI;Persist Security Info=False;Initial Catalog=CompatibilidadCATDBS;Data Source=.\SQLEXPRESS
4. Comprobamos que Digi3D carga correctamente el archivo de dibujo pero nos muestra una ventana indicando que se han localizado advertencias al comprobar la compatibilidad de la tabla de códigos con el modelo de datos.
5. En la *Ventana de resultados* tenemos una entrada por cada advertencia localizada y en la *Ventana de tareas* podemos ver las la tarea (en la sección advertencias) que nos indica de esta advertencia.

El campo clave primaria

Los modelos de datos *CATDBS* y *Geographics* tratan de una manera especial el campo de tipo *clave primaria* de las tablas cuando intentan almacenar un registro en una tabla.

Si cualquiera de estos dos modelos de datos comienza a llenar la información que se debe almacenar en el registro de base de datos que se está creando, cuando llega al campo de tipo *clave primaria* salta al siguiente campo, sin asignar en este campo ningún valor, por lo tanto el campo se intenta almacenar con valor *NULL*.

Y esto es así porque estos modelos de datos esperan que sea la propia base de datos la que genere el número de forma automática.

Si recordamos el esquema de la tabla *Viales*:

Column name	Data Type	Allow Nulls	Primary Key
IDVial	int	No	Si
Nombre	varchar(256)	No	No

El campo *IDVial* es el campo de tipo *clave primaria*. Este campo además no admite nulos, por lo tanto el servidor de bases de datos nos va a obligar a que introduzcamos aquí un número cada vez que almacenemos un registro.

Vamos a comprobarlo en acción:

Laboratorio 13 (*Los campos PrimaryKey deben ser auto-incrementales*)

En este laboratorio vamos a comprobar que a pesar de que se han superado todas las pruebas de compatibilidad Modelo de datos <-> Base de datos <-> Tabla de códigos, aún no tenemos correctamente configurada nuestra base de datos, y el servidor de bases de datos nos va a comunicar un error cada vez que intentemos almacenar algún registro en la base de datos.

La tabla de códigos de este laboratorio dispone de dos códigos: 0 y “Calle”. El código “Calle” enlaza con la tabla *Viales*. Seleccionaremos este código, rellenaremos el nombre del vial, almacenaremos una entidad. El modelo de datos *CATDBS* le informará al servidor de bases de datos que quiere añadir un registro nuevo en la tabla *Viales* y le indicará el valor que debe tener el campo *Nombre* de ese registro, pero no informará del valor que debe tener el campo *IDVial*, ya que este es el campo de tipo *clave primaria* de la tabla y sabemos que ni el modelo de datos *CATDBS* ni *Geographics* informan al servidor de bases de datos el valor que debe tener este campo. El servidor de bases de datos comunicará con un mensaje de error que no se ha enviado ningún valor para el campo *IDVial*. Comprobaremos cómo comunica Digi3D el error generado en el servidor. Solucionaremos el problema realizando un cambio en la base de datos, volveremos a probar a almacenar la entidad y por último comprobaremos que la información se ha almacenado correctamente en la base de datos.

El campo IDVial está marcado como campo de tipo *clave primaria*. Y eso quiere decir que el servidor de bases de datos nos va a garantizar que es imposible crear dos registros en la base de datos con el mismo valor en dicho campo. El valor del campo *IDVial* de todos los registros de la tabla será único siempre. Lo que no será único es el nombre del vial, podríamos tener varios registros (cada uno con su propio IDVial lógicamente) cuyo campo *Nombre* sea idéntico, por ejemplo *Gran Vía*.

Vamos a recordar lo que se decía unos párrafos antes: "Y esto es así porque estos modelos de datos esperan que sea la propia base de datos la que genere el número de forma automática."

Que un campo sea el campo clave primaria no quiere decir que el servidor lo vaya a llenar de forma automática. El servidor no tiene ni idea de qué valores almacenar en ese campo (quizás hay que almacenar potencias de dos, o números primos, o seguir una determinada secuencia...), por eso exigirá que introduzcamos un valor cada vez que almacenemos un registro nuevo.

Para solucionar el problema, le indicaremos al servidor de bases de datos que el campo IDVial es *auto-incremental* de manera que si no se especifica ningún valor para el campo IDVial, ahora el servidor sí que sabe qué información almacenar en él, y en vez de lanzar un error (en este caso a Digi3D 2011) optará por almacenar como valor el siguiente al valor máximo almacenado en la tabla. Todo esto de manera segura con múltiples usuarios.

1. Ejecutamos Digi3D, cargamos el archivo [ruta al laboratorio de base de datos]\Laboratorio 13\Modelo.bin, seleccionando como modelo de datos CATDBS y con la cadena de conexión Provider=SQLOLEDB.1;Integrated Security=SSPI;Persist Security Info=False;Initial Catalog=CompatibilidadCATDBS;Data Source=.\\SQLEXPRESS
2. Seleccionamos el código "Calle".
3. Comprobamos que Digi3D 2011 nos muestra en la ventana *Base de datos* una ventana para llenar el nombre de la calle. Si no tenemos activa la *Ventana Campos de la base de datos*, podemos activarla en el menú *Ver/Ventana Campos de la base de datos*.
4. En el campo *Nombre* en la *Ventana Campos de la base de datos*, tecleamos *Gran vía*.
5. Dibujamos una línea y la finalizamos.
6. Comprobamos que Digi3D 2011 nos muestra un cuadro de diálogo informándonos de la condición de error aparte de preguntarnos qué hacer. Por ahora no vamos a entrar en detalle en este cuadro de diálogo.
7. Despliega el botón titulado *Más información* y comprueba que en *Descripción del error* aparece el mensaje enviado por el servidor, y este es que no se puede insertar un valor NULL en el campo *IDVial* de la tabla *Viales*.
8. Deja este cuadro de diálogo así y ejecuta *Microsoft SQL Server Management Studio* y conéctate con la base de datos.
9. Localiza la base de datos CompatibilidadCATDBS y localiza la tabla *Viales*. Pulsa con el botón derecho en la tabla *Viales* y en el menú contextual selecciona *Design* para diseñar la tabla.

10. Hac clic en el campo IDVial y comprueba que en la ventana de propiedades *Column Properties*, en la sección *Table Designer* tienes un parámetro denominado *Identity Specification*. Despliégalos.
11. Cambia el valor de (*Is Identity*) de *No* a *Yes*. Verás que se rellenan de forma automática los campos *Identity Increment* e *Identity Seed*.
12. Cierra esta ventana. El programa Microsoft SQL Server Management Studio intentará almacenar los cambios, pero es posible que no pueda. Si la tabla ya tenía información almacenada, cambiar el diseño de ésta implica reescribirla y eso es algo que por defecto no se puede hacer ya que el programa viene configurado con la máxima seguridad posible y cambiar el diseño de una tabla se considera un cambio peligroso. Si al cerrar la ventana vemos que Microsoft SQL Server Management Studio muestra un mensaje de error indicando que no ha podido, solucionaremos el problema en el menú *Tools/Options*. En el cuadro de diálogo *Options* desplegamos *Designers* y en *Table and Database designers* en la parte de la derecha deselecicnamos *Prevent saving changes that require table re-creation*. Aceptamos y volvemos al paso 9.
13. Volvemos a Digi3D 2011 y pulsamos el botón titulado “Volver a intentarlo” en el cuadro de diálogo *El modelo de datos no ha conseguido almacenar la información en la base de datos* para indicarle a Digi3D que vuelva a intentarlo.
14. Comprobamos que ahora Digi3D 2011 no nos muestra ningún mensaje de error.
15. Ejecutamos la orden LISTA para listar la entidad que acabamos de almacenar y comprobamos que al comienzo del listado, Digi3D 2011 nos muestra los códigos de la entidad:
Código: Calle Tabla: 2 Id=xx
16. En Id tenemos el número de registro de nuestra entidad.
17. Vamos a comprobar en *Microsoft SQL Server Management Studio* la información que se ha almacenado en la base de datos.
18. Localiza la tabla Viales y pulsa con el botón derecho del ratón sobre el nombre de la tabla y en el menú contextual selecciona *Select top 1000 Rows*.
19. Comprobarás que se ha creado un *archivo de consulta* en la que se ha creado una consulta TSQL (Transact SQL). Además ésta consulta se ha ejecutado y podemos ver el resultado en la parte inferior de la ventana.
20. Comprueba que el nombre almacenado en el campo *Nombre* coincide con el nombre que almacenaste en el campo *Nombre* de la ventana Campos de la base de datos para el registro cuyo IDVial coincide con el campo Id mostrado en la orden Lista.
21. Vuelve a Digi3D 2011, dibuja otra línea, esta vez con otro nombre de calle y vuelve a *Microsoft SQL Server Management Studio*.
22. Comprobarás que *Microsoft SQL Server Management Studio* no está mostrando el último registro que acabas de crear con Digi. Esto es porque el resultado que se está mostrando es el resultado en el momento en que se ejecutó la consulta. Para volver a ejecutar todas las consultas del archivo de consultas activo, pulsa el botón *Execute* de la barra de herramientas *SQL Editor*.
23. Ahora sí que ves el último registro.
24. El botón *Execute* ejecuta las líneas seleccionadas del archivo de consultas activo. Si no has seleccionado ninguna línea, se ejecutan todas. Vamos a comprobarlo. Añade al final del archivo la siguiente línea:

```
SELECT Nombre FROM CompatibilidadCATDBS.dbo.Viales WHERE
IDVial=1
```

25. Ahora selecciona esta línea con el ratón o con las teclas de dirección manteniendo pulsada la tecla *Mayúsculas*.
26. Una vez seleccionada la línea, pulsa el botón *Execute*. Se acaba de ejecutar únicamente la línea que acababas de seleccionar.
27. Cierra Microsoft SQL Server Management Studio.
28. Cierra Digi3D 2011.

Configurando los esquemas en la tabla de códigos

Ya sabemos crear una base de datos compatible con el modelo de datos CATDBS de Digi3D 2011. A continuación vamos a ver cómo se comporta Digi3D 2011 en función de la información que almacenemos en el esquema de una tabla en la tabla de códigos.

Laboratorio 14 (Los campos desconocidos no se muestran)

Como sabemos, Digi3D 2011 muestra únicamente los campos enumerados en el esquema de la tabla en la tabla de códigos activa. Si no se enumera un campo, este no se visualizará en la ventana de atributos de base de datos, y por lo tanto cuando se intente almacenar un registro para esa tabla, Digi3D 2011 asumirá que tiene que asignar el valor NULL a dicho campo.

En este laboratorio vamos a añadir el campo [Fecha de finalización] a la tabla Viales admitiendo nulos para comprobar que como no se enumera ese campo en el esquema de la tabla de códigos y la base de datos admite valores nulos para ese campo, podemos almacenar entidades que enlazan con esa base de datos sin ningún problema.

1. Ejecutamos *Microsoft SQL Server Management Studio* y nos conectamos con nuestra base de datos CompatibilidadCATDBS.
2. Editamos el esquema de la tabla Viales y le añadimos un campo que denominaremos *Fecha de finalización* de tipo *date* y marcando el checkbox *Allow Nulls*. Comprobaremos que el IDE de *Microsoft SQL Server Management Studio* ha decorado el nombre del campo entre corchetes, ya que el nombre del campo tiene espacios.
3. Cerramos la ventana y aceptamos el cuadro de diálogo que nos pregunta si almacenar los cambios en la base de datos.
4. Localizamos el archivo [ruta al laboratorio de base de datos]\Laboratorio 14\Laboratorio.tab.xml y lo abrimos con Internet Explorer o con el bloc de notas, localizamos el nodo *databaseSchemas* y comprobamos que no está definido el campo *Fecha de finalización* en el esquema de la base de datos, por lo tanto Digi3D 2011 no lo

mostrará en la ventana Campos de la base de datos y asignará el valor NULL a ese campo cuando intente crear un registro nuevo en la tabla Viales.

5. Cerramos Internet Explorer o el bloc de notas.
6. Ejecutamos Digi3D 2011, cargamos el archivo [ruta al laboratorio de base de datos]\Laboratorio 14\Modelo.bin, seleccionando como modelo de datos CATDBS y con la cadena de conexión Provider=SQLOLEDB.1;Integrated Security=SSPI;Persist Security Info=False;Initial Catalog=CompatibilidadCATDBS;Data Source=.\\SQLEXPRES
7. Seleccionamos el código de Calle
8. Comprobamos que Digi3D 2011 únicamente muestra el campo *Nombre* en la ventana de propiedades de la ventana de bases de datos. Tecleamos el nombre de una calle.
9. Digitalizamos una línea y la finalizamos.
10. Comprobamos que Digi3D 2011 ha almacenado el registro en la base de datos : Ejecutamos *Microsoft SQL Server Management Studio*, nos conectamos con la base de datos seleccionamos la tabla Viales y con el botón derecho del ratón seleccionamos *Select Top 1000 Rows* para comprobar que se ha almacenado la información en la base de datos.

Laboratorio 15 (Valores constantes)

En este laboratorio vamos a hacer que Digi3D almacene un valor constante a un determinado campo de una tabla. Añadiremos a nuestra tabla Viales el campo *Contratista* y programaremos a Digi3D para que almacene el valor *Acme S.A.* en ese campo.

1. Ejecutamos *Microsoft SQL Server Management Studio* y nos conectamos con nuestra base de datos CompatibilidadCATDBS.
2. Editamos el esquema de la tabla Viales y le añadimos un campo que denominaremos *Contratista* de tipo *varchar(50)* y marcando el *checkbox Allow Nulls*.
3. Cerramos la ventana y aceptamos el cuadro de diálogo que nos pregunta si almacenar los cambios en la base de datos.
4. Localizamos el archivo [ruta al laboratorio de base de datos]\Laboratorio 15\Laboratorio.tab.xml y lo abrimos con Internet Explorer o con el bloc de notas, localizamos el nodo *databaseSchemas* y comprobamos que el esquema de la tabla *Viales* incluye un campo *Contratista* de tipo cadena de caracteres con el valor por defecto *Acme S.A.*
5. Cerramos Internet Explorer o el bloc de notas.
6. Ejecutamos Digi3D 2011, cargamos el archivo [ruta al laboratorio de base de datos]\Laboratorio 15\Modelo.bin, seleccionando como modelo de datos CATDBS y con la cadena de conexión Provider=SQLOLEDB.1;Integrated Security=SSPI;Persist Security Info=False;Initial Catalog=CompatibilidadCATDBS;Data Source=.\\SQLEXPRES

7. Seleccionamos el código de Calle.
8. Comprobamos que Digi3D 2011 nos muestra los campos *Nombre* y *Contratista*. Digi3D está mostrando el campo Contratista con el valor asignado por defecto: Acme S.A. Indicar en la tabla de códigos un valor por defecto para un campo en una tabla de la base de datos no implica que ese sea el valor se vaya a almacenar forzosamente en la base de datos. El usuario es libre de cambiar ese valor si quiere.
9. Cambiamos el valor de Acme S.A. a cualquier otro valor.
10. Registramos tres entidades.
11. Comprobamos (no voy a explicar más veces cómo hacerlo) que en la base de datos se han almacenado tres entradas, todas ellas con el nombre de contratista que hemos tecleado.
12. Seleccionamos como código activo el código 0, para que se vacíe la ventana de atributos.
13. Volvemos a seleccionar el código Calle y comprobamos que Digi3D 2011 está memorizando los últimos datos tecleados para la tabla Viales. Este es el comportamiento de Digi3D 2011. El programa memoriza por cada tabla los últimos valores que ha tecleado el usuario.
14. Para volver a dejar los valores por defecto, pulsamos el botón de deshacer de la barra de herramientas de la ventana de bases de datos (botón con una flecha a la izquierda).
15. Comprobamos que ahora el campo Nombre vuelve a estar vacío y el campo Contratista vuelve a estar con el valor por defecto Acme S.A.

Laboratorio 16 (Forzar valor por defecto)

En este laboratorio vamos a indicarle a Digi3D que fuerce a que se almacene el valor por defecto indicado en la tabla de códigos. Si se indica en algún campo en el esquema de la base de datos que se fuerce el valor por defecto, el campo aparecerá en modo sólo lectura, de modo que el usuario no podrá modificar el valor asignado en el campo.

1. Localizamos el archivo [ruta al laboratorio de base de datos]\Laboratorio 16\Laboratorio.tab.xml y lo abrimos con Internet Explorer o con el bloc de notas, localizamos el nodo *databaseSchemas* y comprobamos que el esquema de la tabla *Viales* incluye un campo *Contratista* de tipo cadena de caracteres con el valor por defecto Acme S.A. y con el atributo *forceDefaultValue* con valor “1”.
2. Cerramos Internet Explorer o el bloc de notas.
3. Ejecutamos Digi3D 2011, cargamos el archivo [ruta al laboratorio de base de datos]\Laboratorio 16\Modelo.bin, seleccionando como modelo de datos CATDBS y con la cadena de conexión Provider=SQLOLEDB.1;Integrated Security=SSPI;Persist Security Info=False;Initial Catalog=CompatibilidadCATDBS;Data Source=.\SQLEXPRES

4. Seleccionamos el código de Calle.
5. Comprobamos que Digi3D 2011 nos muestra los campos *Nombre* y *Contratista*. Digi3D está mostrando el campo Contratista con el valor asignado por defecto: Acme S.A. y además es de sólo lectura. No podemos modificar este valor.

Laboratorio 17 (Ocultando campos al usuario)

No tiene mucho sentido mostrarle al usuario un campo que va a tener siempre el mismo valor y que además es de solo lectura como son los campos de tipo *forceDefaultValue*. Los campos ocupan espacio por lo que mi recomendación, a menos que sea estrictamente necesario es ocultar lo máximo posible al usuario.

En este laboratorio vamos a ocultar el campo Contratista para que no ocupe espacio en la ventana Campos de la base de datos. Almacenaremos una entidad y comprobaremos que la información se ha almacenado correctamente en la base de datos.

