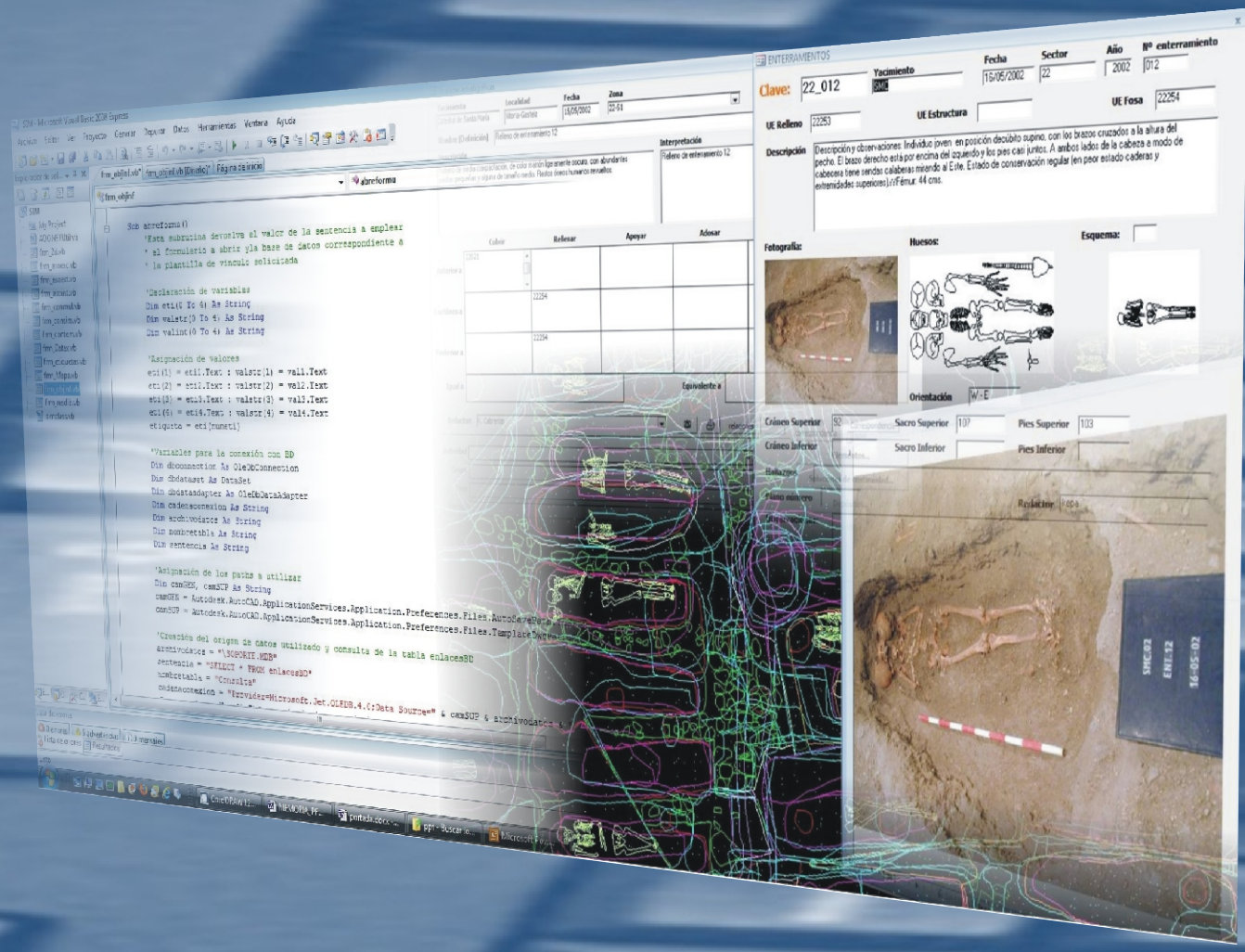
Herramientas de programación de VB.NET aplicadas al Sistema de Información Monumental de la Catedral de Santa María de Vitoria-Gasteiz



Proyecto Fin de Máster
Máster en Geotecnologías Cartográficas en Ingeniería y Arquitectura
Escuela Politécnica Superior de Ávila. Universidad de Salamanca
Autor: Iñaki Koroso
Tutor: José Antonio Martín Jiménez

Septiembre 2011

Herramientas de programación de VB.NET aplicadas al SIM de la Catedral de Santa María

1.- Objetivos del Proyecto Fin de Máster

Entre los objetivos de la titulación que se describen en la Guía Docente del Máster se hace especial hincapié en el compromiso de "*formar a sus alumnos en sus propios recursos de manera que alcancen la dirección de su propio aprendizaje para enfrentar cualquier reto profesional futuro".* Es primordial que el alumno tenga una base teórica suficiente y una perspectiva adecuada para poder utilizar las diferentes herramientas necesarias para la realización de proyectos relacionados con las nuevas tecnologías en Ingeniería Geomática.

El Proyecto Fin de Máster debe ser el reflejo de las nuevas capacidades y competencias adquiridas en la realización del Máster y es deseable que esté lo más próximo posible a la realidad profesional del sector.

Teniendo estas consideraciones en cuenta he titulado mi Proyecto Fin de Máster titulado ***Herramientas de programación de VB.NET aplicadas al Sistema de Información Monumental (SIM) de la Catedral de Santa María.*** Este proyecto ha utilizado la asignatura "Herramientas Informáticas para el Geoprocesado" para profundizar en la aplicación del lenguaje de programación VB.NET sobre un programa de diseño asistido por ordenador como es la serie de Autodesk. Su aplicación profesional es inmediata ya que entre las funciones que actualmente desempeño en mi trabajo profesional está la de responsable del Sistema de Información Monumental de la Catedral de Santa María y una herramienta de este tipo será utilizada desde el mismo momento de su realización.

El objetivo principal es proponer una metodología para el desarrollo de herramientas informáticas en lenguaje VB.NET aplicables a entornos Autocad, a partir de la programación de aplicaciones para el Sistema de Información Monumental (SIM) de la Catedral de Santa María de Vitoria-Gasteiz. Además, y para utilizar al máximo el potencial del lenguaje VB.NET, se han desarrollado programas y aplicaciones para tareas generales del entorno Autocad y Autodesk Map (módulo de Autocad para la gestión de Sistemas de Información Geográfica), así como para la integración de estos programas con aplicaciones gestoras de base de datos.

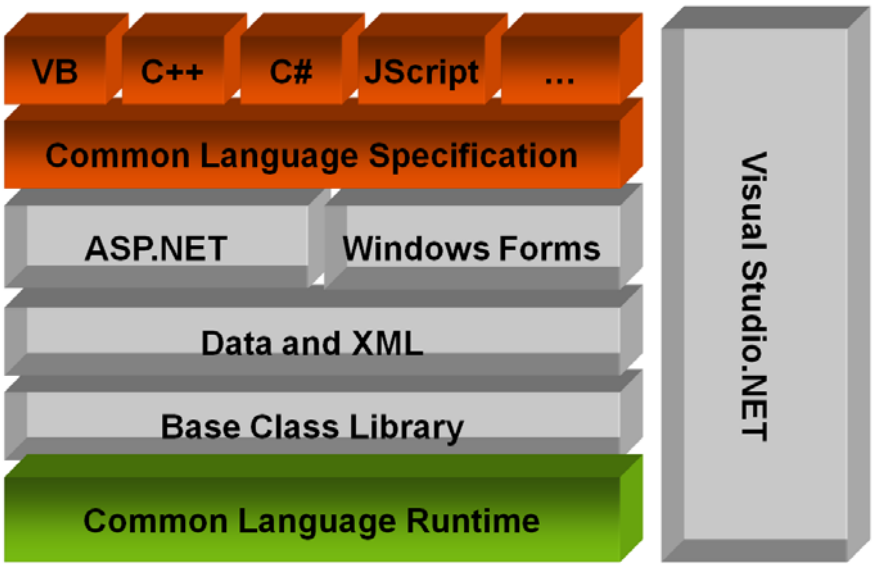
Los objetivos específicos son dos. Por un lado el proyecto pretende animar y motivar a otros usuarios de entornos geomáticos a explorar las posibilidades de la programación aplicadas a trabajos concretos. Por otro lado se quiere acercar las herramientas de análisis y gestión de la información geográfica a los profesionales de diferentes disciplinas que participan en la restauración de la Catedral de Santa María de Vitoria-Gasteiz mediante la programación de recursos específicos para el SIM.

Este Proyecto Fin de Máster se encuadra dentro de un proyecto formativo en el que el intercambio de experiencias resulta muy importante y por ello se aporta el código utilizado en la programación de todas las aplicaciones. El autor es consciente de la utilidad y el valor que las comunidades de desarrolladores de programas tienen en el trabajo de programación y por ello permite la utilización de cualquier parte del código a terceros, siempre y cuando el código generado esté accesible a la comunidad de desarrolladores. Evidentemente, este proyecto espera ser de utilidad para todos aquellos que quieran utilizar el lenguaje de programación VB.NET para el desarrollo de aplicaciones geomáticas.

2.- El lenguaje VB.NET

Visual Basic .NET (VB.NET) es un lenguaje de programación orientado a objetos que se puede considerar una evolución de Visual Basic implementada sobre .NET Framework . Su introducción resultó muy controvertida, ya que debido a cambios significativos en el lenguaje VB.NET no es compatible hacia atrás con Visual Basic debiendo reprogramar la mayor parte del código y de la estructura del programa. Por otro lado el manejo concreto de algunas instrucciones es similar a versiones anteriores de Visual Basic, facilitando así el desarrollo de aplicaciones más avanzadas con herramientas modernas.

La solución .NET Framework es la **plataforma de desarrollo** de código administrado de Microsoft. Integra una gran cantidad de lenguajes que conforman una serie de librerías que pueden ser utilizadas por otros lenguajes de la plataforma. VB.NET es una versión más robusta y potente que otras versiones de Visual Basic y se acerca a lenguajes como C#. Este cambio, valorado positivamente por la mayoría de usuarios repercutió negativamente en la histórica sencillez de programación de Visual Basic.



Elementos que componen la herramienta de desarrollo Microsoft Visual Studio .NET

Los principales beneficios de utilizar la tecnología .NET son la utilización de una plataforma de desarrollo de programación orientada a objetos, una gestión automática de la memoria y la integración en una herramienta de diferentes lenguajes de programación.

A pesar de la gran cantidad de usuarios que utilizan Autocad sólo una minoría programa sus propias aplicaciones. Esto hace que haya una carencia de libros y publicaciones sobre VB.NET y Autocad. Además las aplicaciones más interesantes para el Sistema de Información Monumental estaban basadas en comandos específicos de Autodesk Map por lo que buscar bibliografía específica ha sido tarea casi imposible.

No obstante, la red es una comunidad de usuarios formidable y hay grupos de discusión o blogs en los que se pueden consultar dudas.

3.- El Sistema de Información Monumental de la Catedral de Santa María de Vitoria-Gasteiz.

El desarrollo del Sistema de Información Geográfica que en la catedral hemos denominado Sistema de Información Monumental (SIM) es un proceso complejo y en mejora continua que se coordina desde el Área Técnica de la Fundación Catedral Santa María. La singularidad del proyecto y su continuidad en el tiempo han posibilitado las rutinas de trabajo necesarias para el mantenimiento del sistema.

En primer lugar el elevado número de entidades gráficas que componen el modelo tridimensional ha obligado a dividir el monumento en diferentes partes según criterios funcionales. Otro tanto se ha realizado con las diferentes campañas de excavaciones llevadas a cabo. Esto nos permite preseleccionar aquellos elementos sobre los que queremos realizar consultas optimizando los recursos del sistema y agilizando la gestión de la información. En nuestro caso hemos dividido el interior de la catedral en 45 partes diferenciadas, el exterior lo forman 21 archivos y en el subsuelo hemos optado por la división en función de las campañas realizadas obteniendo un total de 19 archivos.

Convienen remarcar que el modelo geométrico de la Catedral está compuesto en su inmensa mayoría por polilíneas 3d. Esto quiere decir que lo que define el modelo no son superficies sino contornos, ya que una polilínea 3d no es coplanar (a diferencia de una polilínea 2d). Esta característica prioriza la precisión geométrica sobre la sencillez en la gestión y ha marcado de forma importante los métodos de trabajo en el SIM.

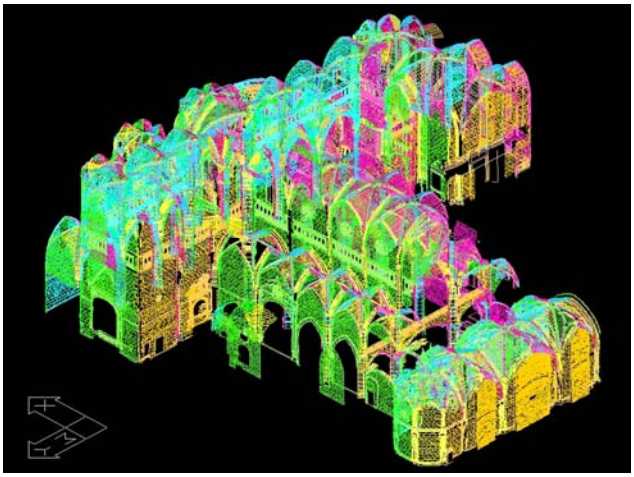


Imagen del modelo tridimensional del interior

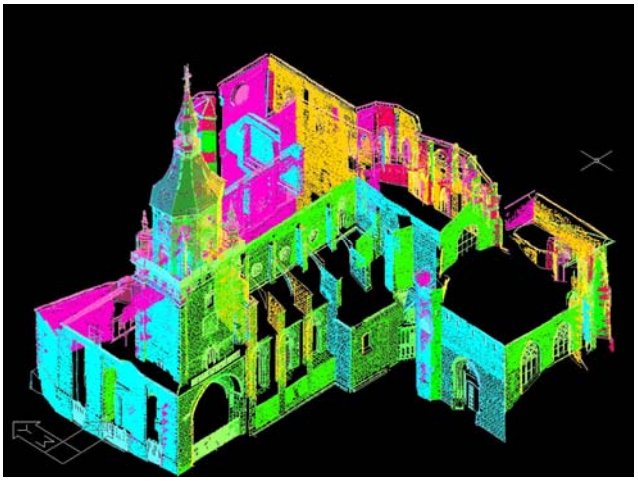
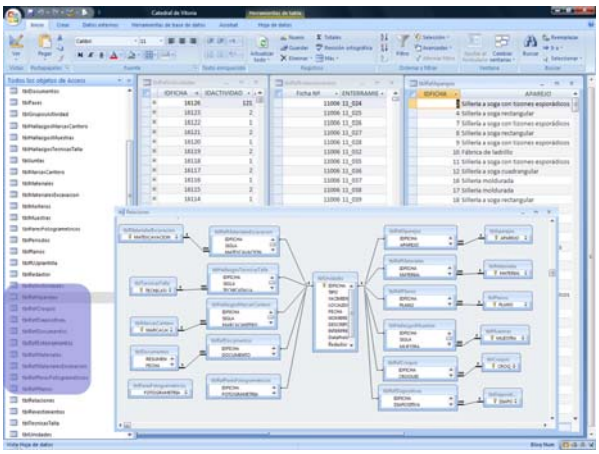
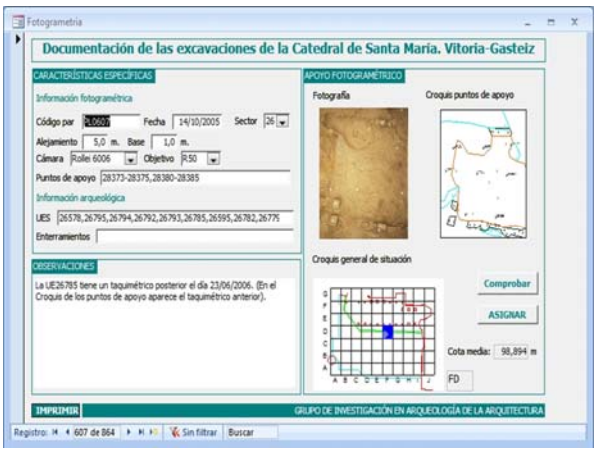


Imagen del modelo tridimensional del exterior

Toda esta información geométrica (más de 600.000 entidades polilíneas 3d) ha sido analizada en sus características temáticas por un amplio equipo interdisciplinar y además toda la información recopilada y generada tiene que estar disponible para su consulta directa desde el sistema. Esto requiere un planteamiento previo a la realización del estudio que posibilite la clasificación de todas las informaciones relevante en bases de datos relacionadas entre sí. La información articulada en bases de datos permite su sistematización y además de la función evidente de mejorar su análisis y gestión tiene la gran ventaja de que se puede adaptar a cualquier sistema gestor de base de datos, evitando que la información caduque cuando la empresa que comercializa el software deja de ofrecer actualizaciones.



Esquema de relaciones entre diferentes tablas de una base de datos del SIM



Formulario correspondiente a la documentación de los pares fotogramétricos utilizados para documentar las excavaciones

Las dos áreas temáticas que caracterizan la totalidad del monumento se refieren a los estudios arquitectónicos, con la base de datos CONSTRUC, cuyo elemento mínimo es la Unidad Constructiva y la correspondiente a las excavaciones arqueológicas que se llama HISTORIA y cuyo campo clave es la Unidad Estratigráfica.

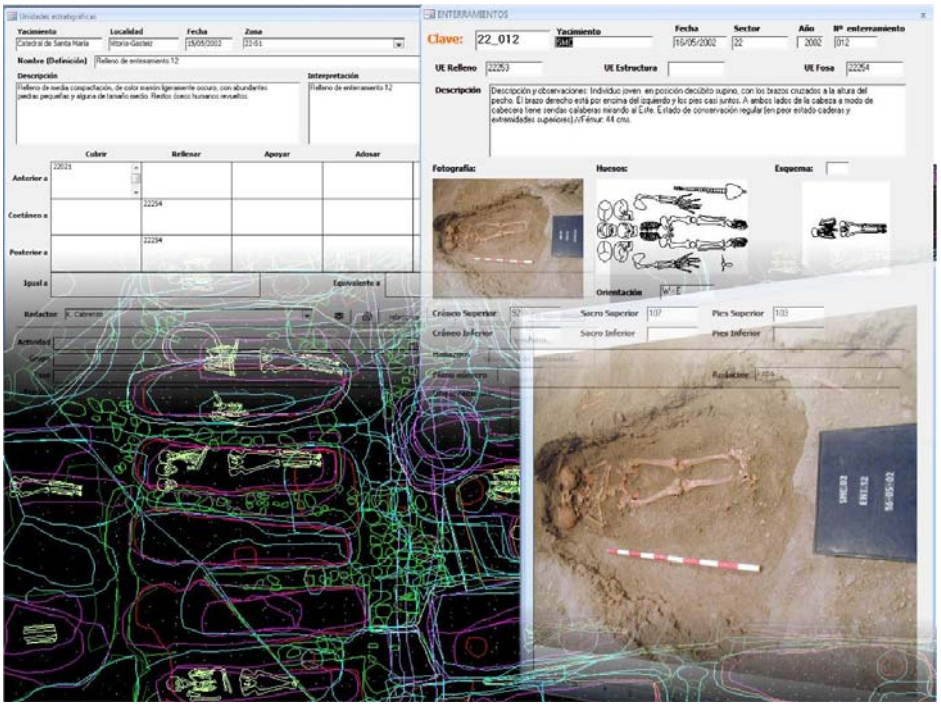
Dentro de CONSTRUC se agrupan las bases de datos referidas a los elementos construidos y todas las características sobre su estado actual, sus patologías, su composición y su comportamiento estructural. Además se puede conocer entre otros aspectos su composición litológica, los ensayos geotécnicos realizados, los tipos de fábrica que se presentan, los diferentes morteros utilizados, los estudios de colonización vegetal así como la información referida a los sondeos realizados en pilares y muros para la determinación de la composición interior de estos. En cuanto a cifras, se puede traducir en unas 1500 unidades constructivas, más de diez áreas temáticas y más de 3000 archivos de imagen.

En la base de datos HISTORIA se puede analizar toda la información referida al proceso de excavación arqueológica, a la lectura de alzados realizada así como a las conclusiones de todos estos trabajos. Dentro de esta base de datos que se analizará en el siguiente capítulo se puede acceder a información detallada como son las fichas de campo realizadas en excavaciones, las características métricas de los pares fotogramétricos utilizados para la documentación métrica, los enterramientos que han aparecido o los fragmentos de cerámica que se han recogido. Entre los alzados y el subsuelo de la catedral se han documentado más de 11000 Unidades Estratigráficas que quedan recogidas, acompañados de más de 2000 enterramientos, 3000 pares fotogramétricos y 20000 imágenes.

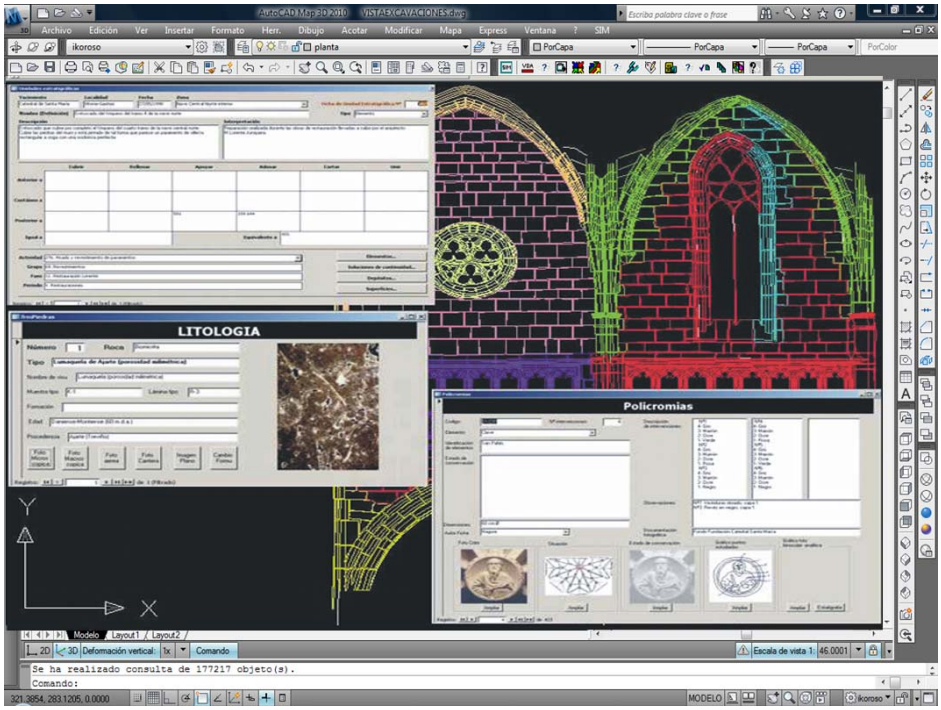
Además de las bases de datos antes descritas, el sistema se completa con la base de datos de POLICROMÍAS que hace referencia al completo estudio de correspondencia de policromías realizado en el pórtico de la catedral y que puede ser consultado en (http://www.catedralvitoria.com/mediateca_extras.php?opc=30_204).

Para la gestión de esta enorme cantidad de datos geográficos, alfanuméricos y gráficos, la Fundación Catedral Santa María utiliza el programa Autodesk Map para la gestión del modelo 3d y sus enlaces con bases de datos (miles de registros y decenas de miles de imágenes).

Las herramientas disponibles en Autodesk Map son inmensas y abarcan un espectro de utilidades mucho mayor que el utilizado en la restauración de la catedral. Esto complica la usabilidad para personas que no conocen a fondo el programa y este proyecto quiere ofrecer a los profesionales que trabajan en la restauración de la Catedral (arqueólogos, restauradores, arquitectos,...) unas herramientas personalizadas, sencillas e intuitivas que les permitan utilizar una serie de recursos de gestión de la información contenida en el sistema sin tener que ser usuarios avanzados de el programa Autodesk Map.



Diferentes tipos de informaciones gestionadas en el SIM



Modelo tridimensional y base de datos enlazadas

4.- Desarrollo del proyecto. Metodología y resultados

4.1 Descripción de la metodología utilizada

La creación de herramientas informáticas va más allá del conocimiento de la sintaxis de un determinado lenguaje de programación. Además si las herramientas se realizan expresamente para un entorno geomático determinado el programador debe conocer perfectamente las características y necesidades de dicho entorno.

Este proyecto se ha planteado como un ejemplo real de cómo crear y utilizar herramientas informáticas en una situación concreta y de esta forma animar y servir de apoyo para que otros profesionales interesados en este camino se decidan a utilizar este recurso. Por ello, el trabajo realizado no se ha limitado a la elaboración de una serie comandos para su utilización exclusiva en la Catedral de Santa María sino que se han descrito todos los pasos necesarios que hay que dar para llegar a ejecutar un programa de VB.NET en un entorno Autocad.

El Sistema de Información Monumental (SIM) está conformado por información geográfica articulada en entidades polilínea 3D y por una serie de atributos de dichas entidades. Estos atributos pueden ser el resultado de estudios, de planes, de tratamientos realizados o de cualquier tipo de característica que pueda ser georeferenciable y están sistematizados en una serie de bases de datos relacionales. Quién escribe conoce perfectamente este entorno geomático ya que en su trabajo se encarga de su mantenimiento y gestión.

VB.NET es un lenguaje orientado a objetos que permite generar aplicaciones de Windows y utilizar librerías específicas de programas instalados para ejecutar tareas dentro de estas aplicaciones instaladas. En nuestro caso utilizamos este lenguaje para programar comandos que se utilizarán tanto como aplicaciones de Windows como para ser utilizados en Autocad y Autodesk Map 3D.

La herramienta específica para desarrollar este tipo de programas es Microsoft Visual Studio pero existe también una versión gratuita que nos permite hacerlo, Microsoft Visual Basic Express Edition. Esta versión gratuita tiene algunas carencias respecto a la de pago pero es suficiente para desarrollar el trabajo planteado en este Proyecto Fin de Máster.

En el proceso de aprendizaje de un lenguaje de programación se va conociendo su estructura, su filosofía, su sintaxis y su utilización. De esta manera VB.NET es introducido como un lenguaje que nos permite realizar aplicaciones para ser ejecutadas en el entorno Windows. Además se suele dejar abierta la posibilidad de utilizar este lenguaje para crear librerías que se puedan utilizar en otros programas. Para la realización de este proyecto se optó por esta última opción.

Sin embargo para poder empezar a programar un comando que realice una serie de tareas concretas en Autocad hay que tener en cuenta previamente una serie de ajustes. Estos ajustes no son evidentes y la bibliografía al respecto es bastante escasa.

Para la realización de este proyecto se ha realizado una búsqueda exhaustiva de bibliografía e información relativa a la programación de VB.NET en el entorno de Autocad. Entre toda la documentación utilizada es interesante destacar el libro *VB.NET for AutoCAD 2010* (indicado en el apartado de bibliografía) que ha sido una excelente guía introductoria para muchos aspectos de la programación.

La estructura de programa en VB.NET es la siguiente:

- Solución: Contienen el conjunto de recursos que se han programado para dar respuesta a un trabajo. Una solución puede estar compuesta por varios proyectos programados en diferentes lenguajes.
    - Proyecto: Se refiere a una aplicación, librería, … en un lenguaje determinado y que responde a una tarea concreta de la solución. Un proyecto podrá contener:
        - Propiedades del proyecto
        - Clases
        - Módulos
        - Formularios

Teniendo en cuenta que uno de los objetivos es animar a otros usuarios a realizar programas, el proyecto describe el software necesario para la realización y ejecución del aplicaciones informáticas, las configuraciones previas necesarias y las librerías utilizadas en el proceso de ejecución.

Entre las aplicaciones desarrolladas hay ejemplos que van desde una aplicación Windows instalable (Kategrafia), programas para crear y gestionar entidades del dibujos, comandos que acceden a las librerías internas de Autocad, aplicaciones más complejas que utilizan la tecnología de acceso a datos ADO.NET para terminar con una serie de herramientas que se incorporan al SIM de la catedral de Vitoria-Gasteiz. Para ello se ha realizado 3 soluciones diferentes y cada solución cuenta a su vez con un proyecto compuesto por varios formularios, módulos, clases y unas propiedades del proyecto definidas. Estas soluciones se han denominado SIM2D, UTIL y SIM y su contenido se describe en los siguientes puntos del presente apartado.