1. Localizamos el archivo [ruta al laboratorio de base de datos]\Laboratorio 17\Laboratorio.tab.xml y lo abrimos con Internet Explorer o con el bloc de notas, localizamos el nodo *databaseSchemas* y comprobamos que el esquema de la tabla *Viales* incluye un campo *Contratista* de tipo cadena de caracteres con el valor por defecto Acme S.A. y con el atributo *forceDefaultValue* con valor “1” pero ahora marcado con el atributo *visibility* con valor “0”.
2. Cerramos Internet Explorer o el bloc de notas.
3. Ejecutamos Digi3D 2011, cargamos el archivo [ruta al laboratorio de base de datos]\Laboratorio 17\Modelo.bin, seleccionando como modelo de datos CATDBS y con la cadena de conexión Provider=SQLOLEDB.1;Integrated Security=SSPI;Persist Security Info=False;Initial Catalog=CompatibilidadCATDBS;Data Source=.\SQLEXPRESS
4. Seleccionamos el código de Calle.
5. Comprobamos que Digi3D 2011 nos muestra únicamente el campo *Nombre*. Ahora no nos muestra el campo *Contratista*.
6. Rellenamos el campo *Nombre* con algún valor y almacenamos una entidad.
7. Comprobamos con *Microsoft SQL Management Studio* que Digi3D 2011 ha almacenado correctamente el valor *Contratista* con el valor *Acme S.A.* en el registro recién añadido.

Laboratorio 18 (Propiedades dinámicas)

En este laboratorio vamos a almacenar propiedades dinámicas en ciertos campos en la base de datos. Para ello utilizaremos valores comodines. Añadiremos a nuestra tabla *Viales* campos para almacenar las coordenadas mínimas y máximas de las entidades almacenadas así como su

área y perímetro. Comprobaremos que Digi3D 2011 calcula estos valores de forma automática cuando almacena las entidades.

- Ejecutamos *Microsoft SQL Server Management Studio*, nos conectamos con nuestro servidor de base de datos y editamos el esquema de la tabla *Viales* para añadir los siguientes campos:

Nombre	Tipo	Admite nulos
Xmin	Real	Si
Ymin	Real	Si
Xmax	Real	Si
Ymax	Real	Si
Area	Real	Si
Perimetro	Real	Si

- Cerramos la ventana y aceptamos el cuadro de diálogo que nos solicita confirmación para modificar el esquema de la tabla.
- Localizamos el archivo [ruta al laboratorio de base de datos]\Laboratorio 18\Laboratorio.tab.xml y lo abrimos con Internet Explorer o con el bloc de notas, localizamos el nodo *databaseSchemas* y comprobamos que el esquema de la tabla *Viales* se han incluido estos campos y que tienen asignado un valor por defecto, cada uno de ellos con un valor comodín que Digi3D 2011 interpretará para transformarlo en dinámicamente. Todos estos campos tienen su atributo *visibility* a falso, por lo tanto el operador no los verá en la ventana de bases de datos.
- Cerramos Internet Explorer o el bloc de notas.
- Ejecutamos Digi3D 2011, cargamos el archivo [ruta al laboratorio de base de datos]\Laboratorio 18\Modelo.bin, seleccionando como modelo de datos CATDBS y con la cadena de conexión Provider=SQLOLEDB.1;Integrated Security=SSPI;Persist Security Info=False;Initial Catalog=CompatibilidadCATDBS;Data Source=.\SQLEXPRES
- Seleccionamos el código de Calle.
- Comprobamos que Digi3D 2011 nos muestra únicamente el campo *Nombre*.
- Rellenamos el campo *Nombre* con algún valor y almacenamos una entidad.
- Comprobamos con *Microsoft SQL Management Studio* que Digi3D 2011 ha almacenado correctamente valores en estos campos.

Laboratorio 19 (Propiedades dinámicas: Sustituidores)

En este laboratorio vamos a utilizar una propiedad dinámica especial, la que pasa la cadena que escribamos por el proceso de *Sustituidores*.

Si indicamos que el valor por defecto para una entidad es: %SUBSTITUTING%, Digi3D 2011 transformará mediante el proceso de sustituidores la frase que se encuentre a continuación de la palabra %SUSTITUTING%. **Debe de haber un espacio** entre la palabra %SUBSTITUTING% y la frase a transformar mediante sustituidores.

- Ejecutamos *Microsoft SQL Server Management Studio*, nos conectamos con nuestro servidor de base de datos y editamos el esquema de la table *Viales* para añadir el siguiente campo:

Nombre	Tipo	Admite nulos
Seguridad	varchar(256)	Si

- Cerramos la ventana y aceptamos el cuadro de diálogo que nos solicita confirmación para modificar el esquema de la tabla.
- Localizamos el archivo [ruta al laboratorio de base de datos]\Laboratorio 19\Laboratorio.tab.xml y lo abrimos con Internet Explorer o con el bloc de notas, localizamos el nodo *databaseSchemas* y comprobamos que el esquema de la tabla *Viales* se ha incluido el siguiente campo

Nombre	Valor por defecto
Seguridad	%SUBSTITUTING% Almacenado por el usuario \$(UserName) desde el ordenador \$(ComputerName) a las \$(Time) del \$(Date)

Este campo tiene su atributo *visibility* a falso, por lo tanto el operador no lo verá en la ventana de bases de datos.

- Cerramos Internet Explorer o el bloc de notas.
- Ejecutamos Digi3D 2011, cargamos el archivo [ruta al laboratorio de base de datos]\Laboratorio 19\Modelo.bin, seleccionando como modelo de datos CATDBS y con la cadena de conexión Provider=SQLOLEDB.1;Integrated Security=SSPI;Persist Security Info=False;Initial Catalog=CompatibilidadCATDBS;Data Source=.\SQLEXPRES
- Seleccionamos el código de Calle.
- Comprobamos que Digi3D 2011 nos muestra únicamente el campo *Nombre*.
- Rellenamos el campo *Nombre* con algún valor y almacenamos una entidad.
- Comprobamos con *Microsoft SQL Management Studio* que Digi3D 2011 ha almacenado correctamente el valor en este campo. En mi caso puedo leer:

“Almacenado por el usuario Jose Angel desde el ordenador JOSEANGEL-VAIO a las 11:18:31 del 01/04/11”

Laboratorio 20 (Extendiendo el sistema de propiedades dinámicas de Digi3D 2011)

Para entender este laboratorio, es necesario que tengas conocimientos de programación.

Es posible que queramos almacenar en un campo una propiedad dinámica que no esté en la lista de propiedades dinámicas implementadas por Digi3D 2011.

Supongamos que nos interesa almacenar en un campo de la base de datos el número de vértices que tiene la entidad con coordenada X menor que un determinado valor.

Si buscamos en el listado de propiedades dinámicas que implementa Digi3D 2011 (%UID%, %ENTITY_CODE%, %ENTITY_LENGTH_2D%...) no vamos a encontrar ninguna propiedad que realice esa operación.

Como Digi3D 2011 es programable podemos programar nuestra propia propiedad dinámica.

El objetivo de este laboratorio es el de crear nuestra propia propiedad dinámica.

Crearemos una propiedad dinámica denominada %NUMERO_VERTICES_CON_X_MENOR_QUE% que exigirá que le pasemos un parámetro que será el valor que utilizaremos para contabilizar el número de vértices con coordenada X menor.

Así pues, si en el campo *defaultValue* de un determinado campo tecleamos %NUMERO_VERTICES_CON_X_MENOR_QUE% 200, nuestra propiedad dinámica almacenará en ese registro el número de vértices que tiene la entidad que se está almacenando con coordenada X menor que 200.

Como nuestra propiedad dinámica es parametrizable, la podemos utilizar en otro campo con otro valor como por ejemplo 100 en vez de 200.

Cuando ejecutamos Digi3D 2011, este comienza a cargar todas las extensiones registradas en el sistema. Se cargan las extensiones encargadas de cargar archivos raster (tiff, jpeg, nitf, ...), las extensiones encargadas de importar/exportar archivos de dibujo (.bin, .dwg, ...) y varios tipos de extensiones, entre las que están las extensiones de **propiedades dinámicas**.

Digi3D 2011 obtiene el listado de ensamblados que implementan propiedades dinámicas en la entrada del registro:

S.O.	Digi3	Clave del registro
de	D de	
32 0	32 o	
64	64	
bits	bits	
32	32	HKEY_LOCAL_MACHINE\Software\Digi21\DigI3D2011\DigING\Extensiones\DY

DynamicPropertyProviders			
64	32	HKEY_LOCAL_MACHINE\Software\Wow64Node\DigI21\DigI3D2011\DigING\Extensiones\DynamicPropertyProviders	
64	64	HKEY_LOCAL_MACHINE\Software\DigI21\DigI3D2011\DigING\Extensiones\DynamicPropertyProviders	

Al ejecutar Digi3D 2011, cuando llega el momento de cargar las extensiones de propiedades dinámicas, enumera las entradas de esa entrada del registro e intenta cargar todos los ensamblados enumerados. Los ensamblados que se carguen sin problemas, formarán parte del sistema de propiedades dinámicas de la aplicación.

Para cuando terminemos este laboratorio habremos añadido una entrada en el registro (para informarle a Digi3D 2011 el nombre del ensamblado que vamos a crear) y habremos hecho que Digi3D 2011 implemente la propiedad dinámica %NUMERO_VERTICES_CON_X_MENOR_QUE%.

1. Ejecuta Visual C# 2010 Express
2. Selecciona la opción del menú Archivo/Nuevo proyecto o pulsa Ctrl+Mays+N
3. En la parte central del cuadro de diálogo Nuevo proyecto selecciona Librería de clases y en Nombre, teclea *ProveedorPropiedadesDinamicas*.
4. Acepta el cuadro de diálogo.
5. Comprobarás que Visual C# 2010 Express ha creado una solución con el nombre *ProveedorPropiedadesDinamicas*. Esta solución tiene un único proyecto con el mismo nombre. Y este proyecto tiene un único archivo Class1.cs
6. Cambia de nombre el archivo Class1.cs en la propia ventana del Explorador de soluciones haciendo clic con el botón derecho del ratón en el archivo Class1.cs, seleccionando la opción Cambiar nombre del menú contextual y tecleando *ContabilizadorDeVerticesConXMenorQueValor.cs*.

Visual C# Express 2010 te preguntará si también quieres cambiar el nombre de la clase que había implementada dentro de ese archivo. Dile que sí y Visual C# 2010 cambiará el nombre de la clase automáticamente.

7. Ahora vamos a añadir como referencia los ensamblados Digi21.DigING y Digi21.DigING.Plugin. Pulsa el botón derecho del ratón en References de nuestro proyecto en la ventana Explorador de soluciones y selecciona la opción Agregar referencia.... en el menú contextual o selecciona la opción del menú Proyecto/Agregar referencia...
8. En el cuadro de diálogo *Agregar referencia*, selecciona la pestaña .NET localiza estos dos ensamblados. Selecciónalos y acepta el cuadro de diálogo.
9. Comprueba en la rama *References* del proyecto en la ventana *Solution Explorer* que ahora nuestro proyecto incluye Digi21.DigING y Digi21.DigING.Plugin entre su lista de referencias.

10. Ahora vamos a indicar que nuestra clase representa la propiedad dinámica %NUMERO_VERTICES_CON_X_MENOR_QUE%. Para ello tenemos que decorar la clase con el atributo *DynamicProperty* de la siguiente manera:

```
[DynamicProperty("NUMERO_VERTICES_CON_X_MENOR_QUE")]
public class ContabilizadorDeVerticesConXMenorQueValor
{
}
```

11. Comprobarás que Visual C# 2010 Express dibuja una línea roja debajo de la palabra *DynamicProperty* ya que no hemos escrito el nombre completo (el atributo *DynamicProperty* está ubicado en el espacio de nombres Digi21.Databases).
12. Pulsa con el botón derecho del ratón en *DynamicProperty* para ver la magia de Visual C# 2010 Express: Aparecerá un menú contextual, selecciona Resolver/using Digi21.Databases y comprobarás que Visual C# 2010 Express ha añadido la cláusula using Digi21.Databases al comienzo del programa.
13. Como vamos a utilizar objetos definidos en los espacios de nombres Digi21.DigiNG.Entities y Digi21.Math, añade al principio del archivo las siguientes líneas:

```
using Digi21.DigiNG.Entities;
using Digi21.Math;
```

14. Ya hemos decorado la clase con la propiedad *DynamicProperty* (que le indica a Digi3D 2011 que esta clase implementa una propiedad dinámica, y además el nombre de la propiedad dinámica que implementa). A continuación tenemos que hacer que la clase implemente el interfaz *IDynamicProperty*, que define el método *Transform* que será el que llame Digi3D 2011 cuando sea necesario.
15. Indica que la clase implementa el interfaz *IDynamicProperty* de la siguiente manera:

```
[DynamicProperty("NUMERO_VERTICES_CON_X_MENOR_QUE")]
public class ContabilizadorDeVerticesConXMenorQueValor : IDynamicProperty
{
}
```

16. Pulsa con el botón derecho del ratón otra vez en *IDynamicProperty* y en el menú contextual selecciona Implementar interfaz para que Visual C# 2010 Express añada por nosotros una implementación para el método *Transform*.
17. Este método recibe dos parámetros: El parámetro que acompaña en la tabla de códigos a la propiedad dinámica (el valor 200 en el caso de nuestro ejemplo) y la entidad que se está almacenando.
- Nuestra clase debería de tener el siguiente aspecto:

```
[DynamicProperty("NUMERO_VERTICES_CON_X_MENOR_QUE")]
public class ContabilizadorDeVerticesConXMenorQueValor : IDynamicProperty
{
    public object Transform(string parameter, Entity entity)
    {
        throw new NotImplementedException();
    }
}
```

```

        }
    }

```

18. Ahora vamos a implementar nuestra transformación. Lo primero que haremos será comprobar si el parámetro tiene asignado algún valor. Recuerda que nuestra transformación requiere un parámetro, el que indica la coordenada X por debajo de la cual vamos a contabilizar vértices.

Si en el atributo *defaultValue* en el esquema de la tabla para un campo Digi3D 2011 se encuentra %NUMERO_VERTICES_CON_X_MENOR_QUE% 200, el valor que tendrá la variable *parameter* será “200”. Pero ¿Qué pasa si la persona que configura la tabla de códigos no pasa un parámetro a la propiedad dinámica? Tenemos que comunicar el error.

Elimina la línea `throw new NotImplementedException();` y añade lo siguiente:

```

    if( parameter == "" )
        throw new ArgumentNullException("Falta el parámetro de La
propiedad dinámica %NUMERO_VERTICES_CON_X_MENOR_QUE%");
```

19. De esta manera estaremos comunicando a Digi3D que se ha producido un error porque falta el parámetro.
20. A continuación vamos a intentar transformar el parámetro en un número de doble precisión. Esto también puede fallar: Imagínate que en el atributo *defaultValue* en la tabla nos encontramos con : %NUMERO_VERTICES_CON_X_MENOR_QUE% HOLA. Si intentamos transformar la palabra HOLA en un número de doble precisión tendremos problemas.
- El tipo *double* dispone de un método *TryParse* que devolverá falso si no se ha podido hacer la transformación.

```

double coordenadaBuscada;
if( !double.TryParse(parameter, out coordenadaBuscada ))
    throw new ArgumentException(string.Format("No se ha podido
transformar el valor {0} en un número de doble precisión", parameter));
```

21. Ya tenemos en la variable *coordenadaBuscada* el valor de la coordenada X pasada por parámetro. A continuación vamos a contabilizar los vértices.
- Vamos a implementar la cuenta únicamente para Puntos, Textos y Líneas.

Las entidades de tipo punto se implementan en Digi3D 2011 mediante la clase *Digi21.DigiNG.Entities.Point*.

Las entidades de tipo texto se implementan mediante la clase *Digi21.DigiNG.Entities.Text*.

Las entidades de tipo línea se implementan mediante la clase *Digi21.DigiNG.Entities.Line*.

Tanto las clases Point, como Text y Line heredan de Entity, lo que implica que todos los Point son también Entity, y todos los Text, y todos los Line, pero no al revés.

Si queremos saber si un Entity es un texto, podemos utilizar la instrucción de C# `is` de modo que si queremos saber si el objeto entity es de tipo Point lo haremos de la siguiente manera:

```
if (entity is Line) {}
```

Y si sabemos a ciencia cierta que un objeto de tipo Entity es de tipo Line, podemos convertirlo a Line con la instrucción de C# `as`.

Si te fijas, el parámetro `entity` de nuestro método Transform es de tipo Entity. Eso es así porque Digi3D 2011 llamará a nuestro método cada vez que se encuentre con la palabra %NUMERO_VERTICES_CON_X_MENOR_QUE% independientemente de si la entidad que está almacenando el usuario es una línea, un punto o un texto. Es nuestra responsabilidad descubrir si la entidad que nos está llegando es una línea, un punto o un texto.

Los objetos Point y Text disponen de una propiedad `Coordinate` que devuelve la coordenada de la entidad.

Los objetos de tipo Line disponen de una propiedad `Coordinates` que devuelve una lista de coordenadas.

Vamos a programar primero los casos de Punto y Texto que son los más sencillos:

```
if (entity is Point)
{
    Point punto = entity as Point;
    if (punto.Coordinate.X < coordenadaBuscada)
        return 1;
    return 0;
}

if (entity is Text)
{
    Text texto = entity as Text;

    if (texto.Coordinate.X < coordenadaBuscada)
        return 1;
}
```

Ahora continuamos con el caso de las líneas, y lo vamos a programar tres veces: el primero con programación imperativa (programación clásica), el segundo con una consulta *Linq to objects* y por último, con un *método de extensión* y una *expresión Lambda*. Si no entiendes las dos últimas deberías estudiar C# 4.0.

Primera implementación:

```
if (entity is Line)
{
    Line linea = entity as Line;
```

```

        int contador = 0;
        foreach(Point3D vértice in línea.Coordinates)
            if( vértice.X < coordenadaBuscada )
                ++contador;

        return contador;
    }
}

```

Segunda implementación, mediante una consulta Linq to Objects (que personalmente me gusta más)

```

if (entity is Line)
{
    Line línea = entity as Line;

    var vérticesConXMenorQue = from vértice in línea.Coordinates
                                where vértice.X < coordenadaBuscada
                                select vértice;
    return vérticesConXMenorQue.Count();
}

```

... y por último mi implementación favorita. Ésta es la versión más elegante, utilizando el método de extensión Count y una expresión lambda:

```

if (entity is Line)
{
    Line línea = entity as Line;

    return línea.Coordinates.Count(vértice => vértice.X <
coordenadaBuscada);
}

```

22. En caso de que la entidad no sea ni línea, ni punto ni texto (que es posible, porque Digi3D 2011 implementa más tipos) vamos a devolver un 0, de modo que nuestro programa completo debería tener el siguiente aspecto:

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Digi21.Databases;
using Digi21.Databases;
using Digi21.DigiNG.Entities;
using Digi21.Math;

namespace ProveedorPropiedadesDinamicas
{
    [DynamicProperty("NUMERO_VERTICES_CON_X_MENOR_QUE")]
    public class ContabilizadorDeVerticesConXMenorQueValor :
    IDynamicProperty
    {
        public object Transform(string parameter, Entity entity)
        {
            if (entity is Line)
            {
                Line línea = entity as Line;
                return línea.Coordinates.Count(vértice => vértice.X <
coordenadaBuscada);
            }
            else
                return 0;
        }
    }
}

```

```

    {
        // Si no se han pasado parámetros, lanzamos una excepción
        indicando que este transformador de propiedades dinámicas requiere un
        parámetro
        if( parameter == "" )
            throw new ArgumentNullException("Falta el parámetro de La
propiedad dinámica %NUMERO_VERTICES_CON_X_MENOR_QUE%");

        // Intentamos transformar el parámetro en un número doble. Si
        se produce un error, lo comunicamos al usuario.
        double coordenadaBuscada;
        if( !double.TryParse(parameter, out coordenadaBuscada ))
            throw new ArgumentException(string.Format("No se ha podido
transformar el valor {0} en un número de doble precisión", parameter));

        if (entity is Point)
        {
            Point punto = entity as Point;
            if (punto.Coordinate.X < coordenadaBuscada)
                return 1;
            return 0;
        }

        if (entity is Text)
        {
            Text texto = entity as Text;

            if (texto.Coordinate.X < coordenadaBuscada)
                return 1;
        }

        if (entity is Line)
        {
            Line linea = entity as Line;

            return linea.Coordinates.Count(vértice => vértice.X <
coordenadaBuscada);
        }

        return 0;
    }
}
}

```

23. Compila el programa Ctrl+Mays+B y corrige los errores que puedas haber cometido.
24. Tan solo nos queda registrar nuestra extensión para que Digi3D la cargue al iniciar.
Ejecuta el programa REGEDIT y localiza la entrada indicada en la primera tabla de este ejercicio en función del sistema operativo y la versión de Digi3D 2011 que tengas instalada (32 o 64 bits). Si no existe la carpeta DynamicPropertyProviders, créala, y añade una entrada de tipo DWORD cuyo nombre sea el nombre del ensamblado ProveedorPropiedadesDinamicas y cuyo valor sea 1. Si ponemos aquí un 0 Digi3D 2011 no cargará esa extensión.
25. Ejecuta Digi3D 2011. Si no aparece ningún mensaje de error, es que todo ha ido bien.
26. Ejecuta Microsoft SQL Server Management Studio, conéctate con el servidor de bases de datos y añade a la tabla Viales de la base de datos CompatibilidadCATDBS el campo:

Nombre	Tipo	Admite nulos
VerticesEspeciales	int	Si

27. Localiza con el explorador de archivos el archivo [ruta al laboratorio de base de datos]\Laboratorio 20\Laboratorio.tab.xml y comprueba que la tabla de códigos del laboratorio incluye este campo en el esquema de la tabla Viales y que además en el atributo *defaultValue* utiliza nuestra propiedad dinámica.
28. Abre con Digi3D 2011 el archivo [ruta al laboratorio de base de datos]\Laboratorio 20\Modelo.bin, selecciona el código *Calle* y digitaliza entidades con vértices a la izquierda y a la derecha de la coordenada 200,200. Comprueba en la tabla Viales con Microsoft SQL Server Management Studio el resultado.
29. Para finalizar si quieras, cierra Digi3D 2011, edita el archivo de tablas y provoca un error, como quitar el parámetro a la propiedad dinámica o poniendo como parámetro una cadena de caracteres, vuelve a cargar Digi3D 2011 y dibuja una entidad con el código *Calle* para ver cómo muestra Digi3D 2011 el mensaje de error.

Laboratorio 21 (Cuadros de opciones)

A continuación vamos a crear una nueva tabla *Instalaciones Deportivas*, en la que almacenaremos información de instalaciones deportivas. Nos interesa el nombre de la instalación deportiva, el área (que ordenaremos a Digi3D 2011 calcule de forma automática) y por último un valor numérico que indique el tipo de actividad deportiva que se realiza en cada entidad digitalizada.

Vamos a suponer que este último campo: *Tipo de actividad*, es de tipo numérico, y únicamente puede tener valores de entre los enumerados en la siguiente tabla:

Valor a almacenar	Significado
24537	Fútbol
642	Baloncesto
5	Voleibol
2247	Vóley playa
9087765	Tenis

Como estos valores son difíciles de memorizar, vamos a ayudar al usuario creando una lista de posibles valores gracias al atributo opcional *valueList* que pueden tener los campos de una tabla en el esquema en la tabla de códigos.

El formato del campo *valueList* es el siguiente: (valor|título|descripción)+ es decir: triplas de valores en el que el primer valor es el valor a almacenar en la tabla, el segundo es el título a mostrar al usuario y el tercero es una pequeña ayuda para que el usuario sepa el significado de

la opción (ya que las posibles opciones no tienen por qué ser tan triviales como las de este ejemplo). El separador de palabras es el carácter de barra vertical (|).

Para nuestro listado de opciones, el valor que debería tener el atributo *valueList* es el siguiente:

“24537|Fútbol|Selecciona esta opción si el recinto es un campo de futbol.|642|Baloncesto|Selecciona esta opción si el recinto es una cancha de baloncesto.|5|Voleibol|Selecciona esta opción si el recinto es un campo de voleibol con césped.|2247|Vóley playa|Selecciona esta opción si el recinto es un campo de voleibol en la playa.|9087765|Tenis|Selecciona esta opción si el recinto es una pista de tenis.”

En este laboratorio vamos a utilizar el programa Digi.Tab para crear la tabla de códigos de forma automática.

1. Ejecutamos *Microsoft SQL Server Management Studio*, nos conectamos con nuestro servidor de bases de datos y añadimos la siguiente tabla a la base de datos *ConectividadCATDBS*:

Nombre	Tipo	Admite nulos	Identidad	Primary Key
IDInstalación	int	no	Si, auto-incremental de 1 en 1	Si
Nombre	Varchar(64)	no	n/a	No
Área	real	no	n/a	No
Tipo de actividad	int	no	n/a	No

2. Cerramos el editor de esquemas y asignamos el nombre *Instalaciones Deportivas* a la tabla que acabamos de crear.
3. Añadimos a la tabla *CATDBS* (botón derecho sobre la tabla y seleccionando *Edit Top 200 Rows*) un registro con ID=4 y Tabla=Instalaciones Deportivas.
4. Localizamos el archivo [ruta al laboratorio de base de datos]\Laboratorio 20\Laboratorio.tab.xml y lo eliminamos si existe.
5. Ejecutamos el programa externo DigiTab.exe
6. Activamos la pestaña *Base de datos*. Comprobarás que está vacía porque la tabla de códigos que crea por defecto el programa DigiTab no tiene ninguna tabla.
7. Seleccionamos la opción del menú *Herramientas/Base de datos/Importar esquema de una base de datos en formato CATDBS*.
8. Aparecerá el cuadro de diálogo *Data Link Properties*. En *Provider* selecciona *Microsoft OLE DB Provider for SQL Server* y pulsamos *Next*.

9. En la pestaña *Connection* en el campo *Select or enter a Server Name*, tecleamos `.SQLEXPRESS` o desplegamos el cuadro de opciones (si desplegamos el campo de opciones posiblemente tengamos que esperar mucho tiempo, ya que en ese momento se inicia el proceso de localizar instancias de Microsoft SQL Server por toda la red) así que si sabemos el nombre de la instancia a la que queremos conectarnos, en este caso `.SQLEXPRESS`, pues será mejor escribirla que esperar a que el cuadro de diálogo localice todas las instancias visibles de nuestra red.
10. En el campo *Enter Information to log on to the server* seleccionamos *Use Windows NT Integrated security* que es el modelo más seguro de seguridad y que es además el que se configura por defecto cuando instalamos Microsoft SQL Server Express.
11. En el campo *Select the database on the server*, desplegamos el cuadro de opciones y seleccionamos la base de datos *CompatibilidadCATDBS*.
12. Pulsamos el botón *Test connection* para comprobar que la conexión se puede establecer correctamente y aceptamos el cuadro de mensajes que nos indica que se ha podido establecer la conexión sin problemas.
13. Por último aceptamos el cuadro de diálogo *Data Link Properties*.
14. Comprobarás que el programa DigiTab comienza a analizar la base de datos. El proceso consiste básicamente en comprobar si la base de datos dispone de una tabla CATDBS y en caso afirmativo, intenta localizar todas las tablas enumeradas en la tabla CATDBS.

Si has seguido uno a uno los laboratorios, tu tabla CATDBS debe tener el siguiente aspecto:

ID	Tabla
1	DBDIGI
2	Viales
3	Propietarios
4	Instalaciones Deportivas

Pero en nuestra base de datos, no existe la tabla *Propietarios*, por lo tanto el programa DigiTab nos va a mostrar un cuadro de diálogo informándonos que no ha conseguido localizar la tabla. Pulsamos el botón titulado *Saltar a la siguiente tabla*.

15. Comprobamos que ahora aparecen las dos tablas *Viales* e *Instalaciones Deportivas* en la pestaña *Base de datos*.
16. Seleccionamos el campo *Area* y en su propiedad *Valor por defecto* desplegamos la lista de posibles opciones y seleccionamos `%ENTITY_AREA%`.
Por último cambiamos el valor *Visible* de *Si* a *No*.
17. Selecciona el campo *Tipo de Actividad* y en el campo *Lista de valores* copiamos el siguiente texto: `24537|Fútbol|Selecciona esta opción si el recinto es un campo de futbol.|642|Baloncesto|Selecciona esta opción si el recinto es una cancha de`

baloncesto.|5|Voleybol|Selecciona esta opción si el recinto es un campo de voleybol con césped.|2247|Vóley playa|Selecciona esta opción si el recinto es un campo de volebol en la playa.|9087765|Tenis|Selecciona esta opción si el recinto es una pista de tenis.

18. Cambiamos el valor de la propiedad *Restringir Valores* de *No* a *Si*. De esta manera no vamos a permitir que el usuario teclee el valor que quiera y le vamos a obligar a seleccionar un valor de la lista.
19. Pulsamos el botón *Aplicar* para guardar los cambios realizados a la tabla *Tipo de Actividad*.
20. Seleccionamos la pestaña *Códigos*.
21. Pulsamos el botón *Nuevo* para añadir un código nuevo. Nombra el código “0” y aceptamos el cuadro de diálogo *Nuevo Código*.
22. Pulsamos el botón *Aplicar* para guardar los cambios al código “0”.
23. Pulsamos el botón *Nuevo* para crear el código “1” y aceptamos el cuadro de diálogo *Nuevo código*.
24. En la ventana de propiedades del código, ponemos en la propiedad *Descripción* el valor *Instalaciones deportivas*.
25. En la sección *Base de datos*, desplegamos la lista de la propiedad *Tabla* y selecciona la tabla *Instalaciones Deportivas*.
26. Pulsamos el botón *Aplicar* para guardar los cambios en el código “1”.
27. Pulsamos el botón *Aceptar*. Como no hemos asignado ningún nombre a nuestra tabla de códigos, el programa DigiTab nos pregunta qué queremos hacer con la tabla que acabamos de crear. Pulsamos el botón titulado *Guardar* y en el cuadro de diálogo *Guardar Como* guardamos el archivo en la ruta [ruta al laboratorio de base de datos]\Laboratorio 20 con nombre Laboratorio.tab.xml .
28. Ejecutamos Digi3D 2011, cargamos el archivo [ruta al laboratorio de base de datos]\Laboratorio 21\Modelo.bin, seleccionando como modelo de datos CATDBS y con la cadena de conexión Provider=SQLOLEDB.1;Integrated Security=SSPI;Persist Security Info=False;Initial Catalog=CompatibilidadCATDBS;Data Source=.\SQLEXPRES
29. Seleccionamos el código de Instalaciones Deportivas.
30. Comprobamos que Digi3D 2011 nos muestra únicamente el campo *Nombre*.
31. Ejecutamos Digi3D 2011, cargamos el archivo [ruta al laboratorio de base de datos]\Laboratorio 21\Modelo.bin, seleccionando como modelo de datos CATDBS y con la cadena de conexión Provider=SQLOLEDB.1;Integrated Security=SSPI;Persist Security Info=False;Initial Catalog=CompatibilidadCATDBS;Data Source=.\SQLEXPRES
32. Activamos la ventana de *Ayuda dinámica* en el menú *Ayuda/Ayuda dinámica*.
33. Seleccionamos el código de Instalaciones deportivas.

34. Tecleamos un nombre en el campo nombre .
35. Seleccionamos el campo *Tipo de actividad* y comprobamos que Digi3D 2011 muestra en la ventana de ayuda dinámica una tabla con los distintos valores que se pueden asignar para el campo *Tipo de actividad*.
36. Desplegamos la lista y seleccionamos Vóley Playa.
37. Digitalizamos una línea.
38. Comprobamos con *Microsoft SQL Server Management Studio* que la se ha almacenado el valor correcto (2247) en el campo *Tipo de actividad* del último registro.

Laboratorio 22 (Cuadros de opciones mediante SQL)

En este laboratorio vamos a simular el comportamiento del laboratorio anterior pero haciéndolo más dinámico.

Vamos a ponernos en la piel de la persona que modela la base de datos. Imaginémonos que nos damos cuenta de que falta un deporte practicado en toda la geografía española: *La Petanca*.

Tenemos que comunicar a todos los contratistas que a partir de ahora se pueden identificar campos de Petanca, y que el valor para este campo es el “1”.

Lo normal es que sea la propia base de datos quien almacene en alguna tabla los posibles valores que puede tener un determinado campo de una tabla de la base de datos.

Podríamos crear en la base de datos una tabla *Tipo de instalaciones deportivas* ésta podría tener tres campos y los siguientes valores:

Valor	Título	Descripción
24537	Fútbol	Campo de fútbol
642	Baloncesto	Cancha de baloncesto
5	Voleibol	Campo de voleibol
2247	Vóley playa	Zona en la playa donde se practica el Vóley Playa
9087765	Tenis	Cancha de tenis
1	Petanca	Lugar donde los jubilados juegan a la petanca

Pero en esta tabla únicamente podríamos almacenar información del campo *Tipo de actividad* de la tabla *Instalaciones deportivas*, de modo que si tenemos otro campo en otra tabla con posibles valores, digamos *Tipo de edificio religioso*, tendríamos que tener otra tabla donde se almacenase los tipos de edificios religiosos y así sucesivamente. Este modelo es válido, pero terminaríamos con una base de datos con cientos de tablas.

Otra manera a bordar el problema es crear una única tabla, denominada *Valores Cuadros Opciones* por ejemplo en la que se almacenarán los distintos valores que pueden tener los cuadros de opciones de un campo de una determinada tabla. Esta tabla tendría un aspecto como el siguiente:

Tabla	Campo	Valor	Título	Descripción
Instalaciones Deportivas	Tipo de actividad	24537	Fútbol	Campo de fútbol
Instalaciones Deportivas	Tipo de actividad	642	Baloncesto	Cancha de baloncesto
Instalaciones Deportivas	Tipo de actividad	5	Voleibol	Campo de voleibol
Instalaciones Deportivas	Tipo de actividad	2247	Vóley playa	Zona en la playa donde se practica el Vóley Playa
Instalaciones Deportivas	Tipo de actividad	9087765	Tenis	Cancha de tenis
Edificios Públicos	Tipo de edificio religioso	1	Iglesia	Selecciona esta opción si el edificio es una iglesia o una catedral.
Edificios Públicos	Tipo de edificio religioso	2	Mezquita	Selecciona esta opción si el edificio es una mezquita.
Instalaciones Deportivas	Tipo de actividad	1	Petanca	Lugar donde los jubilados juegan a la petanca

Si te fijas en la tabla anterior los registros no tienen por qué estar continuos en la tabla: tenemos el valor *iglesia* y *mezquita* entre *tenis* y *petanca*. Esto da igual, porque podemos filtrar con una consulta SQL los valores para el campo *Tipo de actividad* de la tabla *Instalaciones Deportivas* con la consulta:

```
SELECT Valor, Título, Descripción
  FROM [CompatibilidadCATDBS].[dbo].[Valores Cuadros Opciones]
 WHERE Tabla = 'Instalaciones Deportivas'
   AND Campo = 'Tipo de actividad'
```

Esta consulta devuelve una tabla con el mismo formato que espera la propiedad *valueList* de los campos en los esquemas de tablas en la base de datos.

Hemos visto que se puede asignar en el atributo *valueList* del campo de una tabla una lista de valores compuestos por la tripla (Valor|Título|Descripción), pero también podemos indicar que queremos crear una consulta SQL.

Esta consulta deberá generar siempre como resultado una tabla con tres campos (sin importar el nombre de estos campos) con la condición de que en el primer campo se almacenarán los valores, en el segundo campo los títulos y en el tercer campo las descripciones de los posibles valores a añadir en la lista.

Y eso es precisamente lo que obtenemos al ejecutar la consulta SQL anterior: una tabla con tres campos en el que en el primer campo se almacenarán los valores, en el segundo los títulos y en el tercero una ayuda para el operador.

Tan solo nos queda indicarle a Digi3D 2011 que el valor del atributo *valueList* para el campo es una consulta SQL, y eso lo podemos hacer poniendo la palabra especial %SQL% al comienzo del valor del atributo *valueList*.

Si Digi3D 2011 se encuentra un valor en *valueList* y este comienza por la palabra %SQL%, entenderá que el resto del valor almacenado en *valueList* es una consulta SQL que genera una tabla de estas características. **Recuerda que debe haber un espacio entre la palabra %SQL% y la consulta SQL.**

De modo que si nuestra base de datos tiene la tabla *Valores Cuadros Opciones* con los valores enumerados más arriba, el valor del atributo *valueList* para el campo *Tipo de actividad* de la tabla *Viales* sería:

```
%SQL% SELECT Valor, Título, Descripción FROM [CompatibilidadCATDBS].[dbo].[Valores Cuadros Opciones] WHERE Tabla = 'Instalaciones Deportivas' AND Campo = 'Tipo de actividad'
```

1. Ejecutamos *Microsoft SQL Server Management Studio*, nos conectamos con nuestro servidor de bases de datos.
2. Pulsamos Ctrl+O o ejecutamos la opción del menú *File/Open/File* para abrir un archivo de guiones SQL.
3. Localizamos el archivo ruta [ruta al laboratorio de base de datos]\Laboratorio 22\Tabla Valores Cuadros Opciones.sql y lo abrimos.
4. Comprobamos que el contenido del archivo contiene instrucciones SQL para crear la tabla *Valores Cuadros Opciones* y rellenarla con información.
5. Pulsamos el botón *Execute* de la barra de herramientas *SQL Editor* de *Microsoft SQL Server Management Studio*.
6. Abrimos la rama *Tables* de la base de datos *CompatibilidadCATDBS* en la ventana *Object Explorer* para comprobar que nuestra base de datos ahora tiene la tabla *Valores Cuadros Opciones*. Si no vemos la tabla, es que *Microsoft SQL Server Management Studio* no ha refrescado el listado de tablas de la base de datos. Para forzarle a que regenere el listado de tablas, seleccionamos la rama *Tables* y pulsamos el botón *Refresh* de la barra de herramientas de la ventana *Object Explorer*.
7. Comprobamos el contenido de la tabla *Valores Cuadros Opciones* haciendo clic con el botón derecho del ratón en el nombre de la tabla y seleccionando la opción *Select Top 1000 Rows* del menú contextual para comprobar que el contenido coincide con el mostrado anteriormente al comienzo de este laboratorio.

8. Localizamos el archivo [ruta al laboratorio de base de datos]\Laboratorio 21\Laboratorio.tab.xml, lo abrimos con *Internet Explorer* o con el bloc de notas y comprobamos que en el campo *Tipo de Actividad* de la tabla *Instalaciones deportivas*, en el atributo *valueList* tenemos asignada la consulta SQL indicada anteriormente.
9. Ejecutamos Digi3D 2011, cargamos el archivo [ruta al laboratorio de base de datos]\Laboratorio 22\Modelo.bin, seleccionando como modelo de datos CATDBS y con la cadena de conexión Provider=SQLOLEDB.1;Integrated Security=SSPI;Persist Security Info=False;Initial Catalog=CompatibilidadCATDBS;Data Source=.\SQLEXPRES
10. Seleccionamos el código de Instalaciones deportivas.
11. Comprobamos que ahora al desplegar el cuadro de opciones *Tipo de Actividad* aparece Petanca entre las opciones.

Laboratorio 23 (Automatizando la asignación de atributos)

En este laboratorio vamos a ir un poco más lejos y vamos a solucionar el problema expuesto al comienzo del capítulo, en el que tenemos una tabla en la que para poder especificar un atributo, tenemos que llenar una determinada combinación de valores.

Recordemos la tabla:

Tipo de edificio	Subtipo de edificio	SubSubtipo de edificio
3. Público	3. Para edificios públicos:	...
4. Privado	1. Administración	...
	2. Policía	20. Para Museos públicos:
	3. Museo	4. Historia
	4. Religioso	5. Pintura
4. Para edificios privados:	4. Para edificios privados:	6. ...
	1. Banco	21. Para Edificios religiosos
	2. Oficina	1. Iglesia
		2. Mezquita
		3. Templo budista...

En este tipo de situaciones, lo ideal sería tener un script que rellene todos estos valores de forma automática para evitar errores e incrementar la productividad del operador.

Digi3D 2011 incorpora una nueva orden ASIGNA_ATRIBUTO que nos sirve para cambiar programáticamente el valor de un determinado campo. Esta orden modifica valores incluso de campos ocultos al operador e incluso marcados como solo lectura.

La orden espera que le pasemos parámetros, y su formato es el siguiente:

Parámetro 1	Parámetro 2	Parámetro 3
Nombre del campo	Valor a asignar	Booleano que indica si convertir el campo en un

campo de solo lectura.

De modo que si tuviéramos una tabla como la anterior y creáramos por ejemplo un menú de opciones con la opción *Templo Budista* podríamos hacer que al seleccionar la opción en el menú se ejecutaran las siguientes órdenes de Digi3D:

```
cod=Edificio
asigna_atributo="Tipo de edificio" 3 1
asigna_atributo=" Subtipo de edificio" 4 1
asigna_atributo=" SubSubtipo de edificio " 3 1
```

Y en la opción para digitalizar un banco nos encontraríamos con algo así:

```
cod=Edificio
asigna_atributo="Tipo de edificio" 4 1
asigna_atributo=" Subtipo de edificio" 2 1
asigna_atributo=" SubSubtipo de edificio " 0 1
```

1. Ejecutamos *Microsoft SQL Server Management Studio*, nos conectamos con nuestro servidor de bases de datos.
2. Pulsamos Ctrl+O o ejecutamos la opción del menú *File/Open/File* para abrir un archivo de guiones SQL.
3. Localizamos el archivo ruta [ruta al laboratorio de base de datos]\Laboratorio 23\Tabla Edificios.sql y lo abrimos.
4. Comprobamos que el contenido del archivo contiene instrucciones SQL para crear la tabla Edificios.
5. Pulsamos el botón *Execute* de la barra de herramientas *SQL Editor* de *Microsoft SQL Server Management Studio* para crear la tabla edificios.
6. Comprobamos que se ha creado correctamente la tabla edificios.
7. Añadimos el registro:

ID	Tabla
5	Edificios

A nuestra tabla CATDBS.

8. Localizamos el archivo [ruta al laboratorio de base de datos]\Laboratorio 23\Laboratorio.tab.xml para comprobar que define dos códigos “0” y *Edificio*. El código *Edificio* está enlazado con la tabla *Edificios*, y en el esquema de la tabla *Edificios* únicamente está marcado como visible el campo *Nombre*, los demás (IDEficio, Tipo de edificio, Subtipo de edificio y SubSubtipo de edificio) son visibles para este laboratorio, pero deberían ser invisibles para evitar distraer al operador.

9. Localizamos el archivo de menú [ruta al laboratorio de base de datos]\Laboratorio 22\Laboratorio.menu.xml para comprobar que define dos opciones, una titulada *Templo budista* que selecciona como código activo el código *Edificio* y que modifica los valores de los campos para reflejar en la tabla que el registro es el de un templo budista y otro con título *Banco* que modifica los valores de los campos para reflejar que el registro es el de un banco.
10. Ejecutamos Digi3D 2011, cargamos el archivo [ruta al laboratorio de base de datos]\Laboratorio 23\Modelo.bin, seleccionando como modelo de datos CATDBS y con la cadena de conexión Provider=SQLOLEDB.1; Integrated Security=SSPI; Persist Security Info=False; Initial Catalog=CompatibilidadCATDBS; Data Source=.\SQLEXPRESS
11. Desplegamos la sección Edificios del menú lateral (podemos hacer aparecer el menú lateral con la opción del menú *Ver/Ventana de menú*).
12. Seleccionamos la opción Banco y comprobamos que *Tipo de edificio* tiene asignado el valor 4 (edificio privado), Subtipo de edificio tiene asignado el valor 2 (Banco).
13. Tecleamos el nombre de un banco, por ejemplo *La Caixa* en el campo nombre y dibujamos una entidad.
14. Seleccionamos ahora la opción del menú *Templo budista* y comprobamos que los valores de los campos ahora reflejan la combinación para indicar que la próxima entidad que digitalicemos será un templo budista.
15. Tecleamos en *Nombre* el valor *Jade*.
16. Digitalizamos otra entidad.
17. Comprobamos en *Microsoft SQL Server Management Studio* que se han almacenado los datos correctamente.

Laboratorio 24 (Mostrando ayuda al usuario)

Quizás sea necesario mostrar una pequeña ayuda al usuario para que comprenda mejor qué valor almacenar en un determinado campo de la base de datos.

Si introducimos una pequeña ayuda en el atributo *description* de los nodos *field* en el esquema de una tabla en la tabla de códigos, cuando el usuario selecciona dicho nodo en la ventana de *Campos de la base de datos* o en el cuadro de diálogo *Editor de códigos* se mostrará esa ayuda en la parte inferior de la ventana de propiedades.

1. Localizamos el archivo [ruta al laboratorio de base de datos]\Laboratorio 23\Laboratorio.tab.xml para comprobar que los campos *Nombre* y *Tipo de actividad* de la tabla *Instalaciones Deportivas* tienen una pequeña descripción.
2. Ejecutamos Digi3D 2011, cargamos el archivo [ruta al laboratorio de base de datos]\Laboratorio 24\Modelo.bin, seleccionando como modelo de datos CATDBS y con la cadena de conexión Provider=SQLOLEDB.1; Integrated

```
Security=SSPI;Persist Security Info=False;Initial
Catalog=CompatibilidadCATDBS;Data Source=.\SQLEXPRES
```

3. Seleccionamos como código activo el código 1 (Recintos deportivos) y comprobamos que la ventana *Campos de la base de datos* nos muestra los campos *Nombre* y *Tipo de Actividad*.
4. Si hacemos clic en cualquiera de estos dos campos, comprobaremos que Digi3D 2011 muestra en la parte inferior de la ventana de propiedades la descripción indicada en la tabla de códigos.

Laboratorio 25 (Cambiando el título de los campos)

Podemos indicarle a Digi3D 2011 que cambie el nombre de los campos mostrados en las ventanas de *Campos de la base de datos* y del *Editor de códigos*. Quizás el nombre de un campo es difícil de entender por parte del usuario, como CP3OXXT94 o quizás los nombres de los campos de la base de datos estén en un determinado idioma y nos interese mostrárselos al usuario en otro idioma.

En este laboratorio vamos a mostrar al usuario la base de datos en inglés.

1. Localizamos el archivo [ruta al laboratorio de base de datos]\Laboratorio 24\Laboratorio.tab.xml para comprobar que los campos *Nombre* y *Tipo de actividad* de la tabla *Instalaciones Deportivas* definen el atributo *Title* que es el título que utilizará Digi3D 2011 cuando muestre esos campos. Comprobaremos que el texto almacenado en los atributos Title está en inglés.
Además la descripción del código 1 está también en inglés, y si editamos el archivo Laboratorio.menu.xml comprobaremos que el menú también está en inglés.
Ejecutamos Digi3D 2011, cargamos el archivo [ruta al laboratorio de base de datos]\Laboratorio 25\Modelo.bin, seleccionando como modelo de datos CATDBS y con la cadena de conexión Provider=SQLOLEDB.1;Integrated Security=SSPI;Persist Security Info=False;Initial Catalog=CompatibilidadCATDBS;Data Source=.\SQLEXPRES
2. Seleccionamos como código activo el código 1 (Sports installations) y comprobamos que la ventana *Campos de la base de datos* no nos muestra los campos *Nombre* y *Tipo de Actividad*, sino *Name* y *Activity type*.

Configurando restricciones en el servidor

Si no sabes TSQL, es el momento de que saltes al Apéndice 2 para aprender lo que es un desencadenador. Una vez finalizados todas las prácticas del apéndice, continúa con este capítulo.

Una vez configurada la tabla de códigos podemos empezar a imponer restricciones para el modelo de datos. Estas restricciones se pueden imponer mediante desencadenadores (*triggers*).

Podemos añadir desencadenadores que se ejecuten al insertar datos en una determinada tabla que analicen la información que se acaba de insertar y que determinen si la entidad es prescindible o si los datos son o no correctos.

Laboratorio 26 (Controlando el perímetro de las entidades)

En este laboratorio vamos a crear una tabla nueva *Hidrografía* en la que se almacenarán los datos de un río (ID, Nombre y Perímetro) y añadiremos un desencadenador que se ejecute al insertar datos en esta tabla. El desencadenador analizará el perímetro de la línea que se acaba de digitalizar y si este es inferior a 100 metros devolverá un mensaje de error al usuario para que descarte la entidad.

1. Ejecuta Microsoft SQL Server Management Studio y conéctate al servidor de bases de datos.
2. Pulsa el botón *New Query* para crear una ventana de consultas nueva.
3. Comprueba si existe la tabla Hidrografía. Si es así, salta al punto 4.
Teclea y ejecuta el siguiente código para crear la tabla hidrografía:

```
USE CompatibilidadCATDBS
GO

CREATE TABLE Hidrografía
(
    ID int not null identity(1,1) primary key,
    Nombre varchar(64),
    Perímetro float
)
GO
```

4. Comprueba si la tabla Hidrografía tiene asociado el trigger TR_Hidrografía (lo puedes hacer abriendo la tabla *Databases/CompatibilidadCATDBS/Tables dbo.Hidrografía/Triggers* en la ventana *Object Explorer*). Si la tabla ya tiene el trigger salta al punto 5.
Teclea y ejecuta el siguiente código para crear el trigger:

```
BEGIN
    DECLARE @perímetro float

    SELECT      @perímetro = Perímetro
    FROM        INSERTED

    IF @perímetro < 100
    BEGIN
        ROLLBACK TRANSACTION
```

```

        RAISERROR ('No es necesario registrar esta
entidad', 16, 1)
      END
END
GO

```

- Ahora comprueba si la tabla CATDBS incluye la tabla Hidrografía. Si no es así, teclea y ejecuta lo siguiente:

```

INSERT INTO CATDBS (ID, Tabla)
VALUES (1 + (SELECT MAX(ID) FROM CATDBS), 'Hidrografía')
GO

```

- Ahora localiza el archivo [ruta al laboratorio de base de datos]\Laboratorio 26\Laboratorio.tab.xml para comprobar que incluye un código 3 (Ríos) enlazado con la tabla Hidrografía y en el esquema de dicha tabla se instruye a Digi3D 2011 para que rellene automáticamente el campo Perímetro con el perímetro de la entidad.
- Ejecuta Digi3D 2011 abriendo el archivo de dibujo [ruta al laboratorio de base de datos]\Laboratorio 26\Modelo.bin
- Selecciona el código de Río.
- Dibuja una línea con un perímetro inferior a los 100 metros para comprobar cómo Digi3D 2011 muestra un cuadro de diálogo con el mensaje lanzado desde el servidor con la instrucción RAISERROR. Pulsa el botón eliminar entidad.
- Vuelve a Microsoft SQL Server Management Studio para comprobar que el registro no se ha añadido.
- Dibuja ahora una línea con un perímetro superior a 100 metros para comprobar que Digi3D 2011 no nos muestra ningún mensaje de error. Comprueba en la base de datos que se ha creado el registro sin problemas.

Laboratorio 27 (Controlando el ancho de una vía y su velocidad máxima)

En este laboratorio vamos a comprobar cómo nos comunica Digi3D errores enviados por un desencadenador.

Cargaremos un modelo con una tabla de códigos que nos va a permitir indicar el ancho de la vía así como el número de carriles y la velocidad, introduciremos datos incorrectos e intentaremos almacenar una entidad para comprobar las distintas opciones que nos permite hacer Digi3D.

- Ejecutamos Microsoft SQL Server Management Studio y nos conectamos con nuestra base de datos.
 - Como nuestra tabla *Viales* ya tiene muchos campos, vamos a crear una nueva que denominaremos *Viales2* y un desencadenador *TR_Viales2*, pero esta vez con código SQL.
- Abrimos el archivo [ruta al laboratorio de base de datos]\Laboratorio 23\Crear Tabla y

Trigger.sql y leemos el código fuente para familiarizarnos con lo que hace el programa. Básicamente destruye la tabla Viales2 si esta existiera y la vuelve a crear. Además destruye el trigger TR_Viales2 en caso de existirlo y lo vuelve a crear.

14. Pulsamos el botón *Execute* de la barra de herramientas *SQL Editor* y comprobamos que *Microsoft SQL Server Management Studio* nos muestra en la ventana de salida un mensaje indicando que el programa se ha ejecutado correctamente.
15. Ejecutamos Digi3D 2011 abriendo el archivo de dibujo [ruta al laboratorio de base de datos]\Laboratorio 26\Modelo.bin
16. Seleciconamos el código *Calle* y en la ventana de *Campos de la base de datos* Tecleamos los siguientes valores

Campo	Valor
Nombre	N-I
Ancho	1.0
Carriles	4
Velocidad	120

17. Dibujamos una línea.
18. Al finalizar la línea, Digi3D 2011 intentará almacenar la información en la tabla *Viales2* de la base de datos. Como la tabla tiene asociado un desencadenador, este se ejecutará y descubrirá que el ancho de la carretera no es válido, ya que la carretera deberá tener al menos $3.5 * \text{número de carriles}$ metro de ancho. Digi3D 2011 nos muestra un cuadro de diálogo indicando el mensaje de error devuelto por el servidor de bases de datos: *La vía no es suficientemente ancha para el número de carriles* y nos facilita 4 opciones.
19. Volvemos a Microsoft SQL Server Management Studio para comprobar que el registro no se ha añadido a la tabla ejecutando en una consola el siguiente programa:

```
USE CompatibilidadCATDBS
SELECT * FROM Viales2
```

20. Volvemos a Digi3D 2011 y pulsamos el botón titulado *Eliminar la entidad*.

El proceso de almacenar una entidad en profundidad

Te adelanto que esta sección es complicada. Si no quieres profundizar tanto, salta a la siguiente sección. Es posible que tengas que leer esta sección varias veces para entenderla bien.

Si analizamos el ejemplo anterior, podemos intuir qué hace Digi3D cuando se almacena una entidad: recorre todos los códigos de la entidad y almacena los datos de aquellos que tengan enlace con base de datos en la base de datos para a continuación modificar los valores Tabla e ID de cada código y por último almacenar la entidad en el archivo de dibujo.

Pero la vida real no es tan sencilla, porque podemos encontrarnos con problemas a la hora de almacenar la entidad. Podemos encontrarnos con problemas al asignar un valor en un campo, o podemos encontrarnos problemas al almacenar el registro en la base de datos porque o no tenemos permiso o algún campo tiene un valor inválido.

A continuación te muestro en pseudocódigo qué hace **Digi3D** 2011 cuando le ordenamos almacenar una entidad:

```

procedimiento Guardar EntidadEnArchivo(entidad)
    si ArchivoEnlazadoConBaseDeDatos entonces
        para cada código ε entidad
            si ID del código = SIN_ENLACE_BBDD entonces
                si código está asociado con una tabla(código) entonces
                    datos ← DatosTecleadosPorUsuarioParaCodigo(código)
                    si DetectadoErrorAI(GuardarDatosEnTabla(código, datos))
                        acción = DialogoNoHaIntroducidoCampoObligatorio()
                        seleccionar acción
                        caso VolverAProyectar
                            fin caso
                        caso ConservarLaEntidad
                            OrdenarAlExportadorQueAlmaceneLaEntidad(entidad)
                            salir del procedimiento
                        caso EliminarLaEntidad
                            salir del procedimiento
                    fin si
                fin si
            fin si
        fin para cada
    fin si

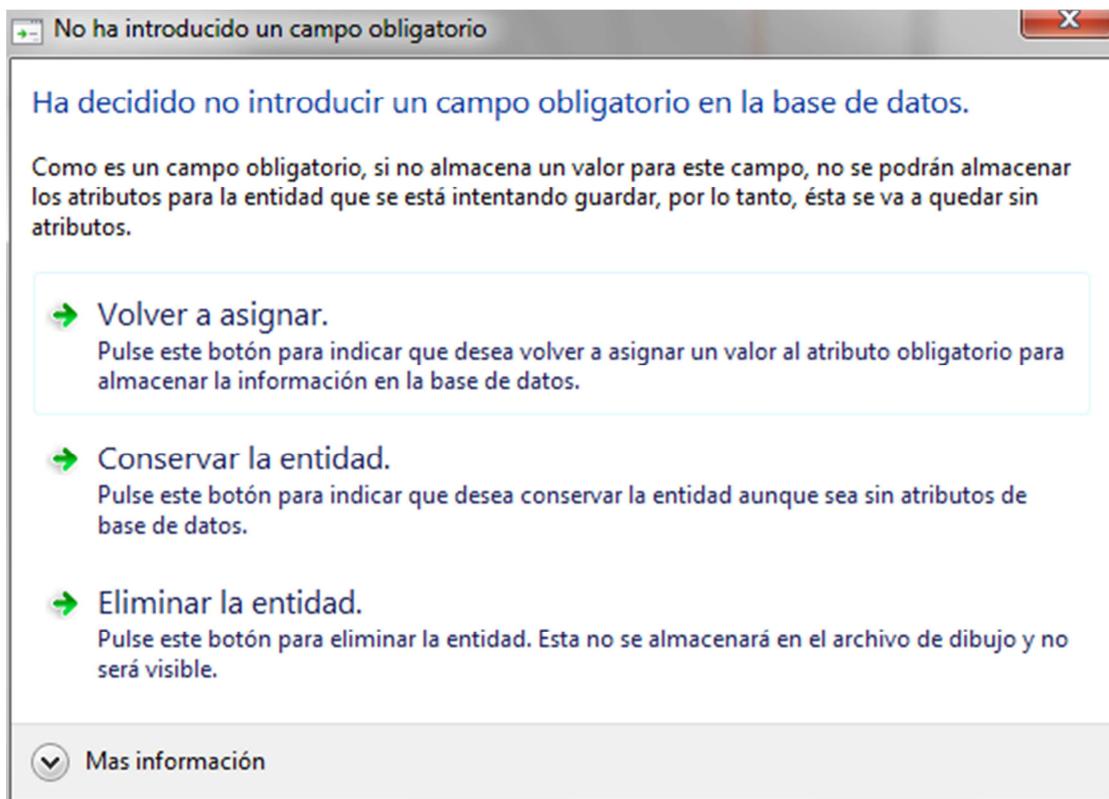
```

*OrdenarAlExportadorQueAlmaceneLaEntidad(*entidad*)
fin procedimiento*

Cuando se almacena una entidad, Digi3D 2011 ejecuta el procedimiento *GuardarEntidadEnArchivo*.

Si el archivo donde se va a almacenar la entidad está enlazado con una base de datos, se recorrerán todos sus códigos para averiguar si tienen enlace con base de datos y se llamará a la función *GuardarDatosEnTabla* (que explicaré a continuación) por cada código.

Si la función *GuardarDatosEnTabla* devuelve *falso* se mostrará un cuadro de diálogo al usuario preguntándole qué hacer:



El usuario tiene tres opciones:

1. *Volver a asignar*: se volverá a llamar a *GuardarDatosEnTabla*
2. *Conservar la entidad*, se almacenará la entidad en el archivo de dibujo y se finalizará inmediatamente el procedimiento.
3. *Eliminar la entidad*: Se finalizará inmediatamente el procedimiento sin almacenar la entidad en el archivo de dibujo.

Imagínate que tenemos una entidad con 20 códigos, todos ellos con enlace a base de datos. Si al almacenarla, en la iteración 18 la función *GuardarDatosEnTabla* devuelve *falso*, Digi3D 2011 mostrará al usuario el cuadro de dialogo mostrado anteriormente. Tanto si el usuario selecciona *Conservar la entidad* como *Eliminar la entidad* el programa no eliminará los

registros que ya había almacenado en la base de datos, de modo que la información de los 17 códigos anteriores **se conservarán** en la base de datos.

Ahora vamos por qué es posible que la función *GuardarDatosEnTabla* devuelva verdadero o falso:

```

función GuardarDatosEnTabla(código, datos)

    // Fase 1
    registro ← nuevo registro en la base de datos

    // Fase 2
    para cada campo en datos
        si no AsignaValorEnRegistro(registro, valor para el campo según datos)
            devolver falso
        fin si
    fin para cada

    // Fase 3
    número_de_registro, acción = AlmacenarRegitroEnBaseDatos(registro)
    seleccionar acción
        caso verdadero
            código.Tabla ← ObtenerNúmeroDeTabla(datos)
            código.Id ← ObtenerNúmeroDeRegistro(registro)
            devolver verdadero
        caso falso
            acción ← DialogoModeloDatosNoPermiteAlmacenarRegisro
            seleccionar acción
                caso EditarLosAtributos
                    acción ← MostrarAlUsuarioEditorAtributos(datos)
                    si acción = verdadero
                        devolver GuardarDatosEnTabla(código, datos)
                    si no
                        devolver falso
                    fin si
                caso VolverAProbar
                    devolver GuardarDatosEnTabla(código, datos)
                caso ConservarLaEntidad
                    devolver verdadero
                caso EliminarLaEntidad
                    devolver falso
            fin seleccionar
        fin seleccionar
    fin función

```

La función *GuardarDatosEnTabla* está dividida en tres fases:

1. Ordenar a la base de datos que cree un registro vacío.
2. Rellenar el registro con la información que ha tecleado el usuario en la ventana *Campos de la base de datos*.
3. Almacenar el registro en la base de datos.

La primera fase no puede provocar errores, sin embargo la segunda y tercera sí.

En la segunda fase, Digi3D 2011 recorre todos los campos que ha llenado el usuario en la ventana de *Campos de la base de datos* (representado por la variable *datos*) en el pseudocódigo llamando a la función *AsignaValorEnRegistro* .

Si esta función devuelve falso, la función *GuardarDatosEnTabla* finaliza inmediatamente devolviendo falso, desencadenando que se le muestre al usuario el cuadro de diálogo *No ha introducido un campo obligatorio*.

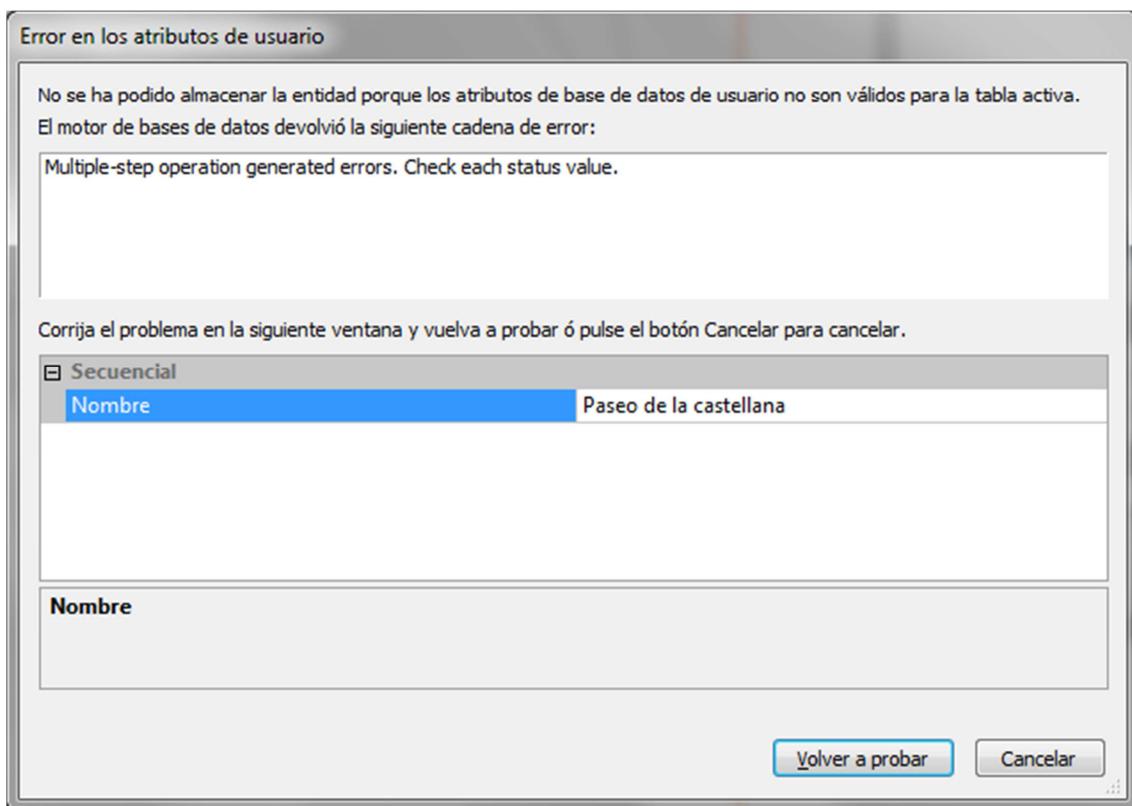
Antes de continuar con la fase 3 vamos a estudiar qué hace la función *AsignaValorEnRegistro*:

```
función AsignaValorEnRegistro(registro, valor)
    si detectado error al asignar valor del campo en registro(registro, valor) entonces
        valor nuevo, acción ← DialogoErrorEnLosAtributosDeUsuario
        seleccionar acción
            caso Aceptar
                devolver AsignaValorEnRegistro(registro, valor nuevo)
            caso Cancelar
                devolver falso
            fin seleccionar
        si no
            devolver verdadero
        fin si
    fin función
```

Ésta función asigna el valor almacenado por el usuario en el campo correspondiente del registro que vamos a almacenar en la base de datos.

Esta operación puede generar fallos, por ejemplo, si el campo en la base de datos tiene un tamaño de dos caracteres y el usuario ha tecleado una cadena con 4 caracteres, o si el campo es numérico y el usuario ha tecleado una cadena alfanumérica, por ejemplo “ABCDE”, o ...

Si se detecta un fallo en esta fase, Digi3D 2011 mostrará un cuadro de diálogo indicándole al usuario que el valor almacenado en ese campo es incorrecto, invitándole a que introduzca un valor correcto:



El cuadro de diálogo le muestra al usuario el campo que ha generado el error.

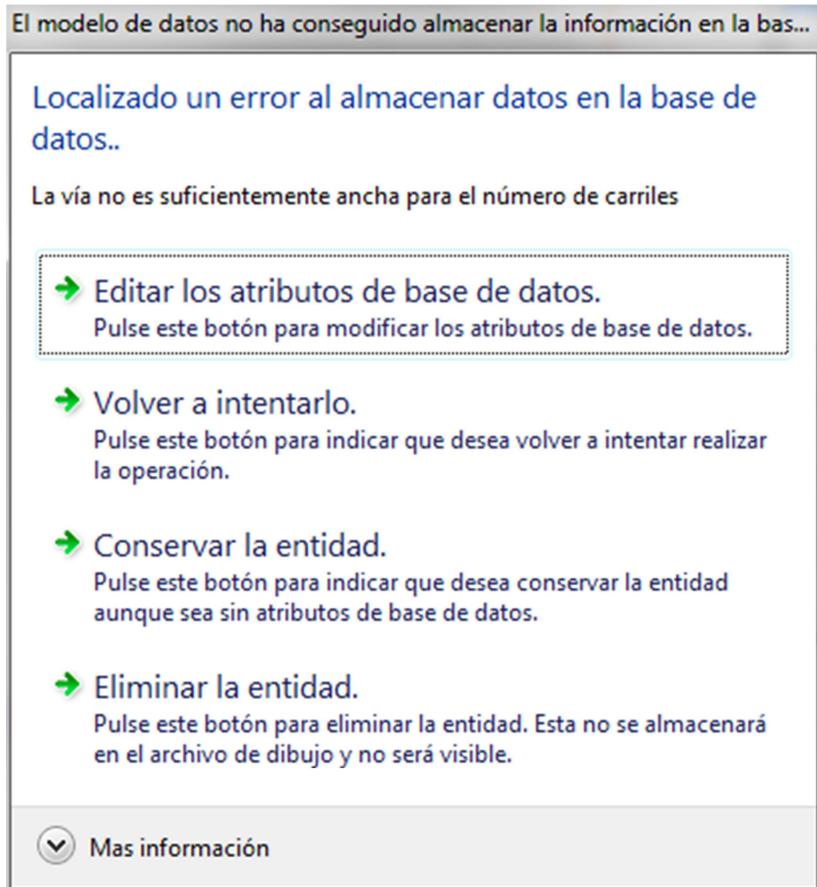
El usuario puede optar por cambiar el valor y pulsar el botón *Volver a probar* o cancelar.

Si pulsa el botón *Volver a probar* la función *AsignaValorEnRegistro* volverá a llamarse recursivamente. Si vuelve a fallar, se volverá a mostrar este cuadro de diálogo y así sucesivamente hasta que la asignación sea correcta (en cuyo caso la función devolverá *verdadero*) o hasta que el usuario pulse Cancelar (en cuyo caso devolverá falso).

Una vez conseguido asignar todos los campos tecleados por el usuario en el registro nuevo, saltamos a la fase 3 de la función *GuardarDatosEnTabla*.

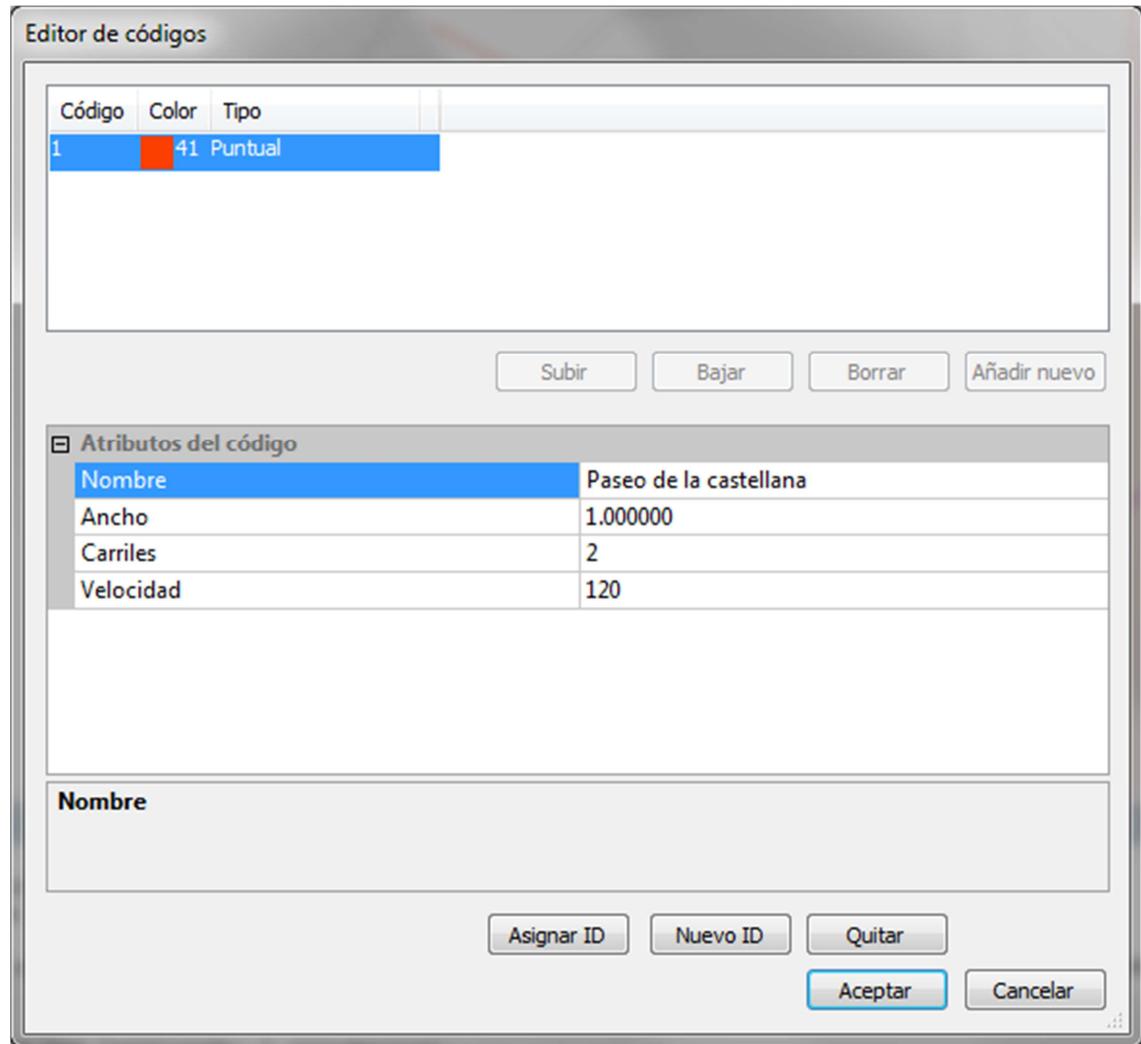
Esta fase consiste en ordenarle a la base de datos que almacene la información en la base de datos.

Esta operación puede generar errores también: el usuario no tiene permisos de escritura en la tabla, alguno de los parámetros no es correcto,... Si se detecta un error, el programa mostrará el cuadro de diálogo *El modelo de datos no ha conseguido almacenar la información en la base de datos*:



El usuario puede seleccionar una de las siguientes opciones:

1. Editar los atributos de la base de datos. Si el usuario pulsa esta opción, el programa mostrará al usuario un cuadro de diálogo con todos los atributos de la entidad para que modifique los parámetros necesarios según el modelo de datos.
Ej: En la captura anterior, el mensaje indica que *La vía no es suficientemente ancha para el número de carriles*. Si sabemos que el modelo de datos nos exige que el ancho de la vía tendrá que ser al menos $3.5 \times$ número de carriles, podríamos pulsar el botón *Editar los atribubos de la base de datos* para que el programa nos muestre el siguiente cuadro de diálogo:



Y podríamos solucionar el problema tecleando por ejemplo 7 en el campo Ancho o 1 en el campo Carriles.

Si el usuario acepta este cuadro de diálogo se volverá a probar a almacenar el registro.

Si lo cancela, se devolverá *falso* al procedimiento [GuardarEntidadEnArchivo](#).

2. Volver a intentarlo: Simplemente se volverá a intentar almacenar el registro sin hacer nada especial. Esta opción sirve en los casos en los que no se ha podido almacenar un registro porque la tabla estaba bloqueada, o porque ha fallado la conexión con la base de datos... Una vez solucionado el problema en el servidor (desbloqueando la tabla por ejemplo) podemos pulsar este botón para volver a intentarlo.
3. Conservar la entidad: Se devolverá verdadero al procedimiento [GuardarEntidadEnArchivo](#) para que este continúe intentando almacenar los registros del resto de códigos de la entidad.
4. Eliminar la entidad: Se devolverá *falso* al procedimiento [GuardarEntidadEnArchivo](#) para que este elimine la entidad.

5. Acceso a base de datos mediante programación .NET

En este capítulo vamos a aprender a realizar aplicaciones DigiNG con acceso a datos.

Aprenderemos a seleccionar como códigos activos códigos que enlazan con registros existentes en la base de datos, aprenderemos a localizar los atributos de base de datos de las entidades del archivo de dibujo, aprenderemos a localizar las entidades que enlazan con un determinado registro de base de datos, ...

Tras finalizar este capítulo sabrás hacer consultas tipo SIG en DigiNG. El límite lo pone tu imaginación.

Copias de seguridad mejoradas

Digi3D 2011 implementa un nuevo sistema de copias de seguridad que mejora considerablemente el sistema de copias de seguridad de versiones anteriores.

Las copias de seguridad de versiones anteriores consistían en crear un archivo con extensión .bik en el mismo directorio que el archivo de dibujo.

Estas copias de seguridad no eran incrementales, el archivo de copia de seguridad tenía siempre el mismo nombre, por lo tanto si cometíamos un error en el modelo y se guardaba una copia de seguridad, no teníamos forma de recuperar la información.

El nuevo sistema de copias de seguridad incorpora dos características nuevas:

- a) Permite indicar el nombre del archivo a generar en el repositorio de copias de seguridad. Este nombre puede ser variable, gracias a la característica de *sustituidores*.
- b) Permite indicar que se generen copias de seguridad al cerrar el archivo de dibujo, para mantener un histórico de copias de seguridad.

La configuración de copias de seguridad está centralizada en el cuadro de diálogo *Herramientas/Configuración*, en la sección *Copia de seguridad* en la que encontramos tres campos:

1. Destino de copias de seguridad.
2. Generar copia de seguridad cada (minutos).
3. Generar copia de seguridad al salir.

En destino de copias de seguridad indicaremos el nombre del archivo a generar.

En el campo *Generar copia de seguridad cada (minutos)* indicaremos cada cuántos minutos queremos que el programa realice una copia de seguridad. Si no deseamos que el programa genere copias de seguridad, podemos poner en este campo el valor “0”.

En el campo *Generar copia de seguridad al salir* indicaremos si queremos que el programa realice una copia de seguridad cada vez que cerremos el modelo.

Laboratorio 1 (Creando un esquema de copias de seguridad muy básico)

En este laboratorio vamos a crear el sistema de copias de seguridad más básico posible, que consistirá en indicarle al programa que cree un archivo denominado *Copia de seguridad.bin* en la carpeta *c:\Copias de seguridad*. Configuraremos el programa para que realice copias de seguridad al cerrar la ventana de dibujo.

Comprobaremos que a pesar de que no existe el directorio c:\copias de seguridad, el programa lo va a crear de forma automática.

1. Comprueba que no existe la carpeta c:\Copias de seguridad. Si ya existe, **no la elimines**, pues tendrá copias de seguridad. Si no existe comprobarás que Digi3D 2011 la va a crear automáticamente.
2. Ejecuta Digi3D 2011.
3. Cancela el cuadro de diálogo Nuevo Proyecto.
4. Abre el cuadro de diálogo de configuración mediante la opción del menú *Herramientas/Configuración*.
5. En el campo *Destino de copias de seguridad* teclea *c:\Copias de seguridad\Copia de seguridad.bin*.
6. En el campo *Generar copia de seguridad al salir* selecciona la opción *Si*.
7. Acepta el cuadro de diálogo de configuración.
8. Abre un archivo de dibujo nuevo, con extensión .bin.
9. Dibuja alguna entidad.
10. Cierra el archivo de dibujo.
11. Comprueba que el programa ha creado el archivo *c:\Copias de seguridad\Copia de seguridad.bin*.

Laboratorio 2 (Error al crear el directorio de copias de seguridad)

En este laboratorio vamos a configurar Digi3D 2011 para que almacene copias de seguridad en un directorio inalcanzable. Configuraremos como unidad destino L: (suponiendo que el ordenador donde se está ejecutando el programa no dispone de esa unidad) para comprobar cómo comunica Digi3D 2001 el error.

1. Ejecuta Digi3D 2011.
2. Cancela el cuadro de diálogo Nuevo Proyecto.
3. Abre el cuadro de diálogo de configuración mediante la opción del menú *Herramientas/Configuración*.
4. En el campo *Destino de copias de seguridad* teclea ⁸*L:\Copias de seguridad\Copia de seguridad.bin*.
5. En el campo *Generar copia de seguridad al salir* selecciona la opción *Si*.
6. Acepta el cuadro de diálogo de configuración.
7. Abre un archivo de dibujo nuevo, con extensión .bin.
8. Dibuja alguna entidad.
9. Cierra el archivo de dibujo.
10. Comprueba que el programa muestra un cuadro de diálogo indicando que ha sido imposible crear el directorio para ubicar la copia de seguridad.
Si el problema hubiera sido ocasionado porque el dispositivo de copias de seguridad está desconectado, podríamos pulsar el botón titulado “*Volver a probar*”.
11. Pulsa el botón titulado *Cancelar*

⁸ Si tu equipo tiene una unidad L, cambia el valor L por el de una unidad inexistente.

Laboratorio 3 (Mejorando el sistema de copias de seguridad)

En el Laboratorio 1 indicamos que el archivo a crear en la carpeta *c:\Copia de seguridad* siempre es: *Copia de seguridad.bin*, de modo que en este archivo tendremos siempre como copia de seguridad el último archivo cargado en la aplicación.

Veámoslo con un ejemplo:

1. Ejecuta Digi3D 2011 y configura en el campo *Destino de copias de seguridad: c:\Copias de seguridad\Copia de seguridad.bin*
2. Abre un archivo de dibujo *Círculo.bin* en el directorio de trabajo.
3. Si el archivo no existía, dibuja un círculo con la orden CIR2P.
4. Cierra el archivo de dibujo.
5. Abre el archivo de dibujo *Cuadrado.bin* en el directorio de trabajo.
6. Si el archivo no existía, dibuja un cuadrado con la orden 2P_AA o pulsando cualquier botón de la barra de herramientas Cuadrados.
7. Cierra el archivo de dibujo.
8. Abre el archivo de dibujo *c:\Copias de seguridad\Copia de seguridad.bin*. Comprobarás que es un cuadrado. Hemos perdido la copia de seguridad del círculo.
9. Cierra el archivo de dibujo.
10. Aparecerá un mensaje de error. Pulsa el botón titulado *Más información* para comprobar la causa del error: No se puede hacer una copia de seguridad con el mismo nombre del archivo que tenemos abierto.
11. Pulsa el botón *Cancelar* para cancelar la creación de la copia de seguridad.

Podemos utilizar la característica *sustituidores*⁹ para mejorar nuestro sistema de copias de seguridad.

Vamos a configurar Digi3D 2011 para que almacene en el directorio *c:\Copias de seguridad* una copia del archivo de dibujo con el mismo nombre que el archivo de dibujo cargado, es decir, que al cerrar el archivo de dibujo *Círculo.bin* se genere una copia de seguridad denominada *c:\Copias de seguridad\Círculo.bin* y al cerrar el archivo de dibujo *Cuadrado.bin* se genere una copia de seguridad denominada *c:\Copias de seguridad\Cuadrado.bin*.

1. Ejecuta Digi3D 2011 y configura en el campo *Destino de copias de seguridad: c:\Copias de seguridad\\$(DrawingFileName)*
2. Abre un archivo de dibujo *Círculo.bin* en el directorio de trabajo.
3. Si el archivo no existía, dibuja un círculo con la orden CIR2P.
4. Cierra el archivo de dibujo.
5. Abre el archivo de dibujo *Cuadrado.bin* en el directorio de trabajo.
6. Si el archivo no existía, dibuja un cuadrado con la orden 2P_AA o pulsando cualquier botón de la barra de herramientas Cuadrados.

⁹ Consulta el apéndice 1 para obtener información acerca de qué son los sustituidores y una referencia de todos los sustituidores posibles.

7. Cierra el archivo de dibujo.
8. Comprueba que ahora en el directorio *c:\Copias de seguridad* tenemos los dos archivos: *Círculo.bin* y *Cuadrado.bin*.

Laboratorio 4 (Copias de seguridad incrementales)

En este laboratorio vamos a hacer que el sistema de copias de seguridad nunca elimine copias de seguridad anteriores, para mantener un histórico de copias de seguridad.

Podemos aprovecharnos de los sustituidores *\$(DateUnderscore)¹⁰* y *\$(TimeUnderscore)* a la hora de dar un nombre al archivo de copia de seguridad.

En este laboratorio vamos a hacer que Digi3D 2011 cree copias de seguridad de los archivos en el directorio *c:\Copias de seguridad* con el nombre *Copia de seguridad el día dd_mm_aa a las hh_mm_ss de [nombre del archivo de dibujo]*.

1. Ejecuta Digi3D 2011 y configura en el campo *Destino de copias de seguridad: c:\Copias de seguridad\Copia de seguridad el día \$(DateUnderscore)a las \$(TimeUnderscore) de \$(DrawingFileName)*
2. Abre un archivo de dibujo *Círculo.bin* en el directorio de trabajo.
3. Si el archivo no existía, dibuja un círculo con la orden CIR2P.
4. Cierra el archivo de dibujo.
5. Abre el archivo de dibujo *Cuadrado.bin* en el directorio de trabajo.
6. Si el archivo no existía, dibuja un cuadrado con la orden 2P_AA o pulsando cualquier botón de la barra de herramientas Cuadrados.
7. Cierra el archivo de dibujo.
8. Comprueba que ahora en el directorio *c:\Copias de seguridad* tenemos los dos archivos: *Círculo.bin* y *Cuadrado.bin*.

Laboratorio 5 (Copias de seguridad ordenadas por día)

El sistema de copias de seguridad es capaz de crear múltiples directorios simultáneamente, de modo que podríamos configurar un sistema de copias de seguridad que almacene las copias de seguridad de cada día en un directorio correspondiente a dicho día.

¹⁰ Nota: No podemos utilizar los sustituidores *\$(Date)* y *\$(Time)* porque devuelven valores en los formatos dd:mm:aa y hh/mm/ss respectivamente, y está prohibido crear archivos cuyos nombres tengan los caracteres ":" y "/".

1. Ejecuta Digi3D 2011 y configura en el campo *Destino de copias de seguridad: c:\Copias de seguridad\\$(DateUnderscore)\Copia de seguridad a las \$(TimeUnderscore) de \$(DrawingFileName)*
2. Abre un archivo de dibujo *Círculo.bin* en el directorio de trabajo.
3. Si el archivo no existía, dibuja un círculo con la orden CIR2P.
4. Cierra el archivo de dibujo.
5. Abre el archivo de dibujo *Cuadrado.bin* en el directorio de trabajo.
6. Si el archivo no existía, dibuja un cuadrado con la orden 2P_AA o pulsando cualquier botón de la barra de herramientas Cuadrados.
7. Cierra el archivo de dibujo.

Comprueba que ahora en el directorio *c:\Copias de seguridad* el programa ha creado una carpeta cuyo nombre coincide con la fecha de hoy en la que tenemos una copia de seguridad de los archivos: *Círculo.bin* y *Cuadrado.bin*.

Laboratorio 6 (Copias de seguridad en la nube)

No tiene mucho sentido realizar copias de seguridad en el mismo en el que estamos utilizando el programa, ya que si el ordenador se estropea o *desaparece* habremos perdido todas las copias de seguridad.

Lo ideal es almacenar las copias de seguridad en un servidor, y yo personalmente ubicaría ese servidor fuera de las dependencias de la empresa.

A día de hoy existen numerosísimos servicios gratuitos de almacenamiento seguro en la nube. El más sencillo de usar (y por ello el año 2010 se obtuvo el premio al mejor servicio en la nube en España) es sin duda *DropBox*. Su éxito consiste en su integración con el sistema operativo.

Una vez creada la cuenta e instalado el software, disponemos de una carpeta en el ordenador con la cual podemos interactuar como si fuera una carpeta más, pero con la particularidad de que cada vez que copiamos algo en esta carpeta, el software se encarga de sincronizarlo con el servidor en Internet.

Además, si utilizamos la misma cuenta con varios ordenadores, al copiar algo en una carpeta, aparte de copiarse en Internet, se copia en los otros ordenadores.

Al instalar el programa, si realizamos la instalación típica, éste crea la carpeta como hija de tu carpeta de usuario.

En Windows 7, si el usuario se llama Jose Angel, la carpeta DropBox estará ubicada en *c:\Users\Jose Angel\Dropbox*.

En este laboratorio vamos a crear una cuenta de *DropBox* y configuraremos Digi3D 2011 para que realice copias en *DropBox*.

1. Abre tu navegador de internet y entra en la dirección <http://www.dropbox.com>

2. Selecciona la pestaña *Create an account*.
3. En *First name* pon tu nombre.
4. En *Last name* pon tus apellidos.
5. En *Email* pon tu dirección de correo electrónico.
6. En *Password* pon la contraseña que se utilizará para el cifrado de tus archivos así como para poder acceder al servicio.
7. Por último pulsa el botón *Create account*.
8. Localiza el enlace *Install Dropbox* o entra en <http://www.dropbox.com/install>
9. Descarga el programa pulsando el botón *Free Download*.
10. Una vez descargado, instálalo.
11. Selecciona la opción que indica que ya tienes cuenta en DropBox y pulsa siguiente.
12. Introduce tus credenciales y finaliza la instalación.
13. Ahora tienes el icono de Dropbox en el área de notificaciones de Windows. Al hacer doble clic en él, se abrirá la carpeta de Dropbox.
14. Ejecuta Digi3D 2011 y configura en el campo *Destino de copias de seguridad*: [ruta a tu carpeta dropbox]\Copias de seguridad\\$(DateUnderscore)\Copia de seguridad a las \$(TimeUnderscore) de \$(DrawingFileName).
Ejemplo: c:\Users\Jose Angel\Dropbox\\$(DateUnderscore)\Copia de seguridad a las \$(TimeUnderscore) de \$(DrawingFileName)

Apéndice 1: Sustituidores

Digi3D 2011 incorpora una nueva característica denominada *Sustituidores* que permite cambiar valores en tiempo real.

Los sustituidores no son más que palabras comodín que se sustituirán por un valor que normalmente se obtiene interrogando al sistema operativo.

Los sustituidores siguen el siguiente patrón ***\$(nombre_del_sustituidor)***

Los sustituidores implementados por Digi3D son los siguientes:

Nombre	Descripción	Valor típico
\$(Digi3DTables)	Directorio de tablas de Digi3D.	C:\ProgramData\
\$(Desktop)	Escritorio del usuario activo.	C:\Users\[usuario]\Desktop
\$(ProgramFiles)	Directorio donde se instalan las aplicaciones.	C:\Archivos de programa
\$(ApplicationData)	Directorio donde se almacenan las configuraciones de las distintas aplicaciones.	C:\ProgramData
\$(Temp)	Directorio donde se almacenan archivos temporales.	C:\Users\[usuario]\AppData\Local\Temp
\$(Username)	Nombre del usuario activo.	
\$(ComputerName)	Nombre del ordenador.	
\$(Date)	Fecha en formato dd/mm/aa	
\$(DateUnderscore)	Fecha en formato dd_mm_aa	
\$(Day)	Día	
\$(Month)	Mes	
\$(Year)	Año	
\$(Time)	Hora en formato	

	hh:mm:ss
\$(TimeUnderscore)	Hora en formato hh_mm_ss
\$(Hour)	Hora
\$(Minute)	Minutos
\$(Second)	Segundos
\$(UserProfile)	Carpeta personal del usuario activo. C:\Users\[usuario]
\$(PathOfDrawingFile)	Directorio del archivo de dibujo (sin incluir el nombre del archivo de dibujo)
\$(DrawingFilePath)	Directorio y nombre del archivo de dibujo con su extensión.
\$(DrawingFilePathNoExtension)	Directorio y nombre del archivo de dibujo sin extensión.
\$(DrawingFileName)	Nombre del archivo de dibujo con su extensión.
\$(DrawingFileNameNoExtension)	Nombre del archivo de dibujo sin extensión.
\$(DrawingFileExtension)	Extensión del archivo de dibujo.

Apéndice 2: Tutorial de TSQL

Microsoft SQL Server implementa una evolución del lenguaje SQL denominada Transact SQL, que básicamente transforma el lenguaje de consultas SQL en un lenguaje de programación. Se pueden declarar variables, dispone de instrucciones de control, bucles, funciones, procedimientos almacenados, triggers...

Aspectos generales del lenguaje TSQL

TSQL no es sensible al contexto, lo que significa que no se diferencian mayúsculas y minúsculas.

Se pueden agrupar varias instrucciones en una línea si las separamos por el carácter punto y coma (;).

Podemos poner comentarios de una única línea si ésta comienza por dos caracteres guión (--)

Podemos poner comentarios de múltiples líneas marcando el comienzo del comentario con los caracteres dividir seguido de asterisco /*) y finalizando el comentario con los caracteres asterisco seguido de dividir (*/).

Las cadenas de caracteres se delimitan siempre con comillas simples. Ej: 'Gran vía'

Impresión de información en la consola de Microsoft SQL Server Management Studio

Podemos utilizar la función PRINT [valor a imprimir] para imprimir un resultado en la consola de salida del programa.