4.2 Primeros Pasos en VB.NET y Autodesk Map

A continuación se enumeran los pasos que debe dar cualquier usuario que quiera empezar a programar herramientas de Autocad utilizando VB.NET.

1.- Iniciar Visual Basic Express Tools

Este programa se puede conseguir de forma gratuita desde la página http://www.microsoft.com/express/Downloads/#2010-Visual-Basic
Una vez descargado debe ser instalado siguiendo las instrucciones del instalador.

2.- Crear un nuevo proyecto

Al iniciar el programa Visual Basic Express Tools el entorno que aparece nos resultará familiar ya que sigue el estilo de cualquier programa de Microsoft. Dentro del menú superior "Archivo" la primera opción es "Nuevo Proyecto…".

3.- Añadir referencias autocad

Desde la pantalla de propiedades del proyecto creado podemos acceder a la ficha de referencias y debemos añadir las siguientes librerías:

- *acdbmgd.dll. Database services and DWG file manipulation (like ObjectDBX)*
- *acmgd.dll. AutoCAD Application specific (like ObjectARX)*
- *ManagedMapApi.dll.*

Una vez que se han añadido las referencias, la seleccionamos y en sus propiedades asignamos COPY LOCAL = FALSE lo que nos va a permitir depurar los programas que vayamos desarrollando.

4.- Añadir los espacios de nombre de autocad.

Desde el explorador de soluciones se abre el archivo de clase, por defecto class1.vb y se añaden las siguientes líneas de código.

```
Imports System
Imports System.Collections
Imports Autodesk.AutoCAD.EditorInput
Imports Autodesk.AutoCAD.DatabaseServices
Imports Autodesk.AutoCAD.Colors
Imports Autodesk.AutoCAD.ApplicationServices
Imports Autodesk.AutoCAD.Geometry
Imports Autodesk.AutoCAD.Runtime
Imports Autodesk.AutoCAD.Windows
Imports Autodesk.Gis.Map
Imports Autodesk.Gis.Map.DisplayManagement
Imports Autodesk.Gis.Map.Project
```

... y similares

5.- Creación de un comando

Desde el archivo class1.vb introducimos el siguiente código

```
Public Class Class1
  <CommandMethod("HelloWorld")> _
        Public  Sub HelloWorld()
        Dim ed As Editor = Application.DocumentManager.MdiActiveDocument.Editor
        ed.WriteMessage("Hello World")
  End Sub
End Class
```

6.- Para poder depurar desde el programa hay que realizar alguna de estas opciones:

+ Introducir el camino de la aplicación para depurar (hay que tener visual studio)
+ Editar el archivo vbproj.user e insertar la negrita

```
<PropertyGroup Condition=" '$(Configuration)|$(Platform)' == 'Debug|AnyCPU' ">
<StartAction>Program</StartAction>
<StartProgram>C:\Program Files\Autodesk\AutoCAD Map 3D 2011\acad.exe</StartProgram>
<DebugSymbols>true</DebugSymbols>
<DebugType>full</DebugType>
<DefineDebug>true</DefineDebug>
<DefineTrace>true</DefineTrace>
<OutputPath>bin\Debug\</OutputPath>
<DocumentationFile>MyVBAcadApp.xml</DocumentationFile>
```

7.- Compilación del programa.

Una vez que hemos escrito el programa pulsamos BUILD y nos lo compila. Con esto tenemos un dll que se cargará con la instrucción NETLOAD en Autocad y después ya podemos ejecutar el comando tecleándolo en autocad.

8.- Modificación del registro.

Para que un dll se cargue automáticamente al abrir autocad es necesario editar el registro y añadir un carpeta (Prueba) y añadir los siguientes valores

```
[HKEY_USERS\S-1-5-21-2181049137-1717527822-2492260493-
1001\Software\Autodesk\AutoCAD\R18.1\ACAD-9002:40A\Applications\prueba]
"DESCRIPTION"="Prueba para cargar puntonet"
"LOADCTRLS"=dword:00000002
"MANAGED"=dword:00000001
"LOADER"="g:\\dokumentuak\\MASTER\\PFM\\SIM.NET\\SIM\\bin\\Release\\SIM.dll"
```

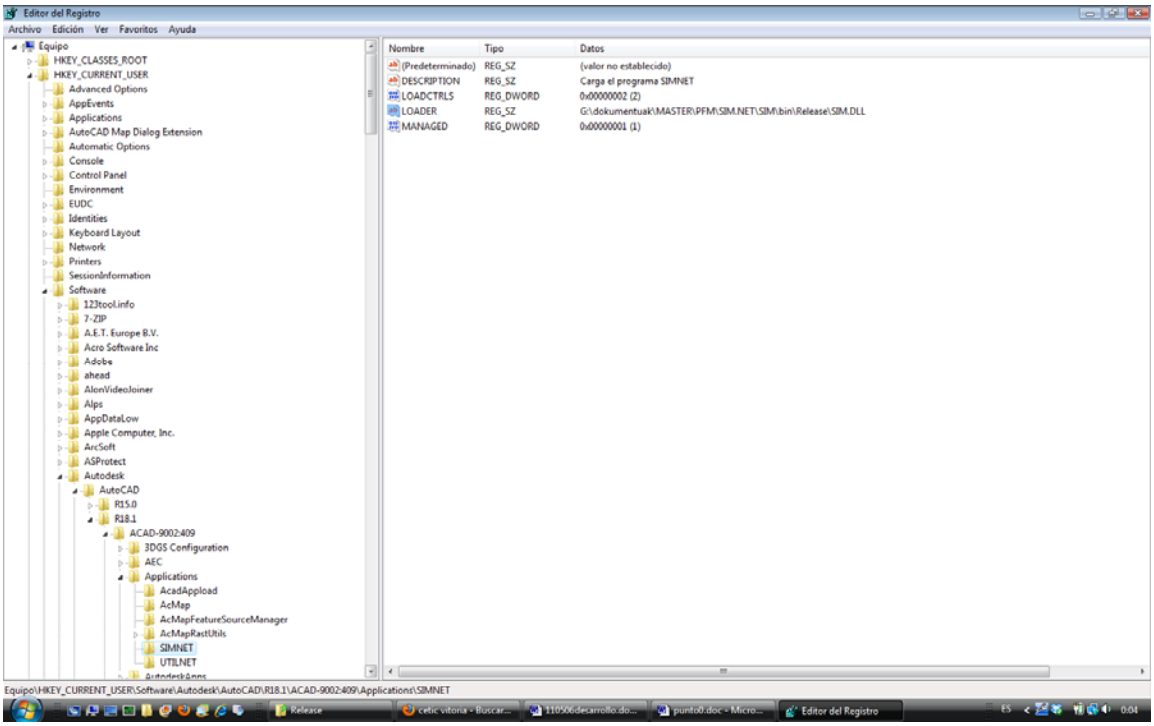


Imagen del registro en el que se han añadido las entradas correspondientes para que Autocad cargue automáticamente el dll indicado al iniciar el programa

4.3 Herramientas para el desarrollo de aplicaciones Windows.

Solución: SIM2D

Kategrafia

Como ya se ha indicado anteriormente geometría de la catedral está representada por un modelo tridimensional gestionado desde Autocad. Para complementar a este modelo y hacerlo accesible a usos más tradicionales se realizó una colección completa de planos compuesta por 13 plantas, 4 alzados y 33 secciones. En total son 50 archivos para el Estado Inicial y otros tantos para el Estado Proyectado y el Estado Actual.

Aunque existen planos índice en el que se indican dónde encontrar los planos referidos a diferentes partes de la catedral, los usuarios ocasionales tienen dificultades para encontrar y seleccionar el plano más adecuado para sus necesidades. Para solucionar este problema se ha creado Kategrafia.

Este módulo ha sido diseñado para ser utilizado de forma independiente y se distribuye como un ejecutable más (kategrafia.msi). Se utiliza para navegar por la cartografía de la catedral y muestra sobre los menus desplegables todas los nombres de las colecciones de planos, sus miniaturas y su acceso con un click (del botón derecho o izquierdo) a su ubicación en el disco o directamente a su archivo PDF.

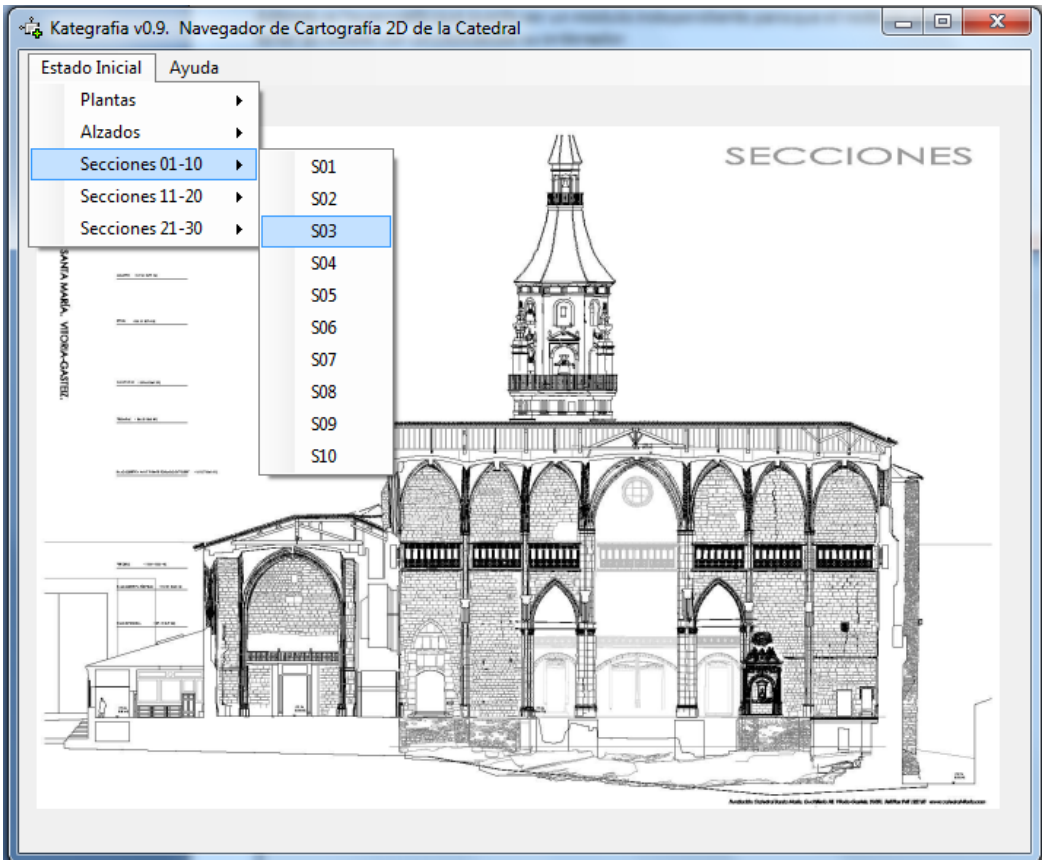


Imagen de la aplicación KATEGRAFIA

Una de las características más interesantes de las aplicaciones desarrolladas por Visual Basic .NET (y Microsoft Visual Studio) es la denominada “impacto cero”. Ésta es la terminología que emplea Microsoft para explicar que se puede distribuir una aplicación simplemente copiando el resultado de su compilación, sin necesidad de nada más: establecer entradas en el registro, incluir extensiones (.dll) en el directorio System de Windows, etc. Desde luego, es una característica muy interesante que elimina muchos de los problemas que se presentan actualmente en las aplicaciones Windows, donde parece que cualquier aplicación pueda modificar como quiera el sistema operativo.

Sin embargo, en nuestro caso el programa necesita mantener una estructura de subcarpetas definida para poder funcionar correctamente por lo que nos interesa crear un proyecto de instalación que nos permita ejecutar un programa de instalación “al uso” que mantenga la estructura de subcarpetas definida.

El usuario está acostumbrado a utilizar un programa específico para instalar las aplicaciones por lo que no le supondrá ningún problema. Además, esto representa algunas ventajas adicionales:

- La posibilidad de desinstalar automáticamente la aplicación, si así lo reclama el usuario, a través de Agregar o quitar programas del Panel de control.
- Si por cualquier razón la instalación falla (por ejemplo, porque el sistema operativo no es compatible con la aplicación .NET), el proceso anulará los pasos realizados para que el equipo quede en el mismo estado en que se encontraba antes de iniciar la instalación.

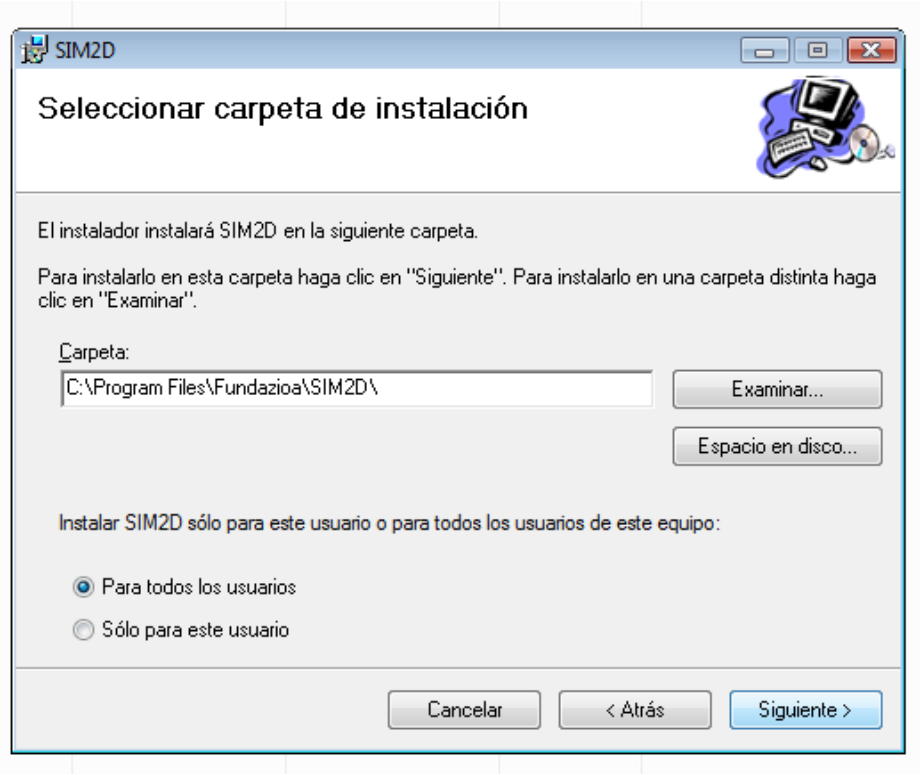


Imagen del programa de instalación desarrollado

4.4 Herramientas comunes para AutoCAD – Autodesk Map.

Solución: UTIL

Comando DesactivaCapa (vb_utilidades.vb)

Este comando nos invita a seleccionar una serie de elementos en el dibujo y lo que hace es desactivar las capas en las que se alojan los elementos seleccionados.

El desarrollo del programa se describe a continuación. Una vez seleccionados los elementos se comprueba que la selección es correcta y después se recorren todos sus elementos leyendo la capa en la que están. Se accede a la biblioteca de capas del dibujo y se congela la capa que coincide con la del elemento analizado. Si dicha capa es la actual o está congelada no se hace nada.

Es una herramienta muy útil para gestionar dibujo con un gran número de capas y en el que el usuario no conoce exactamente el significado de cada una.

Comando imexport (frm_imexport.vb)

Este programa permite manejar puntos. Tiene las siguientes opciones:

Visualizar Archivo: Se solicita un archivo con formato txt y a continuación se puede ver en el campo de texto situado a la derecha del cuadra de diálogo.

Importar puntos: Lee el fichero seleccionado e importa los puntos en el dibujo actual. Si cada registro del fichero tiene 3 campos (X,Y,Z) se importan directamente en la capa actual, si tiene 4 campos (X,Y,Z, Descriptor) se importan en la capa que se llame como el Descriptor (si no existe esa capa se importa en la capa actual)

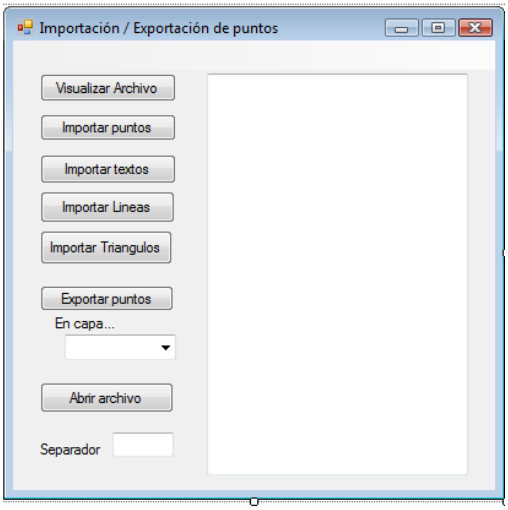
Importar textos: Funciona con registros que tienen entre 3 y 5 campos. El programa escribe un texto en las coordenadas indicadas en el registro y con la altura introducida por el usuario. El usuario decide qué campo es el que quiere añadir a la capa actual.

Importar líneas: Con registros de 6 campos del tipo X, Y, Z, X_1, Y_1, Z_1, el programa dibuja la línea que une los dos puntos incluidos en el registro. Para añadir líneas se utiliza la capa actual.

Importar triángulos: Con registros de 9 campos del tipo X, Y, Z, X_1, Y_1, Z_1, X_2, Y_2, Z_2, el programa dibuja en la capa actual el triángulo que une los tres puntos incluidos en el registro.

Exportar puntos (en capa). Este comando solicita introducir el archivo que se creará con las coordenadas de los puntos del dibujo. En la lista desplegable se puede indicar la capa que queremos utilizar como filtro para la exportación de puntos. En caso de dejarlo en blanco se exportarán los puntos de todo el dibujo (situado en capa activas). Se utiliza el valor introducido en el campo “separador” ya que será el que se utilice como separador de campos en el fichero de exportación de puntos. En caso de que este cuadro de texto esté en blanco el separador será el tabulador.

Abrir archivo: Aparece un letrero de dialogo en el que seleccionamos un fichero y nos lo abre utilizando el Bloc de Notas.



Formulario de Imexport

Programa Lote (vb_lote.vb)

La clase vb_lote.vb que está en la solución UTIL encuadrada en el proyecto SIM.NET corre sobre Autocad y define el comando LOTE.

Este comando solicita un archivo con extensión .txt que contiene los caminos y nombres de los archivos dwg sobre los que se quieren realizar una serie de tareas (subrutinas). Después de realizar las tareas guarda el archivo de dibujo .dwg, lo cierra y abre el siguiente. *Parámetros: Archivo txt con dibujos, nombre del cajetin.*

Las subrutinas realizadas para este apartado son:

definecajetin(nombrecajetin, nombreDwg): Agrega una definición de cajetin con el nombre del dwg indicado.

borraelementoscapa_espaciopapel(): Borra los elementos que están en el espación papel y en una capa que se especifica en el interior del programa. *Parámetros: Capa de la que borra todos los elementos*

insertaCajetin(nombrecajetin): Inserta en el espacio papel y en la capa especificado por el usuario el cajetin definido. *Parámetros: Capa en la que se inserta el bloque*

explotavaloresBloque(): Explota el bloque que está en una capa determinada. *Parámetros: Capa en la que se ha insertado el bloque*

cambiavalorestextos(): Cambia de valor los textos según la BBDD de cajetines2D.mdb. *Parámetros: Base de datos cajetines2D.mdb*

imprimePDF(): Imprime un pdf según define el usuario el papel, las plumillas, el zoom, la escala,...*Parámetros: Caracteristicas de impresión*

4.5 Herramientas para interaccionar con BD: ADO.NET

Solución: UTIL

Curso realizado e-learning de Visual Basic.NET y ADO.NET

El primer contacto con la programación en VB.NET fue a través de la asignatura del Máster "Herramientas informáticas para el geoprocesado". Para complementar los conocimientos adquiridos se ha realizado el curso de Visual Basic .NET dirigido por el CETIC (Centro de Tecnologías de la Información y la Comunicación) del ayuntamiento de Vitoria-Gasteiz ya que incluía en su temario el enlace a bases de datos a través de la tecnología ADO.NET que es la que se ha utilizado en algunos programas y comandos de este proyecto. Este curso con 30 horas de teoría y 44 de práctica fue realizado del 20 de enero de 2011 al 21 de abril de 2011.

Microsoft Visual Basic .NET
Programación

Curso de desarrollo de aplicaciones utilizando la tecnología de programación Microsoft .NET. El lenguaje utilizado es Visual Basic .NET, cuyas particularidades se estudian en la primera parte del curso. Después el curso se centra en la construcción de aplicaciones Windows y finalmente estudia en profundidad el acceso a bases de datos utilizando la tecnología ADO .NET.

Idioma
Spanish (Spain)
Requisitos
Browser: Microsoft Internet Explorer, 5.0
Permitir descargar controles ActiveX firmados
Horas de teoria 30
Horas de práctica 44

Dirigido por el CETIC www.**vitoria**-gasteiz.org/**cetic** *el temario es el siguiente:*

El lenguaje Visual Basic .NET
Introducción a .NET
Entorno Integrado de Desarrollo
Fundamentos de programación
Fundamentos de programación (II)
Matrices y estructuras de control
Procedimientos
Pensar en objetos
Pensar en objetos (II)
Herencia
Interfaces y espacios de nombres
Tipos de datos como clases

Aplicaciones Windows
Introducción
Formularios
Trabajar con menús
Barras de herramientas
Cuadros de diálogo
Controles básicos
Controles básicos (II)
Características gráficas de .NET
Características gráficas (II)
Características gráficas (III)
Eventos de ratón y teclado
Aplicaciones MDI
Excepciones

Acceso a bases de datos
Introducción
El lenguaje SQL
Introducción a ADO .NET
Conjuntos de datos
Conjuntos de datos (II)
Conjuntos de datos (III)
Actualizar el conjunto de datos
Estructura del DataSet
Programar el DataSet
Objetos Command y DataReader
Crystal Reports
Crystal Reports (II)
Proyectos de instalación

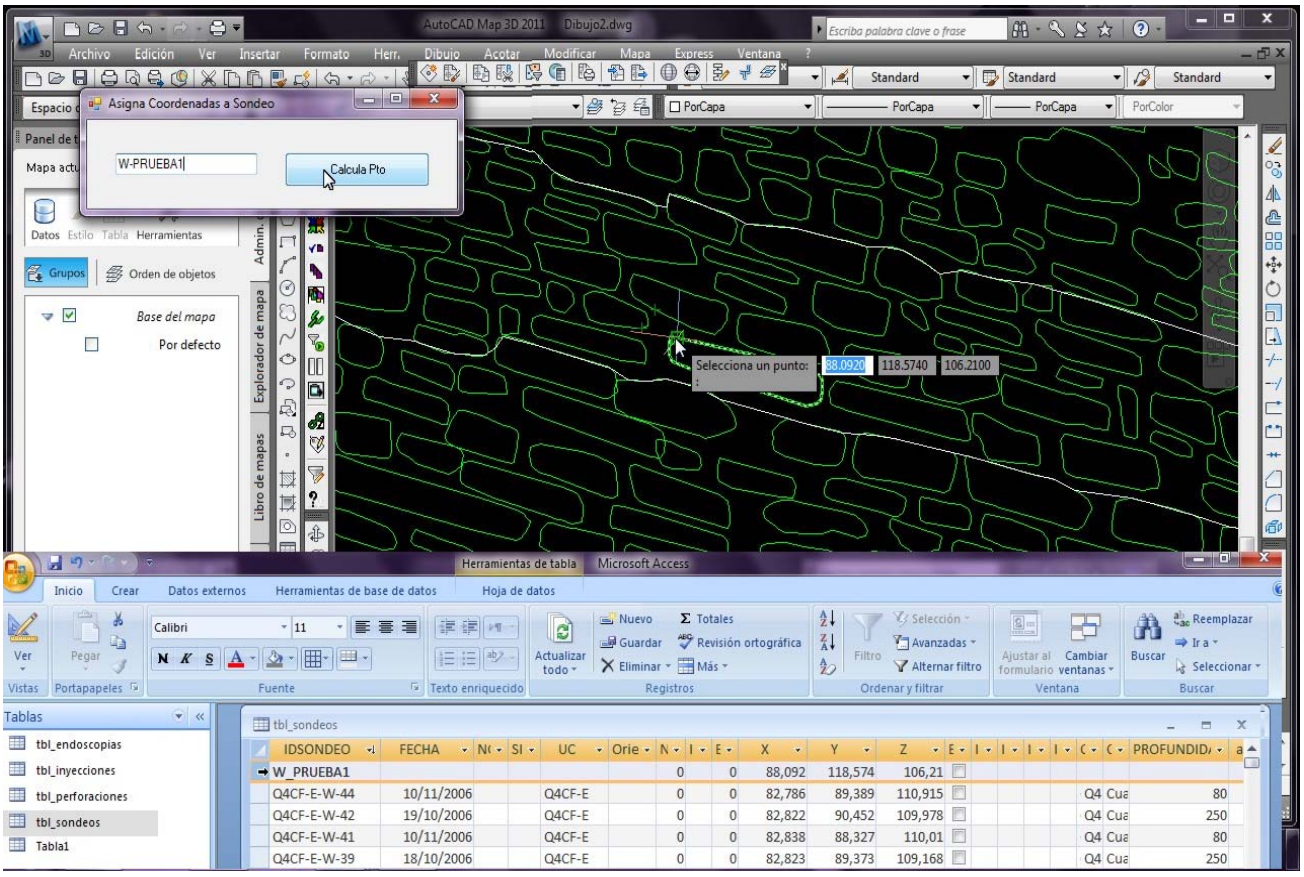
Comando Asignar coordenadas a sondeo (frm_asigcoorsondeo.vb)

Este programa crea un registro nuevo en la tabla de sondeos correspondiente y le asigna las coordenadas resultantes del punto seleccionado en el dibujo de Autocad.

El desarrollo es el siguiente:

Al ejecutar el comando *"AsignarCoordenadas*" el nombre del sondeo será el introducido en la casilla de texto y pulsando sobre el botón de comando "Calcula Pto" el programa indicará al usuario que seleccione un punto en el dibujo. Una vez seleccionado el punto se genera una instrucción SQL que inserta un registro en una tabla predefinida y a continuación vuelve a mostrar el letrero de diálogo. Si ocurre un error al insertar el registro el programa lo detecta y muestra un mensaje de error.

Este comando ha sido utilizado para georeferenciar los sondeos realizados. El programa es el encargado de añadir las coordenadas eliminando pasos intermedios. De esta forma se ahorra tiempo y se eliminan las posibilidades de error por parte del usuario al introducir de forma manual las coordenadas

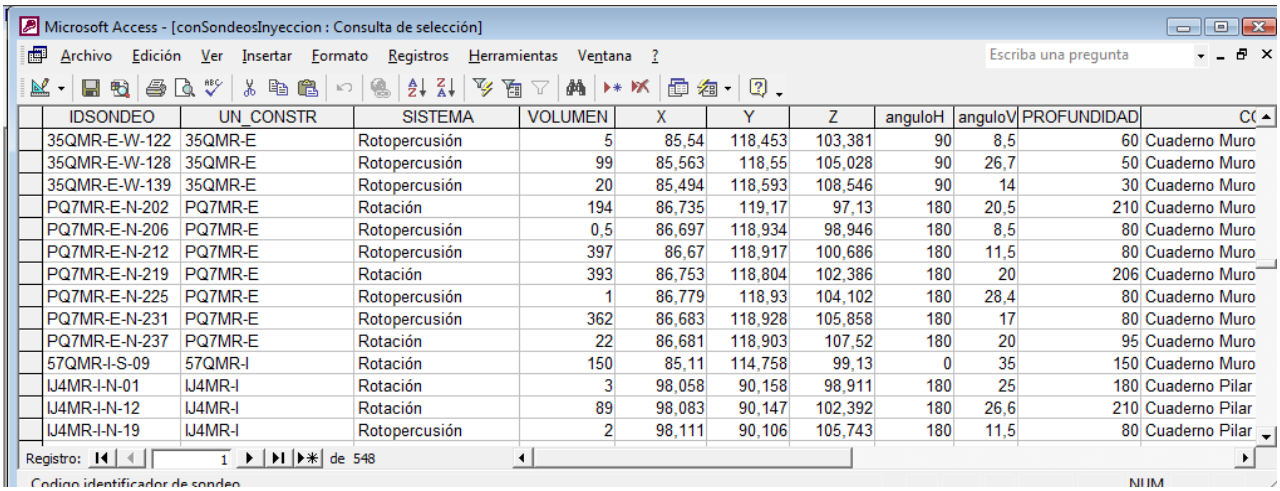


Se puede ver una imagen del formulario principal en la que el usuario ha escrito W-PRUEBA y el momento en el que selecciona las coordenadas sobre el dibujo. En la parte inferior se puede ver la tabla correspondiente a los sondeos en la que el programa ha creado un nuevo registro con el campo clave IDSONDEO que se corresponde con el texto introducido y con los campos X,Y y Z con los valores correspondientes.