```
PRINT 'Hola mundo'
```

Podemos concatenar con el signo más (+) varias cadenas para formar un resultado compuesto.

```
PRINT 'Hola' + ' ' + 'mundo'
```

Podemos imprimir el valor devuelto por una función (tanto de las suministradas por el propio SQL Server como las que desarrollemos nosotros) siempre y cuando el tipo devuelto por la función sea una cadena de caracteres.

```
PRINT 'La base de datos activa es: ' + DB_NAME()
```

Si la función no devuelve una cadena de caracteres, podremos utilizar la función `CAST` cuyo formato es:

```
CAST(<función> AS <tipo_destino>)
```

La función del sistema `GETDATE` nos devuelve la fecha en el servidor, pero el valor devuelto no es una cadena de caracteres, de modo que si queremos imprimir la fecha del servidor en la consola, tendremos que aplicarle un `CAST` a un tipo de datos de cadena de caracteres.

```
PRINT 'Ahora son las: ' + CAST(GETDATE() AS varchar(30))
```

No te preocupes por el tipo `varchar(30)` vas a aprender lo que significa a continuación.

Variables TSQL

En TSQL podemos declarar variables mediante la instrucción:

```
DECLARE @variable AS tipo_de_datos(tamaño_si_el_tipo_lo_requiere)
```

Los nombres de variables siempre deben comenzar con el carácter arroba (@).

tipo_de_datos puede ser uno de los siguientes tipos: bigint, numeric, bit, smallint, decimal, smallmoney, int, tinyint, money, float, real, date, datetimeoffset, datetime2, smalldatetime, datetime, time, char, varchar, nchar, nvarchar, ntext, binary, varbinary, image, cursor, timestamp, hierarchyid, uniqueidentifier, sql_variant, xml y table.

Algunos tipos de datos requieren que indiquemos un tamaño como por ejemplo los tipos cadena de caracteres (char, varchar, nchar, nvarchar) o los contenedores binarios (binary, varbinary, image).

Los tipos char y varchar se utilizan para almacenar cadenas de caracteres ASCII.

Los tipos nchar y nvarchar se utilizan para almacenar cadenas de caracteres Unicode.

Existe una diferencia entre char/nchar y varchar/nvarchar, y es que los primeros siempre ocupan en memoria el tamaño indicado en la declaración de la variable mientras que los segundos son variables.

Tipo	Valor almacenado	Tamaño ocupado en memoria
char(30)	'Hola'	30
varchar(30)	'Hola'	4

Si queremos declarar varias variables, utilizaremos el carácter coma (,) como separador.

```
DECLARE @nombre varchar(256)
```

```

DECLARE @velocidad int
o
DECLARE @nombre varchar(256),
@velocidad int

```

Es obligatorio declarar una variable antes de poder utilizarla (asignarle un valor o comprobar el valor que tiene).

Para asignar un valor a una variable lo podemos hacer con la instrucción SET:

```
SET @nombre_de_variable = valor | función
```

Valor puede ser una constante, otra variable, el resultado de una función o el resultado de una consulta SQL (lógicamente el resultado de la consulta SQL debe tener un único registro).

```

SET @velocidad = 120
SET @velocidad2 = @velocidad
SET @hora = GETDATE()
SET @nombre = (SELECT Nombre FROM Viales WHERE IDVial=1)

```

Podemos utilizar operadores +,-,*,/ y % (el operador % devuelve el resto de la división entera) para realizar operaciones aritméticas o concatenar cadenas de caracteres.

```
SET @velocidadActual = @velocidadActual + 1
```

También podemos asignar múltiples variables mediante la instrucción SELECT:

```

SELECT
    @[nombre variable 1]=campo1,
    @[nombre variable 2]=campo2,
    ...,
    @[nombre variable n]=campoN
FROM [tabla]
WHERE [condición que haga que la consulta SELECT devuelva un único registro]

```

Ejemplo:

```

SELECT      @nombre=Nombre ,
            @area=Area ,
            @hora=GETDATE()
FROM Viales
WHERE IDVial = 1

```

Laboratorio 1

En este laboratorio vamos a crear dos variables: la primera denominada *nombre* donde pretendemos almacenar el nombre de una vía.

La segunda denominada *identificadorVial* en la que almacenaremos el identificador de la entidad cuyo nombre queremos almacenar en la variable *nombre*.

Luego asignaremos en la variable *identificadorVial* el identificador de la vía que nos interesa y en la variable *nombre* el nombre de la vía que tiene dicho identificador de vía.

Para finalizar, imprimiremos el nombre de la vía.

1. Ejecutamos Microsoft SQL Server Management Studio, nos conectamos con nuestro servidor de bases de datos.
2. Pulsamos el botón *New Query* de la barra de herramientas *Standard*. Con eso conseguimos que Microsoft SQL Server Management Studio cree un archivo .sql vacío.
3. Desplegamos cuadro de opciones de la barra de herramientas *SQL Editor* y comprobamos que tiene tantas opciones como bases de datos tenga nuestro servidor de bases de datos.
Si tecleamos instrucciones TSQL en el archivo .sql que se acaba de crear y las ejecutamos, éstas se ejecutarán contra la base de datos que esté seleccionada en ese cuadro de opciones, por lo tanto, seleccionamos la base de datos *CompatibilidadSQL*.
4. Tecleamos en nuestro archivo .sql lo siguiente:

```
-- Declaramos las variables nombre e identificadorVial
DECLARE @nombre varchar(256),
        @identificadorVial int

-- Asignamos el identificador de vía que nos interesa
SET @identificadorVial = 1

        -- Asignamos en la variable nombre el nombre de la vía que
        -- nos interesa
SET @nombre =
        (SELECT Nombre FROM Viales WHERE IDVial = @identificadorVial )

-- Imprimimos el nombre
PRINT @nombre
```

5. Pulsamos el botón *Execute* de la barra de herramientas *SQL Editor* para comprobar que se ejecuta el conjunto de instrucciones y que el resultado es *Gran vía*.

Funciones TSQL

Podemos declarar funciones en nuestra base de datos. Las funciones son un conjunto de instrucciones a las que se le asigna un nombre que devuelven un valor. Este valor puede ser de

cualquier tipo. Además, pueden recibir parámetros para realizar las operaciones necesarias para poder devolver ese valor.

La forma de crear/modificar una función es:

```
CREATE/ALTER FUNCTION <nombre de la función>
(
    <parámetro 1>
    <parámetro 2>
    ...
    <parámetro n>
)
RETURNS
    <tipo>
AS
BEGIN
    <cuerpo de la función>
END
```

Para devolver un valor se utiliza la instrucción RETURN <valor_a_devolver>.

Por último, para poder ejecutar una función tendremos que utilizar su nombre completo, que consiste en el nombre del creador de la función (normalmente dbo de Database Owner) más el carácter punto (.) más el nombre de la función y a continuación sus parámetros entre paréntesis y separados por comas.

Laboratorio 2

En este laboratorio vamos a crear una función que nos devolverá el nombre de una vía en función de su identificador.

1. Ejecutamos Microsoft SQL Server Management Studio, nos conectamos con nuestro servidor de bases de datos.
2. En la ventana *Object Explorer* abrimos la rama *CompatibilidadCATDBS/Programmability/Functions/Scalar-valued functions*. Si localizamos la función *dbo.NombreVia*, la eliminamos con el botón derecho del ratón, seleccionando *Delete* en el menú contextual y aceptando el cuadro de diálogo que nos pide confirmación para eliminar la función.
3. Pulsamos el botón *New Query* de la barra de herramientas *Standard*. Con eso conseguimos que Microsoft SQL Server Management Studio cree un archivo .sql vacío.
4. Seleccionamos *CompatibilidadCATDBS* en el cuadro de opciones *Available Databases* de la barra de herramientas *SQL Editor*.
5. Tecleamos en nuestro archivo .sql lo siguiente:

```

CREATE FUNCTION NombreVia
(
    @id AS integer
)
RETURNS varchar(256)
AS
BEGIN
    -- Declaramos la variable donde se almacenar el
    -- nombre de la vía
    DECLARE @nombre varchar(256)

    -- Asignamos en la variable nombre el nombre de la vía
    -- que nos interesa
    SET @nombre =
        (SELECT Nombre FROM Viales WHERE IDVial = @id )

    -- Devolvemos el nombre de la vía
    RETURN @nombre
END

```

6. Pulsamos el botón *Execute* de la barra de herramientas *SQL Editor* para comprobar que Microsoft SQL Server Management Studio nos comunica que los comandos se han completado satisfactoriamente.
7. Seleccionamos la opción *Scalar-valued Functions* en la ventana *Object Explorer* y pulsamos el botón *Refresh* de la barra de herramientas de la ventana *Object Explorer* para comprobar que ahora nuestra base de datos tiene la función *dbo.NombreVia*.
8. Pulsamos el botón *New Query* de la barra de herramientas *Standard*. Ahora vamos a imprimir los nombres de las vías con identificadores 1, 2 y 3.
9. Tecleamos en nuestro archivo .sql lo siguiente:

```

PRINT dbo.NombreVia(1)
PRINT dbo.NombreVia(2)
PRINT dbo.NombreVia(3)

```

10. Pulsamos el botón *Execute* de la barra de herramientas *SQL Editor* para comprobar que Microsoft SQL Server Management Studio nos muestra los nombres de las vías 1, 2 y 3.

Bucles TSQL

TSQL dispone de instrucciones que nos permiten repetir una serie de instrucciones hasta que se cumpla una determinada condición.

Para ello utilizaremos la instrucción WHILE que tiene la siguiente forma:

```

WHILE <condición>
BEGIN
    <conjunto de instrucciones>
END

```

Podemos cancelar prematuramente el bucle con la instrucción BREAK y podemos saltar a la siguiente iteración con la instrucción CONTINUE.

Laboratorio 3

En este laboratorio vamos a imprimir los nombres de las vías con identificadores 1 a 10 con un bucle en vez de llamar 10 veces a la función PRINT como hicimos en el laboratorio 23.

1. Ejecutamos Microsoft SQL Server Management Studio, nos conectamos con nuestro servidor de bases de datos.
2. Pulsamos el botón *New Query* de la barra de herramientas *Standard*. Con eso conseguimos que Microsoft SQL Server Management Studio cree un archivo .sql vacío.
3. Seleccionamos CompatibilidadCATDBS en el cuadro de opciones *Available Databases* de la barra de herramientas *SQL Editor*.
4. Tecleamos en nuestro archivo .sql lo siguiente:

```
-- Declaramos la variable que utilizaremos como índice de la
-- vía a imprimir
DECLARE @índice int

-- Inicializamos el índice con el identificador de la primera
-- vía a imprimir
SET @índice = 1

-- Iteramos 10 veces
WHILE @índice <= 10
BEGIN
    -- Imprimimos el nombre de la vía cuyo índice coincide
    -- con el valor almacenado en la variable @índice
    PRINT dbo.NombreVia(@índice)

    -- Incrementamos el índice una unidad
    SET @índice = @índice + 1
END
```

5. Pulsamos el botón *Execute* de la barra de herramientas *SQL Editor* para comprobar que Microsoft SQL Server Management Studio nos muestra los nombres de las vías 1 a 10.

Procedimientos almacenados TSQL

Un procedimiento almacenado es un conjunto de instrucciones agrupadas con un nombre que puede recibir una serie de parámetros.

La forma de declarar/modificar un procedimiento almacenado es la siguiente:

```
CREATE/ALTER PROCEDURE <nombre del procedimiento almacenado>
    @parámetro 1 <tipo 1>,
    @parámetro 2 <tipo 2>,
```

```

...
@parámetro n <tipo n>
AS
BEGIN
    <instrucciones del procedimiento almacenado>
END

```

Para ejecutar un procedimiento almacenado, utilizamos la instrucción EXEC:

EXEC <nombre del procedimiento almacenado> parámetro_1, parámetro_2, ..., parámetro_n

Laboratorio 4

En este laboratorio vamos a crear un procedimiento almacenado que recibe como parámetros un número de identificador de vía inferior y un número de identificador de vía mayor. El procedimiento almacenado imprimirá el nombre de todas las vías cuyo identificador de vía esté dentro de ese rango.

1. Ejecutamos Microsoft SQL Server Management Studio, nos conectamos con nuestro servidor de bases de datos.
2. En la ventana *Object Explorer* abrimos la rama *CompatibilidadCATDBS/Programmability/Stored Procedures*. Si localizamos el procedimiento almacenado *dbo.ImprimeNombresVias*, lo eliminamos con el botón derecho del ratón, seleccionando *Delete* en el menú contextual y aceptando el cuadro de diálogo que nos pide confirmación para eliminar el procedimiento almacenado.
3. Pulsamos el botón *New Query* de la barra de herramientas *Standard*. Con eso conseguimos que Microsoft SQL Server Management Studio cree un archivo .sql vacío.
4. Seleccionamos CompatibilidadCATDBS en el cuadro de opciones *Available Databases* de la barra de herramientas *SQL Editor*.
5. Tecleamos en nuestro archivo .sql lo siguiente:

```

CREATE PROCEDURE ImprimeNombresVias
    @idMenor int,
    @idMayor int
AS
BEGIN
    DECLARE @contador int

    SET @contador = @idMenor

    WHILE @contador <= @idMayor
    BEGIN
        PRINT dbo.NombreVia(@contador)

        SET @contador = @contador + 1
    END
END

```

6. Pulsamos el botón *Execute* de la barra de herramientas *SQL Editor* para comprobar que Microsoft SQL Server Management Studio nos indica que el procedimiento se ha creado satisfactoriamente.
7. Pulsamos el botón *New Query* de la barra de herramientas *Standard* para crear una nueva ventana y tecleamos lo siguiente:

```
EXEC dbo.ImprimeNombresVias 1,5
```

8. De esta manera ordenaremos al procedimiento almacenado que imprima los nombres de las primeras 5 vías.
9. Pulsamos el botón *Execute* de la barra de herramientas *SQL Editor* y comprobamos que el procedimiento almacenado imprime los nombres de las vías correctamente.

Instrucciones de control de flujo TSQL

TSQL dispone de instrucciones que nos permiten controlar el conjunto de instrucciones a ejecutar en función de si se cumplen un con conjunto de instrucciones.

Para ello utilizaremos la instrucción IF:

```
IF <condición 1>
BEGIN
    <conjunto de instrucciones a ejecutar si se cumple la condición 1>
END
ELSE IF <condición 2>
BEGIN
    <conjunto de instrucciones a ejecutar si se cumple la condición 1>
END
...
ELSE IF <condición n>
BEGIN
    <conjunto de instrucciones a ejecutar si se cumple la condición n>
END
ELSE
BEGIN
    <conjunto de instrucciones a ejecutar sino se ha cumplido ninguna de las condiciones
    anteriores>
END
```

Laboratorio 5

En este laboratorio vamos crear un procedimiento almacenado que imprimirá un mensaje indicando si la velocidad pasada por parámetros está dentro de un rango definido. Vamos a suponer que el valor velocidad debe estar comprendido entre 20 y 120 ambos inclusive.

1. Ejecutamos Microsoft SQL Server Management Studio, nos conectamos con nuestro servidor de bases de datos.
2. En la ventana *Object Explorer* abrimos la rama *CompatibilidadCATDBS/Programmability/Stored Procedures*. Si localizamos el procedimiento almacenado *dbo.ImprimeTestVelocidad*, lo eliminamos con el botón derecho del ratón, seleccionando *Delete* en el menú contextual y aceptando el cuadro de diálogo que nos pide confirmación para eliminar el procedimiento almacenado.
3. Pulsamos el botón *New Query* de la barra de herramientas *Standard*. Con eso conseguimos que Microsoft SQL Server Management Studio cree un archivo .sql vacío.
4. Seleccionamos CompatibilidadCATDBS en el cuadro de opciones *Available Databases* de la barra de herramientas *SQL Editor*.
5. Tecleamos en nuestro archivo .sql lo siguiente:

```
CREATE PROCEDURE ImprimeTestVelocidad
    @velocidad int
AS
BEGIN
    IF @velocidad < 20
    BEGIN
        PRINT 'La velocidad no puede ser menor que 20 km/h'
    END
    ELSE IF @velocidad > 120
    BEGIN
        PRINT 'La velocidad no puede ser mayor que 120 km/h'
    END
    ELSE
    BEGIN
        PRINT 'La velocidad es correcta'
    END
END
```

6. Pulsamos el botón *Execute* de la barra de herramientas *SQL Editor* para comprobar que Microsoft SQL Server Management Studio nos indica que el procedimiento se ha creado satisfactoriamente.
7. Pulsamos el botón *New Query* de la barra de herramientas *Standard* para crear una nueva ventana y tecleamos lo siguiente:

```
EXEC dbo.ImprimeTestVelocidad 50
EXEC dbo.ImprimeTestVelocidad 250
EXEC dbo.ImprimeTestVelocidad 12
```

8. Pulsamos el botón *Execute* de la barra de herramientas *SQL Editor* para comprobar que Microsoft SQL Server Management Studio nos muestra en la ventana de resultados los mensajes generados por el procedimiento almacenado indicándonos si la velocidad es o no correcta.

Comunicación de errores TSQL

En el laboratorio anterior, hemos impreso en la ventana de salida (con la instrucción PRINT) mensajes para indicar que la velocidad pasada por parámetros al procedimiento almacenado no estaba comprendida entre los valores 20 y 120.

Este tipo de comunicación no sirve para realizar control de errores, ya que el segundo programa, el que llamaba tres veces al procedimiento almacenado, no se ha enterado de que dos de las llamadas han generado un error.

Sería ideal que el programa que llama al procedimiento almacenado descubriera de alguna manera que si pasamos como parámetro 12 se va a generar un error y actuar en consecuencia.

En TSQL los errores se comunican mediante la función RAISERROR cuyo formato es el siguiente:

`RAISERROR ('<mensaje a mostrar al usuario>',<severidad>,<estado>)`

Severidad es un número de 0 a 25.

Si se lanza un error cuyo valor severidad esté comprendido entre 0 y 10 (inclusive) no se considerará un error grave y por lo tanto no interrumpirá la ejecución de un programa, lo que significa que se seguirán ejecutando las instrucciones que vengan a continuación de la llamada a RAISERROR.

Si se lanza un error cuyo valor severidad esté comprendido entre 11 y 18 (ambos inclusive) se considerará un error suficientemente grave como para interrumpir la ejecución de código en el grupo de código actual y ejecutar el conjunto de instrucciones CATCH (que veremos más adelante) que es un conjunto de instrucciones a ejecutar si se detecta un error.

Normalmente no podremos lanzar errores con valor severidad entre 20 y 25, ya que para poder hacerlo tendremos que ser miembros del grupo de usuarios SQL Sysadmin. Si lanzamos errores con este tipo de severidad, se considera tan grave el hecho de lanzarlos que se finalizará la conexión con el servidor de bases de datos.

El valor *estado* es un número que puede servir para identificar qué parte del programa ha generado el error, y es un número entre 1 y 127, de modo que si en un programa tenemos varias instrucciones RAISERROR, podríamos utilizar el estado 1 en el primer RAISERROR y 2 en el segundo y así sucesivamente para identificar rápidamente el punto en el que se ha lanzado el error.

Laboratorio 6

En este laboratorio vamos a crear un procedimiento almacenado denominado *CompruebaVelocidad* que va a generar mensajes de error si la velocidad pasada por parámetros no está dentro de los valores 20 y 120. Luego ejecutaremos el procedimiento

almacenado para que compruebe tres velocidades y ver cómo el procedimiento almacenado genera errores en vez de mensajes en la consola que ningún programa puede interceptar.

1. Ejecutamos Microsoft SQL Server Management Studio, nos conectamos con nuestro servidor de bases de datos.
2. En la ventana *Object Explorer* abrimos la rama *CompatibilidadCATDBS/Programmability/Stored Procedures*. Si localizamos el procedimiento almacenado *dbo.CompruebaVelocidad*, lo eliminamos con el botón derecho del ratón, seleccionando *Delete* en el menú contextual y aceptando el cuadro de diálogo que nos pide confirmación para eliminar el procedimiento almacenado.
3. Pulsamos el botón *New Query* de la barra de herramientas *Standard*. Con eso conseguimos que Microsoft SQL Server Management Studio cree un archivo .sql vacío.
4. Seleccionamos CompatibilidadCATDBS en el cuadro de opciones *Available Databases* de la barra de herramientas *SQL Editor*.
5. Tecleamos en nuestro archivo .sql lo siguiente:

```
CREATE PROCEDURE CompruebaVelocidad
    @velocidad int
AS
BEGIN
    IF @velocidad < 20
        BEGIN
            RAISERROR ('Velocidad menor que 20 km/h', 16, 1)
        END
    ELSE IF @velocidad > 120
        BEGIN
            RAISERROR ('Velocidad mayor que 120 km/h', 16, 2)
        END
END
```

6. Pulsamos el botón *Execute* de la barra de herramientas *SQL Editor* para comprobar que Microsoft SQL Server Management Studio nos indica que el procedimiento se ha creado satisfactoriamente.
7. Pulsamos el botón *New Query* de la barra de herramientas *Standard* para crear una nueva ventana y tecleamos lo siguiente:

```
EXEC dbo.CompruebaVelocidad 50
EXEC dbo.CompruebaVelocidad 250
EXEC dbo.CompruebaVelocidad 12
```

8. Pulsamos el botón *Execute* de la barra de herramientas *SQL Editor* para comprobar que Microsoft SQL Server Management Studio nos muestra en la ventana de resultados los errores generados por el procedimiento almacenado si la velocidad pasada por parámetros no es correcta.

Captura de errores en TSQL

Ya sabemos cómo se deben comunicar los errores en TSQL. Las llamadas a RAISERROR las podemos hacer nosotros en nuestros programas como hemos hecho en el laboratorio anterior

o las puede generar el propio SQL Server. Si intentamos por ejemplo realizar una consulta sobre una tabla en la que no tenemos permisos de lectura, SQL Server generará un error con RAISERROR, si intentamos realizar una operación matemática incorrecta, como dividir un número por cero, SQL Server generará un error con RAISERROR... Todos los componentes de SQL Server comunican los errores mediante TSQL.

En el punto anterior he comentado que si el valor severidad está comprendido entre 11 y 18, se puede interrumpir la ejecución del programa forzando a que se ejecute un conjunto de instrucciones denominado CATCH.

He dicho *se puede* porque si no indicamos a TSQL que tenemos un grupo de instrucciones CATCH, el programa no se interrumpirá y se ejecutarán las instrucciones que continúen después de la instrucción que ha generado el error.

Veámoslo con un ejemplo, si ejecutas el siguiente código:

```
PRINT 'A'  
RAISERROR ( 'Esto es un error simple' , 18 , 1 )  
PRINT 'B'
```

Verás que en la consola de salida, se imprime una 'A' seguida de una 'B' y al final el mensaje de error.

Si nos interesa paralizar la ejecución del programa si se detecta un error y realizar una serie de operaciones únicamente en caso de que se detecte el error, debemos utilizar las instrucciones TRY/CATCH que tienen la siguiente forma:

```
BEGIN TRY  
    <instrucciones que pueden generar error>  
END TRY  
BEGIN CATCH  
    <instrucciones a ejecutar si se genera un error>  
END CATCH
```

Laboratorio 7

En este laboratorio vamos a capturar los errores generados por el procedimiento almacenado *dbo.CompruebaVelocidad* si la velocidad pasada por parámetros no está comprendida entre 20 y 120 y muestra un mensaje al usuario indicando que se ha localizado un error.

1. Ejecutamos Microsoft SQL Server Management Studio, nos conectamos con nuestro servidor de bases de datos.
2. Pulsamos el botón *New Query* de la barra de herramientas *Standard*. Con eso conseguimos que Microsoft SQL Server Management Studio cree un archivo .sql vacío.
3. Seleccionamos CompatibilidadCATDBS en el cuadro de opciones *Available Databases* de la barra de herramientas *SQL Editor*.

4. Tecleamos en nuestro archivo .sql lo siguiente:

```

BEGIN TRY
    EXEC dbo.CompruebaVelocidad 250
    PRINT 'La velocidad era correcta'
END TRY
BEGIN CATCH
    PRINT 'Se ha producido un error'
END CATCH

```

5. Pulsamos el botón *Execute* de la barra de herramientas *SQL Editor* para que el programa muestra el mensaje *Se ha producido un error* y nunca llega a mostrar el mensaje *La velocidad era correcta*.

Funciones que únicamente son válidas en el contexto de un catch

Si recuerdas los mensajes de error que genera el procedimiento almacenado *dbo.CompruebaVelocidad*, éstos eran o ‘Velocidad menor que 20 km/h’ o ‘Velocidad mayor que 120 km/h’, sin embargo, en el laboratorio anterior, estamos mostrando en la consola de resultados la cadena ‘Se ha producido un error’, es decir, estamos obviando el mensaje de error que ha generado el procedimiento almacenado.

Existen una serie de funciones a las que únicamente podremos llamar dentro de un bloque de código CATCH que nos van a permitir obtener el mensaje asociado con la instrucción RAISERROR que ha generado el error, así como su severidad y su estado.

Estas funciones son las siguientes:

Nombre de la función	Descripción
ERROR_MESSAGE()	Devuelve la <i>cadena de error</i> del RAISERROR que ha forzado la ejecución del código CATCH.
ERROR_SEVERITY()	Devuelve el valor <i>severidad</i> del RAISERROR que ha forzado la ejecución del código CATCH.
ERROR_STATE()	Devuelve el valor <i>estado</i> del RAISERROR que ha forzado la ejecución del código CATCH.
ERROR_LINE()	Devuelve el número de línea del programa TSQL que ha generado el error.
ERROR_PROCEDURE()	Devuelve el nombre del procedimiento almacenado que ha provocado el error.
ERROR_NUMBER()	Devuelve el número de error.

En este laboratorio vamos a capturar los errores generados por el procedimiento almacenado *dbo.CompruebaVelocidad* si la velocidad pasada por parámetros no está comprendida entre 20 y 120 y mostrará la descripción del error tal y como la generó el procedimiento almacenado.

1. Ejecutamos Microsoft SQL Server Management Studio, nos conectamos con nuestro servidor de bases de datos.
2. Pulsamos el botón *New Query* de la barra de herramientas *Standard*. Con eso conseguimos que Microsoft SQL Server Management Studio cree un archivo .sql vacío.
3. Seleccionamos CompatibilidadCATDBS en el cuadro de opciones *Available Databases* de la barra de herramientas *SQL Editor*.
4. Tecleamos en nuestro archivo .sql lo siguiente:

```
BEGIN TRY
    EXEC dbo.CompruebaVelocidad 250
    PRINT 'La velocidad era correcta'
END TRY
BEGIN CATCH
    PRINT ERROR_MESSAGE()
END CATCH
```

5. Pulsamos el botón *Execute* de la barra de herramientas *SQL Editor* para que el programa muestra el mensaje *Se ha producido un error* y nunca llega a mostrar el mensaje *La velocidad era correcta* y ahora sí que se muestra el motivo del error.

Desencadenadores en TSQL

Los desencadenadores son procedimientos que se ejecutan cuando sucede un evento (inserción, eliminación o modificación de un registro) en una tabla de la base de datos.

Si añadidos un nuevo registro en una tabla, se ejecutan los desencadenadores asociados con el evento *insert* en dicha tabla. Microsoft SQL Server creará una tabla virtual denominada *inserted* accesible en el contexto del código del desencadenador, con la misma estructura que la tabla sobre la que se está ejecutando el desencadenador, con la información que se está añadiendo.

Si eliminamos un registro de una tabla, se ejecutan los desencadenadores asociados con el evento *delete* en dicha tabla. Microsoft SQL Server creará una tabla virtual denominada *deleted* accesible en el contexto del código del desencadenador, con la misma estructura que la tabla sobre la que se está ejecutando el desencadenador, con la información que se está eliminando.

Si modificamos un registro, se ejecutan los desencadenadores asociados con el evento *update* en dicha tabla. Microsoft SQL Server creará dos tablas virtuales denominadas *inserted* y *deleted* accesibles en el contexto del código del desencadenador, con la misma estructura que la tabla sobre la que se está ejecutando el desencadenador, con la información que se está insertando en la primera y eliminando en la segunda. Una actualización consiste en eliminar un registro y crear otro nuevo.

Los desencadenadores son muy útiles, ya que nos permiten realizar comprobaciones en los registros afectados para cancelar la operación por ejemplo, o crear un alta en una tabla de LOG para informar del movimiento, o enviar un correo electrónico o utilizar un servicio de SMS para notificar que ha sucedido un evento...

Declaramos un desencadenador mediante la instrucción:

```
CREATE/ALTER TRIGGER <nombre_del_desencadenador>
ON <nombre_de_tabla>
FOR [INSERT,UPDATE,DELETE]
AS
< código_del_desencadenador >
```

Tienes que tener en cuenta que los desencadenadores se ejecutan después de haber realizado la modificación en la tabla, es decir, que si el desencadenador se lanza con el evento INSERT, cuando se lanza, la tabla ya tiene el registro insertado.

Si deseamos que se elimine porque ese registro no cumple con las reglas de negocio de la base de datos podremos hacerlo cancelando la transacción mediante ROLLBACK TRANS o ROLLBACK TRANSACTION.

Laboratorio 9

En este laboratorio añadiremos un campo *Ancho* (de tipo *float*), un campo *Carriles* (de tipo *tinyint*) y un campo *Velocidad* (de tipo *tinyint*) y a la tabla *Viales*. El campo *carriles* deberá ser al menos 1.

Continuaremos creando un procedimiento almacenado denominado *CompruebaAnchoVia* que comprobará que el ancho pasado por parámetros será al menos *número_de_carriles * 3.5* y que en todo caso nunca superará *número_de_carriles * 5.0*.

Crearemos un desencadenador que se ejecutará cada vez que se añada un nuevo registro en la tabla *Viales* que comprobará que el valor indicado en el campo *Carriles* es 1 o mayor, comprobará que el valor indicado por el usuario en el campo *Ancho* está dentro de los límites (delegando esa función en el procedimiento almacenado *CompruebaAnchoVia*) y por último comprobará que la velocidad máxima en la vía está dentro de los límites (delegando esa función en el procedimiento almacenado *CompruebaVelocidad*).

Como el cuerpo del desencadenador puede localizar errores y nos interesa que si se localiza un error no continúen ejecutándose más instrucciones, haremos que todo el cuerpo del desencadenador esté dentro de un bloque TRY/CATCH, y como nos interesa que el mensaje que le pase al módulo que ha intentado dar el alta (que por ahora es Microsoft SQL Server Management Studio, pero dentro de muy poco será Digi3D 2011), reenviaremos ese mensaje de error en el cuerpo del CATCH.

1. Ejecutamos Microsoft SQL Server Management Studio, nos conectamos con nuestro servidor de bases de datos.
2. Desplegamos Databases/CompatibilidadCATDBS/Tables en la ventana *Object Explorer* y con el botón derecho del ratón sobre la tabla *Viales* seleccionamos *Design*.
3. Añadimos los siguientes campos:

Nombre	Tipo	Admite nulos
Ancho	float	Si
Carriles	tinyint	Si
Velocidad	tinyint	Si

4. Cerramos la ventana de diseño de la tabla y aceptamos el cuadro de diálogo que nos pide confirmación para modificar la tabla.
5. En la ventana *Object Explorer* abrimos la rama *CompatibilidadCATDBS/Programmability/Stored Procedures*. Si localizamos el procedimiento almacenado *dbo.CompruebaAnchoVial*, lo eliminamos con el botón derecho del ratón, seleccionando *Delete* en el menú contextual y aceptando el cuadro de diálogo que nos pide confirmación para eliminar el procedimiento almacenado.
6. Pulsamos el botón *New Query* de la barra de herramientas *Standard*.
7. Seleccionamos CompatibilidadCATDBS en el cuadro de opciones *Available Databases* de la barra de herramientas *SQL Editor*.
8. Tecleamos en nuestro archivo .sql lo siguiente:

```

CREATE PROCEDURE CompruebaAnchoVia
    @anchoVia float,
    @carriles smallint
AS
BEGIN
    IF (@anchoVia < 3.5 * @carriles)
        BEGIN
            RAISERROR ('La vía no es suficientemente ancha para
el número de carriles', 16, 1)
        END
    ELSE IF (@anchoVia > 5 * @carriles)
        BEGIN
            RAISERROR ('La vía es demasiado ancha', 16, 2)
        END
END

```

9. Pulsamos el botón *Execute* de la barra de herramientas *SQL Editor*.
10. Pulsamos el botón *New Query* de la barra de herramientas *Standard* y tecleamos y ejecutamos el siguiente código:

```

CREATE TRIGGER TR_Viales
ON Viales

```

```

AFTER INSERT
AS
BEGIN TRY
    /* Declaramos las variables donde almacenaremos los valores
    velocidad y ancho que se están intentando insertar en la
    base de datos */
    DECLARE @ancho float,
            @carriles tinyint,
            @velocidad tinyint

    /* Asignamos los valores que se están intentando almacenar
    en las variables declaradas */
    SELECT @ancho = Ancho,
           @carriles = Carriles,
           @velocidad = Velocidad
    FROM INSERTED

    /* Aplicamos las reglas de negocio */
    IF @carriles < 1
    BEGIN
        RAISERROR ('El número de carriles debe ser al menos
1', 16, 1)
    END
    EXEC dbo.CompruebaAnchoVia @ancho, @carriles
    EXEC dbo.CompruebaVelocidad @velocidad
END TRY
BEGIN CATCH
    -- Cancelamos la transacción para evitar que se almacene
    este registro

    ROLLBACK TRANSACTION
    -- Reenviamos el mensaje de error al llamador (Digi3D 2011)
    DECLARE @ErrorMessage nvarchar(256);
    DECLARE @ErrorSeverity int;
    DECLARE @ErrorState int;

    SELECT
        @ErrorMessage = ERROR_MESSAGE(),
        @ErrorSeverity = ERROR_SEVERITY(),
        @ErrorState = ERROR_STATE();

    RAISERROR (@ErrorMessage, @ErrorSeverity, @ErrorState)
END CATCH

```

11. Pulsamos el botón *New Query* de la barra de herramientas *Standard* y tecleamos y ejecutamos esto:

```

-- Insertamos una vía con valores válidos
INSERT INTO Viales (Nombre, Ancho, Carriles, Velocidad)
VALUES ('M-30', 19, 5, 90)

-- Insertamos u'na vía con valores inválidos
INSERT INTO Viales (Nombre, Ancho, Carriles, Velocidad)
VALUES ('M-50', 19, 5, 200)

```

12. Podemos comprobar que en el segundo caso no se añade el registro porque se nos envía un error indicando que la velocidad no puede ser mayor que 120 km/h. Podemos comprobarlo además ejecutando una consulta SQL que nos muestre todos los registros de la tabla Viales cuyo nombre comience por M-:

```
SELECT * FROM Viales WHERE Nombre LIKE 'M-%'
```