Comando Creación de sondeos

Este programa permite incorporar al modelo tridimensional entidades líneas y cilindros que indican las perforaciones realizadas en el proceso de consolidación de muros y pilares realizado en la restauración de la Catedral de Santa María de Vitoria-Gasteiz.

El proceso de trabajo realizado parte de la identificación sobre el papel del lugar en el que se realizan las perforaciones. Los datos del proceso realizado en el sondeo (perforación, endoscopia, recuperación de testigo e inyección) son almacenados en una base de datos que entre otras características almacena en método de perforación (rotación o rotopercusión), azimut de la perforación, inclinación, longitud y litros inyectados. Sus coordenadas 3d han sido incorporadas previamente a la base de datos utilizando el comando “AsignarCoordenadas” descrito en el apartado anterior.


Microsoft Access - [conSondeosInyeccion : Consulta de selección]

Archivo Edición Ver Insertar Formato Registros Herramientas Ventana ? Escriba una pregunta

IDSONDEO	UN_CONSTR	SISTEMA	VOLUMEN	X	Y	Z	anguloH	anguloV	PROFUNDIDAD	C(
35QMR-E-W-122	35QMR-E	Rotopercusión	5	85,54	118,453	103,381	90	8,5	60	Cuaderno Muro
35QMR-E-W-128	35QMR-E	Rotopercusión	99	85,563	118,55	105,028	90	26,7	50	Cuaderno Muro
35QMR-E-W-139	35QMR-E	Rotopercusión	20	85,494	118,593	108,546	90	14	30	Cuaderno Muro
PQ7MR-E-N-202	PQ7MR-E	Rotación	194	86,735	119,17	97,13	180	20,5	210	Cuaderno Muro
PQ7MR-E-N-206	PQ7MR-E	Rotopercusión	0,5	86,697	118,934	98,946	180	8,5	80	Cuaderno Muro
PQ7MR-E-N-212	PQ7MR-E	Rotopercusión	397	86,67	118,917	100,686	180	11,5	80	Cuaderno Muro
PQ7MR-E-N-219	PQ7MR-E	Rotación	393	86,753	118,804	102,386	180	20	206	Cuaderno Muro
PQ7MR-E-N-225	PQ7MR-E	Rotopercusión	1	86,779	118,93	104,102	180	28,4	80	Cuaderno Muro
PQ7MR-E-N-231	PQ7MR-E	Rotopercusión	362	86,683	118,928	105,858	180	17	80	Cuaderno Muro
PQ7MR-E-N-237	PQ7MR-E	Rotación	22	86,681	118,903	107,52	180	20	95	Cuaderno Muro
57QMR-I-S-09	57QMR-I	Rotación	150	85,11	114,758	99,13	0	35	150	Cuaderno Muro
IJ4MR-I-N-01	IJ4MR-I	Rotación	3	98,058	90,158	98,911	180	25	180	Cuaderno Pilar
IJ4MR-I-N-12	IJ4MR-I	Rotación	89	98,083	90,147	102,392	180	26,6	210	Cuaderno Pilar
IJ4MR-I-N-19	IJ4MR-I	Rotopercusión	2	98,111	90,106	105,743	180	11,5	80	Cuaderno Pilar

Registro: 1 de 548

Código identificador de sondeo NUM

Consulta sobre la BD de sondeos utilizada para la reconstrucción 3D de los sondeos

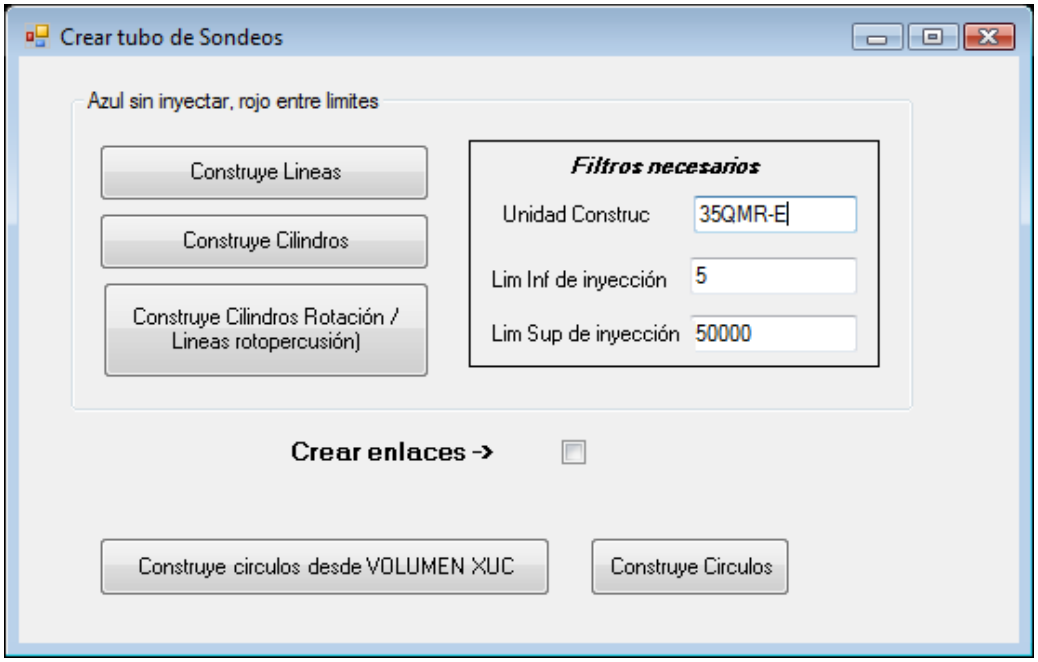
Para acceder al formulario principal el usuario debe escribir en la barra de comandos “creasondeos”. Una vez que el formulario aparece hay que definir un filtro para indicar en qué unidad constructiva se encuentran los sondeos que quiere dibujar y los límites superior e inferior del volumen de la inyección. Por defecto el límite inferior es 5 litros (cuando se inyecta menos se supone que únicamente se rellena el orificio realizado para la perforación) y el superior es 50000 litros.

El programa establece un código de colores por el cual los sondeos en azul indican que el volumen de inyección se encuentra entre los límites definidos y el color rojo significa que se encuentra fuera de los límites establecidos.

Para su representación 3d existen tres posibilidades:

- Todos los sondeos se muestran como líneas (cmd_lin)
- Todos los sondeos se muestran como cilindros (cmd_cil)
- Los sondeos por rotación (existe recuperación de testigo) aparecen como cilindros y los sondeos por rotopercusión (sin recuperación de testigo) se dibujan como lineas (cmd_rot_per)

Por último se ofrece al usuario la posibilidad de crear el enlace correspondiente con el campo IDSONDEO en la platilla de vinculo ENSAYOS definida en el SIM. Para ello se deberá seleccionar la casilla “Crear enlaces”.



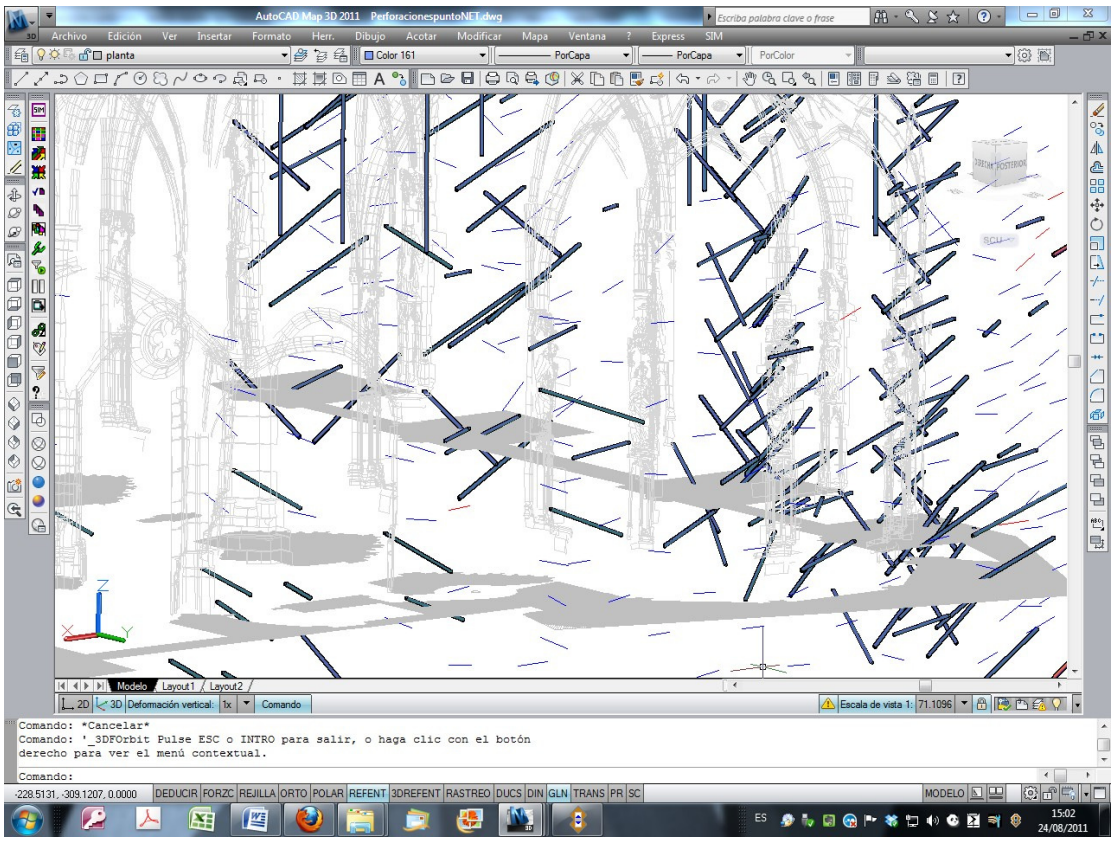
Formulario principal utilizado para construir los sondeos en el dibujo

En cualquiera de las modalidades de creación de sondeos elegida (líneas, cilindros o mixta) el proceso es similar.

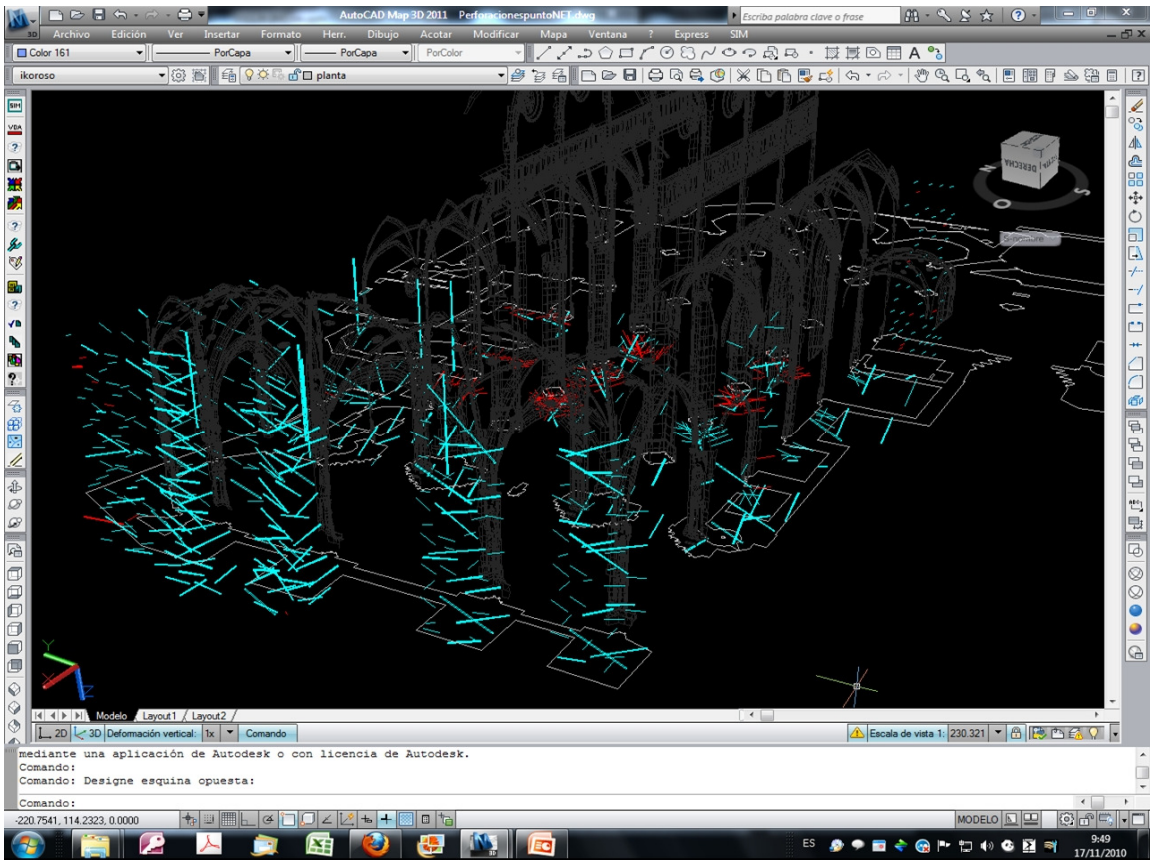
En cuanto al funcionamiento, el programa asigna las variables utilizadas para el filtro (unidad constructiva y limites inferior y superior), a continuación declara y asigna las variables que se utilizarán para la consulta a base de datos (nombre de la base de datos, nombre de la tabla, tipo de origen de datos, sentencia SQL,..) y se crea un objeto Dataset que representa un conjunto completo de datos, incluyendo las tablas que contienen, ordenan y restringen los datos, así como sus relaciones. A continuación se realiza para cada registro del Dataset seleccionado un proceso por el que se leen sus coordenadas, ángulos de incidencia, profundidad, volumen de inyección,… y se llama a una subrutina (construirsondeos/construircilindros) que es la encargada de interaccionar con el dibujo de autocad y crear la entidad correspondiente.

Esta subrutina prepara las variables necesarias para trabajar con entidades en el dibujo actual de autocad, calcula el punto final de la línea o del cilindro con los valores de los ángulos de incidencia y la profundidad y añade el elemento correspondiente al dibujo (línea/cilindro). Posteriormente calcula el color que debe asignar (rojo/azul) y si esta entidad debe ser enlazada realiza los pasos correspondientes y vuelve al código del comando pulsado y se realiza de nuevo el proceso para el siguiente registro del Dataset.

Finalmente, y si todo el proceso es satisfactorio, aparece un letrero de diálogo recordando los códigos de color y si se ha enlazado o no los elementos.



Cilindros y líneas creadas con los valores extraidos de la consulta a las tablas de sondeos (orientación, inclinación y profundidad) a través del módulo ADO.NET.



El proceso de inyección de muros y pilares ha sido sistemático, consiguiendose reforzar su estructura y hacerla más compacta, distribuyendo de forma correcta las cargas que soporta.

4.6 Herramientas específicas de Autodesk Map 3D

Solución: SIM

Como ya se ha explicado anteriormente la relación entre la información geométrica y la temática del Sistema de Información Monumental (SIM) de la Catedral de Santa María de Vitoria-Gasteiz se gestiona mediante el programa Autodesk Map 3D. Este programa es el implementado por la empresa Autodesk para trabajar con Sistemas de Información Geográfica ya que añade al programa básico de Autocad una serie de módulos que permiten el enlace a base de datos, la creación de topología, el acceso a formatos de datos de otros sistemas GIS, etc.

El manejo de esta herramienta no es excesivamente complejo pero la gran cantidad de posibilidades y opciones que ofrece obliga a tener de un grado de experiencia elevado para obtener un buen rendimiento. Mediante los comandos que se describen a continuación y que se recogen en el archivo SIM.dll se quiere simplificar las tareas más habituales del SIM de tal manera que un nuevo usuario pueda manejar rápidamente las herramientas básicas e importantes utilizadas en la gestión de la información del SIM.

La programación en VB.NET para el programa Autodesk Map 3D es menos habitual que la programación para aplicaciones web, aplicaciones Windows e incluso Autocad. También hay que tener en cuenta la gran cantidad de opciones y aplicaciones que desarrolla Autocad y pensar que las opciones programadas para el SIM de la Catedral de Vitoria-Gasteiz responden a necesidades concretas del monumento. Esta característica a puesto en valor de manera sobresaliente algunos foros específicos para VB.NET y AutoCAD. Sin embargo el autor de estas líneas ha echado en falta una comunidad más numerosa de desarrolladores que pudieran abarcar todos los rincones de Autodesk Map 3d.

Conviene recordar que el prototipo del SIM se definió durante los años 1997-98 y que en el año 2003 se redefinió y se adoptó el programa Autodesk Map para la gestión de la información. Posteriormente se desarrollaron algunas herramientas en lenguaje VBA. Sin embargo a partir de la versión 2010 AutoCAD deja de incluir soporte para VBA en el instalador (aunque lo mantiene como descarga) y en la versión 2012 está anunciado que no haya soporte para VBA. Es decir que todas las aplicaciones desarrolladas por este lenguaje dejarán de ser operativas. Sin embargo la actualización de VBA a VB.NET no es de ninguna manera automática y aunque comparten algunos comandos se puede decir que VB.NET es un lenguaje diferente. Por todo ello la actualización de VBA a VB.NET se ha realizado reprogramando desde cero todas las aplicaciones y aprovechando ocasionalmente algún recurso o algún comando.

A continuación se hace un repaso de las herramientas desarrolladas, se expone su justificación y se comenta el código utilizado en su desarrollo.

Esquema de enlaces con bases de datos

Como se ha comentado anteriormente Autodesk Map 3D nos permite enlazar las bases de datos con los elementos geométricos del dibujo. Para ello se debe definir en el dibujo una “Plantilla de vinculo” en la que se índica el campo, la tabla y la base de datos que queremos enlazar. Con un ejemplo entenderemos en qué consiste. Supongamos que definimos una plantilla de vinculo denominada PRUEBA y el indicamos que lo que queremos enlazar es el valor del campo clave “idficha” de la tabla “Indice” de la base de datos “prueba.mdb” que se encuentra en “c:\temp\prueba.mdb”. De esta forma enlazamos únicamente los valores del campo “idficha” pero a través de la plantilla de vinculo podemos acceder a todo tipo de consultas SQL con cualquier tabla de pruebas.mdb que tenga definida una relación con el campo “idficha”.

En nuestro caso se han definido cuatro plantillas de vínculo principales:

- MATERIALES: Se enlaza con el campo IDMATERIAL de la consulta con_Materiales de la base de datos construc.mdb.
- UC: Se enlaza con el campo UN_CONSTR de la consulta con_UC de la base de datos construc.mdb
- UE: Se enlaza con el campo IDFICHA de la consulta con_UE de la base de datos historia.mdb
- POLICRO: Se enlaza con el campo CODIGO de la consulta con_general de la base de datos policromias.mdb

Todas las entidades del modelo tridimensional del monumento (más de 500.000) tienen enlaces con la plantilla MATERIALES, UC y UE y además en el pórtico también tienen con POLICRO.

En las excavaciones únicamente está enlazado la plantilla UE. Esto es lógico ya que en excavaciones no podemos hablar del tipo de material (tipo de piedra, metal, vidrio, etc) ni existen Unidades Constructivas ni elementos polícromos.

Videotutoriales disponibles en internet

Los comandos implementados en este apartado cuentan con un videotutorial en el que se explica su funcionamiento dentro del entorno del Sistema de Información Monumental (SIM) de la Catedral de Santa María de Vitoria-Gasteiz.

El objetivo de estos videotutoriales es mostrar a cualquier usuario del SIM cómo funcionan los comandos implementados.

En relación al Proyecto Fin de Máster, los videotutoriales quieren contribuir a que cualquier persona interesada en conocer el trabajo desarrollado y que no tenga acceso a la información manejada en el SIM (información geográfica y bases de datos) pueda observar cómo funcionan los programas en un entorno configurado adecuadamente.

Para ello se ha creado en la plataforma YOUTUBE un canal denominado catedralvitoriaSIM (http://www.youtube.com/user/catedralvitoriaSIM) que aloja todos los videotutoriales realizados. Las direcciones concretas de cada comando se especificarán en el apartado correspondiente.

Comando *asoexc*:

En Autodesk Map 3D el proceso de trabajo comienza al asociar al proyecto actual los dibujos que serán consultados. Para realizar esta tarea el usuario selecciona la opción correspondiente y elige aquellos archivos que le interesan. Para esta elección el usuario identifica el nombre del dibujo pero en nuestro caso los nombres de los archivos (aunque tienen una denominación lógica) no son intuitivos por lo que es difícil que el usuario reconozca el contenido del archivo por el nombre.

El comando *asoexc* se utiliza para enlazar dibujos relativos a las excavaciones realizadas y para ello se nos muestra un letrero de diálogo en el que vemos una imagen en planta de la catedral y en diferentes colores aparecen una serie de zonas correspondientes a diferentes archivos. Los archivos están clasificados por las campañas de excavaciones realizadas en los últimos 12 años en el interior de la catedral y su entorno. El usuario selecciona la casilla de verificación que se encuentra en el interior de cada color y pulsando en el botón de comando asociar se asocian al dibujo. Evidentemente puede seleccionar varias casillas a la vez e incluso aparecen unos controles para seleccionar todas las casillas de edición o ninguna. También es posible disociar todos los dibujos ya asociados en el dibujo.

Esta herramienta permite al usuario identificar de manera visual, directa e intuitiva aquellos dibujos que desea incluir en su trabajo. Los archivos se encuentran dentro de la subcarpeta Excavaciones y es necesario añadir el alias SIGEXC asociado al camino en el que se ha instalado el SIM en el ordenador para que funcione. Esta tarea se describe en el apartado 4.6 Instalación del programa SIM.

El desarrollo del programa es sencillo. En un primer momento declaramos y definimos los elementos necesarios como son el proyecto actual, la colección de dibujos asociados y los dibujos asociados. Posteriormente recorremos todas las casillas de verificación y en caso de que estén seleccionadas asociamos el dibujo correspondiente.

Videotutorial: http://www.youtube.com/watch?v=QGcMmQtkrgo

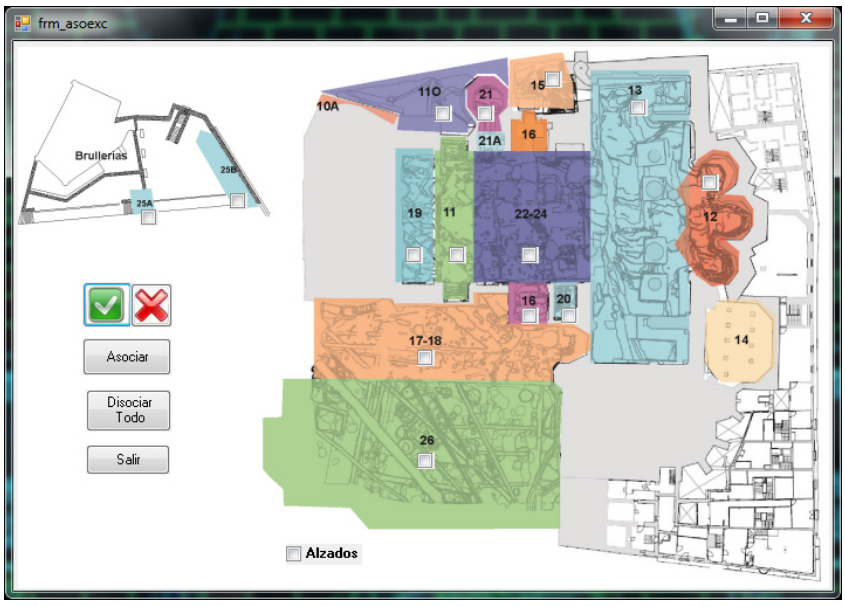


Imagen en planta de la catedral en la que aparecen las diferentes campañas de excavaciones acompañadas de casillas de verificación para permitir el enlace de los dibujos correspondientes de forma visual e intuitiva

Comando asoext.

Este comando es análogo al explicado en la página anterior pero aplicado a los dibujos que contienen el modelo tridimensional exterior de las fábricas del monumento. La codificación utilizada con estos archivos no permite una identificación automática en base al nombre de archivo por lo que es una herramienta muy útil para comenzar a trabajar. En total estamos hablando de 21 archivos con nombre del tipo CNGH-E.dwg, PTN-E.dwg, SYAB-E.dwg y similares y guardados en la carpeta VOLUMENES.

La primera versión de este comando se diseñó en lenguaje VBA y se ha reprogramado (incluyendo los recursos gráficos) en lenguaje VB.NET. El funcionamiento es similar al explicado anteriormente y el usuario puedes seleccionar uno, varios o todos los archivos y enlazarlos directamente desde el formulario.

Cada casilla de edición lleva asociado un nombre correspondiente al archivo que representa y accediendo a la información del proyecto actual se ejecuta un comando que asocia el dibujo correspondiente al proyecto actual.

Videotutorial: http://www.youtube.com/watch?v=QGcMmQtkrgo

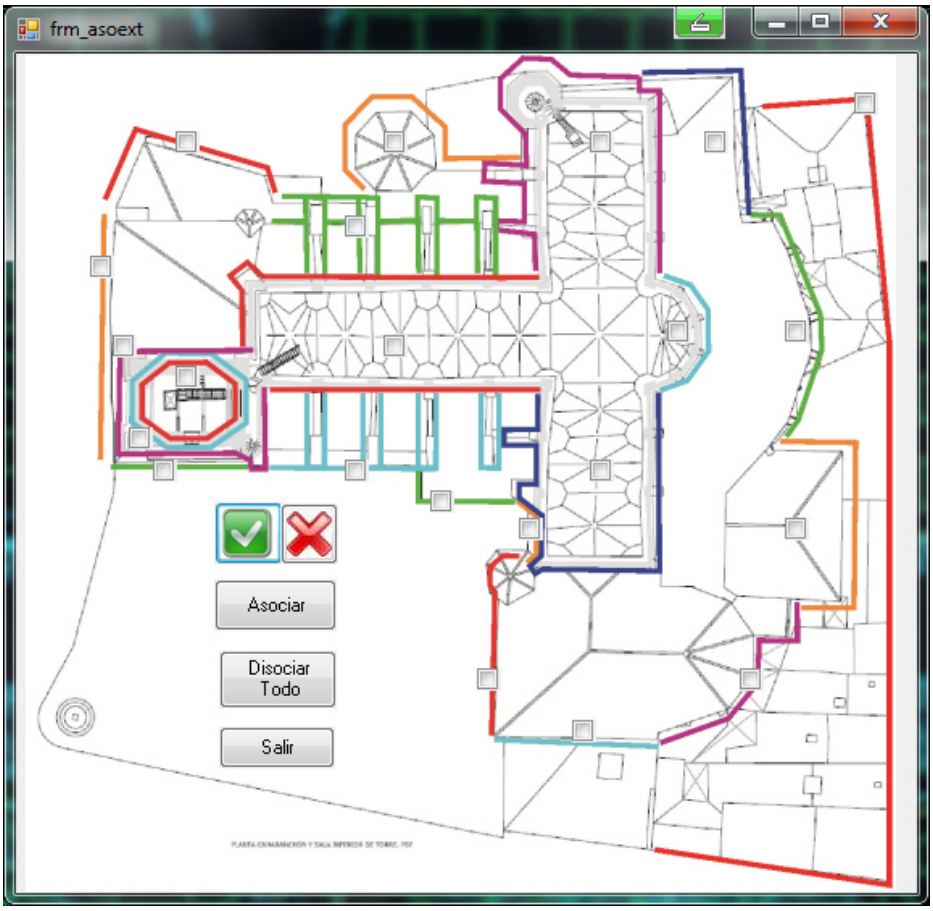


Imagen en planta a nivel de entrecubierta de la catedral en la que se puede observar las diferentes partes en las que se han dividido el exterior de las fábricas de la catedral. Aparecen las casillas de verificación, los botones para seleccionar (desseleccionar) todas las casillas y los botones de comando para asociar las partes seleccionadas y para disociar todos los dibujos ya enlazados.

Comando asoint.

Al igual que los anteriores comandos, éste permite al usuario seleccionar aquella parte del interior de la catedral que necesita incorporar al proyecto actual. En este caso la distribución de los diferentes archivos se hace en función de los espacios que quedan definidos por las naves y los tramos, resultando una especie de cuadrícula. En total 45 archivos que se localizan en la subcarpeta ESPACIOS y que se asocian siguiendo el alias SIGINT. Al igual que los anteriores tiene opciones para seleccionar o quitar la selección de todas las casillas, para asociar los archivos correspondientes a las casillas seleccionadas o para disociar todos los archivos asociados.

La distribución en cuadrícula se basa en la descomposición por espacios del monumento que se realizó durante la restitución fotogramétrica realizada en los años 1996-1998 y que identificaba a cada elemento constructivo en función de su situación en esta cuadrícula.

En definitiva, el usuario que inicia un proyecto en el SIM accede a herramientas visuales que le permiten seleccionar aquellas partes del interior, exterior o excavaciones que necesita. Esta tarea la puede realizar sin necesidad de conocer la denominación de cada fichero o la carpeta en la que esta almacenado.

Videotutorial: http://www.youtube.com/watch?v=QGcMmQtkrgo

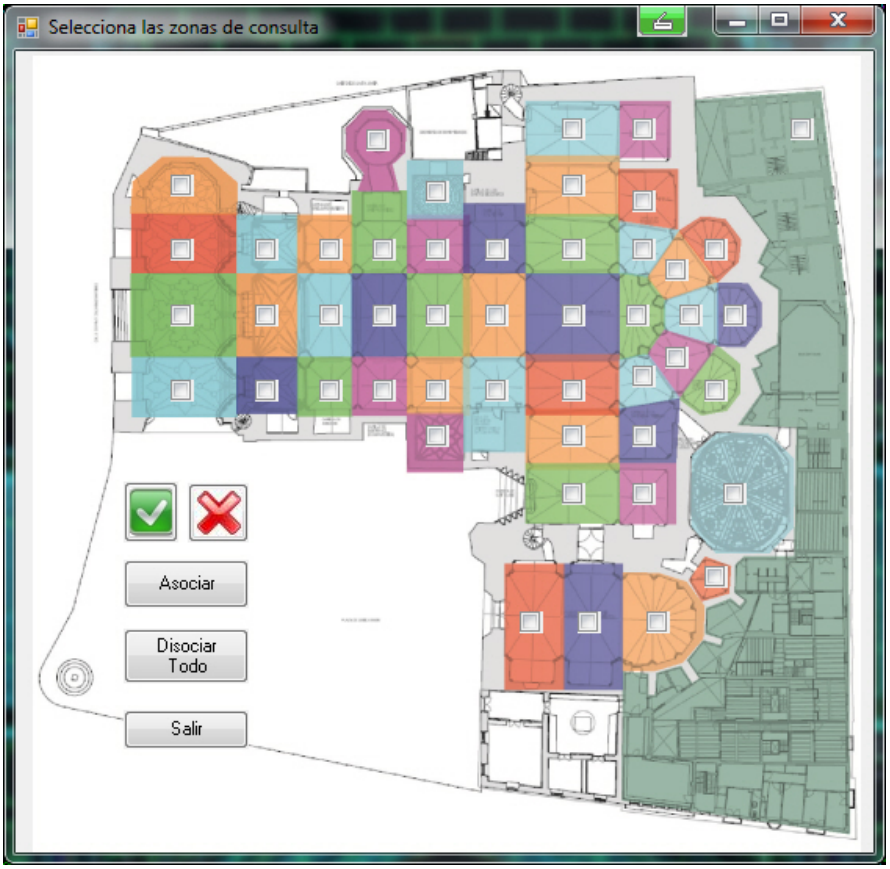


Imagen en planta de la catedral en la que se identifican los 45 archivos pertenecientes a las subcarpeta ESPACIOS que contienen la información geométrica del interior de la catedral

Comando consim.

Como se ha comentado anteriormente todas las entidades del dibujo están enlazadas con varias plantillas de vinculo a través de un campo clave. El objetivo de este comando es visualizar aquellas entidades gráficas que cumplen con una determinada condición temática. Para ello Autodesk Map 3D permite realizar una consulta SQL sobre las bases de datos enlazadas y mostrar aquellas entidades de los dibujos asociados que cumplen las condiciones solicitadas.

Mediante el comando que describimos a continuación este proceso se simplifica significativamente ya que el usuario únicamente debe seleccionar en un desplegable el campo desea consultar, asignar el valor que toma el campo consultado y elegir el color en el que quiere que se resalten las entidades consultadas.

Los valores que ofrece el desplegable no se limitan a los valores del campo vinculado sino también a otros valores relevantes de la base de datos.

Esta simplificación de la tarea de consultar es posible porque el comando es capaz de generar automáticamente las consultas SQL necesarias para cada valor seleccionado. Cada campo posee unos valores propios necesarios para realizar la consulta y para ello se ha diseñado una base de datos denominada SOPORTE.MDB que se incorpora en la subcarpeta SUPPORT y que contiene todos los valores necesarios para generar consultas SQL de forma automática.

Esta base de datos cumple una función muy importante ya que si el usuario quiere añadir nuevos campos a las consultas predefinidas no tendrá que tocar nada del código de programación sino que añadirá un registro en la tabla conIndice de la base de datos de soporte.

conIndice : Tabla

Indice	cplantilla	ccampoclave	cTituloTematico	cFichTxtTematico	textonum
UC	UC	UN_CONSTR	Unidad Constructiva	consul.txt	T
Ensayos	ENSAYOS	idensayo	Ensayos realizados	consul.txt	T
Materiales	MATERIALES	IDMATERIAL	Materiales	tem_materiales.txt	T
UE	UE	IDFICHA	Unidades Estratigráficas	consul.txt	N
Actividad	UE	idactividad	Actividades	consul.txt	N
Grupo de actividad	UE	idgrupo	Grupos de Actividad	consul.txt	N
Fase	UE	fase	Fases	tem_fases.txt	N
Periodo	UE	periodo	Periodos	tem_periodos.txt	N
Piedras	PIEDRAS	numero	Piedras	tem_piedras.txt	N
Policromias	POLICRO	codigo	Policromias Especiales	consul.txt	T

Registro: 1 de 13

Tabla en la que se definen todas las variables que se utilizan para programar las consultas prediseñadas

El usuario debe realizar el proceso siguiente:

1. Elegir el campo que se va a consultar de una lista desplegable. Los valores de esta lista se rellenan al lanzar el formulario y son los valores del campo indice de la tabla conIndice. (p.e. actividad)
2. Indicar el valor que toma el campo señalado para esta consulta (p.e. 25)
3. Indicar el color en el que queremos que se muestren los resultados (p.e. 1)

4. Pulsar el botón de comando “ejecutar” el programa va a la tabla conIndice de la base de datos soporte.mdb y lee la información asociada al campo marcado.

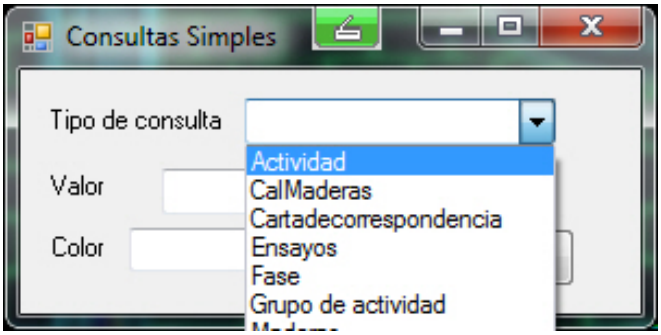


Imagen del letrero de diálogo para la realización de consultas

Suponiendo que hemos elegido “Actividad”, el programa consulta la base de datos SOPORTE.MDB y extrae el nombre de plantilla de vinculo correspondiente “UE” , el campo a consultar “idactividad” y el tipo de campo “numérico”. Con esta información el programa crea una consulta sobre Autodesk MAP del tipo SQL y la sentencia utilizada es construida en función de los valores obtenidos. En nuestro caso:

```
Dim sqlcon As New Query.SqlCondition
sqlcon.WhereCondition = “idactividad=25”
sqlcon.LinkTemplate = “UE”
```

Como resultado aparecerá en el dibujo aquellas entidades que tengan vínculos con la platilla UE y que cumplan la condición de que idactividad=25. Estas entidades tendrán el color 1 que en AutoCAD se corresponde con el rojo.

Además de lo ya explicado el programa permite introducir varios valores en la casilla correspondiente. Estos valores deben ir separados por comas y el programa los interpreta y crea la condición “where” para las entidades que cumplen un valor o el otro. Es decir los diferentes valores son interpretados por el operador booleano OR.

P.e. si en la casilla de valor introducimos “25,136,14” el programa lo interpretará:

```
Dim sqlcon As New Query.SqlCondition
sqlcon.WhereCondition = “idactividad=25 OR idactividad=136 OR
idactividad=14”
sqlcon.LinkTemplate = “UE”
```

y asignará el color correspondiente a todas aquellas entidades gráficas cuya UE asociada pertenece a la actividad 25, 136 o 14.

Un programa similar fue implementado por el autor hace algunos años en lenguaje VBA, la versión que se presenta es en lenguaje VB.NET y sus elementos gráficos también han sido rediseñados.

Videotutorial: http://www.youtube.com/watch?v=MecS1z6IxTY

Comando conmul.

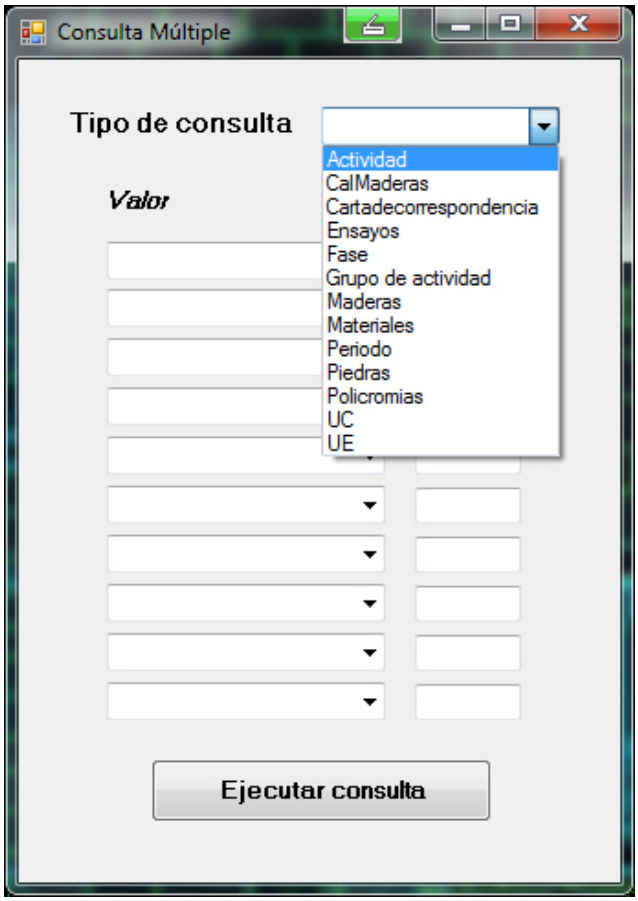
Este comando es una evolución del comando anterior. Da la opción al usuario de introducir una serie de valores referidos a una plantilla de vinculo y visualizarlos con códigos de color diferentes. De esta forma el usuario puede visualizar en el plano entidades coloreadas según el valor (o valores) introducidos. Está limitado a diez códigos de color diferentes, cantidad más que suficiente en la mayoría de los casos.

En modo usuario la forma de interaccionar es similar al comando "*consim*" por lo que no se repetirá lo indicado en el apartado interior. Hay que recordar que en las casillas de valor el usuario puede introducir uno o varios valores separados con comas y el programa los interpretará siguiendo el operador OR.

En lo referente a la programación este comando también es similar al anterior, pero el programa va analizando el valor de cada desplegable y si su contenido no está vacio ejecuta una consulta simple. En resumen se puede definir la consulta múltiple como una serie de consultas simples.

Al igual que el anterior comando sobre esta idea el autor desarrollo un programa en lenguaje VBA y ha sido reprogramado en lenguaje VB.

Videotutorial: http://www.youtube.com/watch?v=MecS1z6IxTY



Formulario utilizado para realizar consultas múltiples sobre los dibujos asociados al proyecto de Autodesk Map

Comando contem.

Las herramientas de consultas anteriormente descritas estaban limitadas a una serie de valores introducidos por el usuario. Sin embargo, habitualmente se desea crear un mapa temático con todos los valores referidos a una característica que aparece el dibujo y no solo con uno (*consim*) o varios valores (*conmul*). Para realizar esto se ha creado la herramienta "contem".

El proceso a seguir es el siguiente:

1.- Seleccionar el campo de la base de datos que se desea consultar. Para ello se muestra un desplegable con una serie de valores predefinidos.

2.- Guardar los valores de la consulta. Dependiendo del campo seleccionado los valores estarán predefinidos o deberán ser introducidos por el usuario. Esto es debido a que algunos campos como Fase, Periodo, Materiales,… tienen unos valores predefinidos que no superan los 20 registros y por lo tanto estos registros son seleccionados automáticamente. Sin embargo si seleccionamos otros campos como UE, UC, los registros correspondientes se cuentan por miles. En este último caso no tiene sentido asignar previamente un color a cada UE sino que el usuario deberá introducir todos los registros contenidos en el dibujo actual. Para ello deberá utilizar la herramienta RESDIB que se explicará más adelante. En definitiva si el campo elegido tiene valores predefinidos esta casilla permanecerá desactivada mientras que si es el caso contrario deberemos pulsar el botón e introduciremos los valores correspondiente en el archivo txt que nos abre automáticamente el block de notas.

3.- Ejecución de la consulta. Pulsando este botón el programa va ejecutando una a una las consultas correspondientes a cada valor.

4.- Crear leyenda. Desde aquí se crea un nuevo dibujo que contiene la leyenda correspondiente a la consulta realizada.

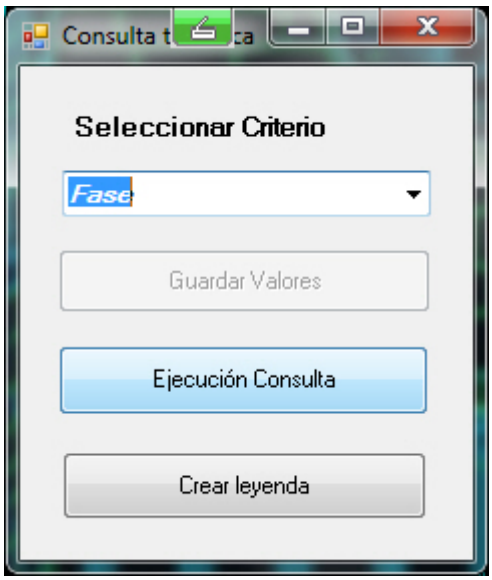


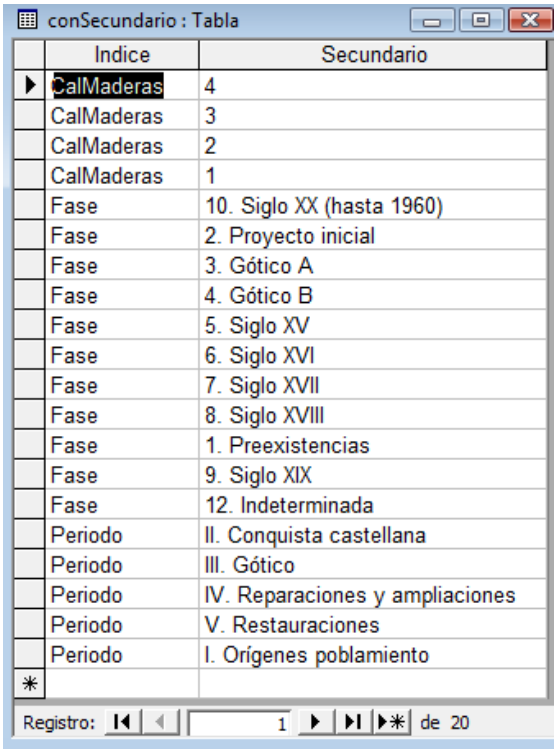
Imagen del formulario utilizado para el comando conTem

Aunque la austera apariencia del letrero de dialogo puede invitar a pensar que este comando es sencillo de programar, los procesos ejecutados son complejos y se describen a continuación.

a) Los valores del desplegable se obtienen mediante una consulta SQL sobre la base de datos soporte.mdb.

```
    archivodatos = "SOPORTE.MDB"
sentencia = "SELECT * FROM conIndice ORDER BY indice"
```

b) Una vez que se ha actualizado el valor del desplegable el programa comprueba si los valores del campo seleccionado se pueden predefinir (ya que su número no es elevado). Para comprobar esto el programa realiza una consulta sobre la tabla conSecundario de Soporte.mdb en la que aparecen los campos que tienen un número limitado de valores junto a esos valores. A continuación se puede ver una imagen con el contenido de la tabla conSecundario.



conSecundario : Tabla

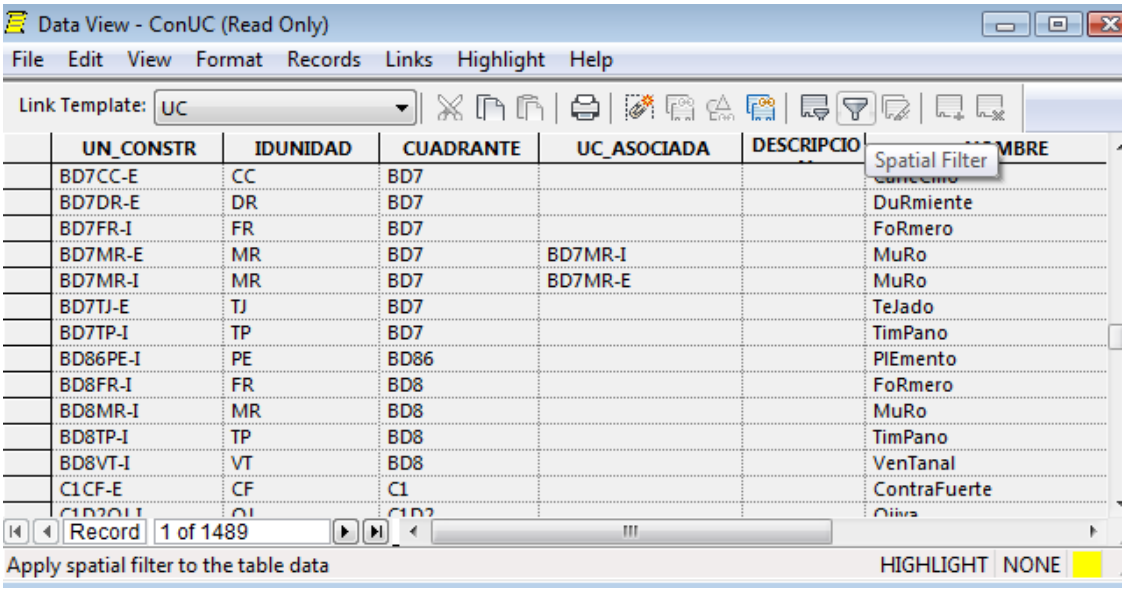
Indice	Secundario
CalMaderas	4
CalMaderas	3
CalMaderas	2
CalMaderas	1
Fase	10. Siglo XX (hasta 1960)
Fase	2. Proyecto inicial
Fase	3. Gótico A
Fase	4. Gótico B
Fase	5. Siglo XV
Fase	6. Siglo XVI
Fase	7. Siglo XVII
Fase	8. Siglo XVIII
Fase	1. Preexistencias
Fase	9. Siglo XIX
Fase	12. Indeterminada
Periodo	II. Conquista castellana
Periodo	III. Gótico
Periodo	IV. Reparaciones y ampliaciones
Periodo	V. Restauraciones
Periodo	I. Orígenes poblamiento

Registro: 1 de 20

Imagen de la tabla conSecundario

Como se puede observar en la imagen, aparecen los valores predefinidos que puede tomar un campo indice. Si el valor seleccionado del desplegable coincide con algún valor de esta tabla el programa toma estos valores para hacer la consulta y establece el botón de comando de “Guardar Valores” como inutilizable.
Si el valor no aparece (UE, UC, ACTIVIDAD, POLICRO,…) el programa interpreta que existen una gran cantidad de valores para ese campo y se activa la casilla de “Guardar valores”

c) Si el usuario debe guardar los valores correspondiente al campo seleccionado que se muestran en el dibujo, debe realizarse una consulta sobre el dibujo. Para ello se puede utilizar el Visor de datos y su herramienta “Spatial filter” incluidos en el software de Autodesk Map o el comando desarrollado en este proyecto RESDIB que se explica más adelante en este apartado.



UN_CONSTR	IDUNIDAD	CUADRANTE	UC_ASOCIADA	DESCRIPCIO	MBRE
BD7CC-E	CC	BD7			
BD7DR-E	DR	BD7			DuRmiente
BD7FR-I	FR	BD7			FoRmero
BD7MR-E	MR	BD7	BD7MR-I		MuRo
BD7MR-I	MR	BD7	BD7MR-E		MuRo
BD7TJ-E	TJ	BD7			TeJado
BD7TP-I	TP	BD7			TimPano
BD86PE-I	PE	BD86			PlEmento
BD8FR-I	FR	BD8			FoRmero
BD8MR-I	MR	BD8			MuRo
BD8TP-I	TP	BD8			TimPano
BD8VT-I	VT	BD8			VenTanal
C1CF-E	CF	C1			ContraFuerte

Herramienta Data View y localización de la herramienta Spatial Filter

Una vez seleccionados los valores, el usuario seleccionará el botón de comando "Guardar Valores" y automáticamente se abrirá una instancia del "Bloc de Notas" con el archivo correspondiente (se obtiene del valor del campo *cFichTxtTematico* de la tabla conIndice de soporte.mdb, ver imagen en página 26). El usuario copiará los valores en este archivo, eliminando los valores previos.

d) Ejecución de consulta. Pulsando este botón el programa realiza las consultas correspondientes al dibujo teniendo en cuenta los valores establecidos para el campo seleccionado (bien los toma de la tabla conSecundario o del archivo txt del campo *cFichTxtTematico*). Los colores que asigna a cada valor los toma del fichero "colores.txt" situado en la carpeta soporte del programa. Este fichero es un listado en el que cada línea es un número que corresponde con un color de Autocad.
La mecánica de la consulta es similar a la descrita en el comando consim, pero en vez de para un único valor se hace una consulta para cada uno de los valores establecidos a partir de la tabla conSecundario o a partir de los guardados en el fichero consul.txt
El resultado es visible en el dibujo en el que sus entidades se han coloreado en función de los valores asociados.

e) Cuando realizamos un mapa temático basado en colores lo que hacemos es asignar un color a cada entidad en función de su valor respecto a un determinado campo. Es imprescindible que el usuario pueda conocer el valor que corresponde a cada color, esto es, su leyenda. Pulsando en el botón "Crear leyenda" el programa creará un nuevo dibujo basado en la plantilla "leyenda.dwt" (suministrado en la carpeta soporte) y aparecerán los valores utilizados en la consulta junto con el color asignado de tal manera que el usuario pueda incorporar al dibujo su leyenda correspondiente, obteniendo un mapa temático. Estos valores los toma el programa del archivo consul.txt que ha sido utilizado para realizar las consultas pertinentes.

Videotutorial: http://www.youtube.com/watch?v=DTwjqx24C40

Comando objinf.

En los comandos anteriores hemos visto cómo consultar aquellas entidades que cumplen con una determinada condición. A continuación vamos a explicar un par de herramientas cuyo objetivo va en la dirección complementaria es decir, el usuario selecciona una (o varias) entidad y el programa le devuelve los valores asociados. Es decir, seleccionando una entidad podemos conocer información temática multidisciplinar referida a ella. Este comando fue programado en lenguaje VBA y ha sido reprogramado totalmente en código VB.NET.

Uno de los objetivos que se buscan con la programación de herramientas para Autodesk Map es simplificar los procesos que habitualmente son ejecutados en la gestión del SIM. En este caso la mecánica es extremadamente sencilla, el usuario pulsa el botón selección, selecciona una entidad y los valores asociados aparecen en el formulario. Además es posible acceder a su ficha correspondiente para el valor seleccionado pulsando sobre el icono correspondiente.

Evidentemente tras esta sencillez se esconde cantidad de código que es aportado en los anexos de este libro. Antes de explicar el funcionamiento interno del programa explicaremos dos subrutinas que se repiten en algunos programas y que tienen un papel central en este comando.

a) A*breformu*. Esta subrutina adquiere los valores de la plantilla de vinculo (el nombre completo de la base de datos y el nombre del formulario) y los asigna a la sentencia SQL. Cuando esta subrutina se ejecuta el nombre de la plantilla de vinculo aparece en una campo de texto y el resto de los campos se consiguen consultando las tablas conIndice y enlacesBD de soporte.mdb.

enlacesBD : Tabla

cPlantillaVinculo	cBasedeDatos	cTablaBase	cCampoClave	cFormularioMostrado
ENSAYOS	construc.mdb	conEnsayos	IDEnsayo	frmEnsayos
ENTERRAMIENTO	historia.mdb	con_enterra	ENTERRAMIENTOS	frmEnterra
Madera	construc.mdb	conMaderas	CLAVE	frmmaderas
MATERIALES	construc.mdb	ConMateriales	IDMATERIAL	frmMaterialesImg
PIEDRAS	construc.mdb	tblPiedras	NUMERO	frmPiedrasImg
POLICRO	policromias.mdb	con General	CODIGO	policromias
UC	construc.mdb	conUC	UN_CONSTR	frmUC
UE	historia.mdb	conUE	IDFICHA	frmUnidades

Registro: 8 de 8

Imagen de la tabla enlacesBD de la base de datos SOPORTE.MDB

b) *Editlink*: El objetivo de esta subrutina es devolver el valor enlazado a una determinada entidad para una plantilla de vinculo indicada. Cuando esta subrutina es llamada ya se conoce el id del objeto y el nombre de la plantilla de vinculo. En caso de que el objeto no tenga vinculado ningún valor el programa devuelve el valor 0 si ha detectado que el campo clave es numérico o el valor “” si ha detectado que el campo clave es texto.

Desarrollo:

1.- Pulsar el botón de comando selección: El programa declara las variables necesarias, les asigna sus valores correspondientes y solicita al usuario que seleccione una entidad del dibujo. Una vez seleccionada esta entidad el programa recorre todas las plantillas de vinculo definidas en el dibujo y consulta el valor enlazado (mediante la *subrutina Editlink*). Cuando la

respuesta es positiva el programa detecta cuantos valores han sido escritos anteriormente y los escribe a continuación en las casillas de texto correspondientes.

2.- Botón de imagen *access*: Una vez que se han calculado los valores correspondientes el usuario puede pulsar el botón de *access* si quiere que se abra un instancia de este programa y automáticamente se muestre la vista de formulario correspondiente al valor descrito en los campos del formulario. Para realizar esta tarea el programa lee la plantilla de vinculo y el valor enlazado y utilizando la *subrutina "Abreformu"* devuelve nombre del fichero, formulario, camino y sentencias necesaria. Una vez realizada la consulta a la ficha la instancia de access debe ser cerrada.

Videotutorial: http://www.youtube.com/watch?v=FdAe0cylC8A

Imagen del formulario del comando OBJINF

Comando resdib.

El anterior comando permitía extraer la información almacenada en una determinada entidad de dibujo. Sin embargo habitualmente queremos conocer los valores de una serie de entidades. Para ello se utiliza el comando *resdib*. La mecánica del formulario es sencilla ya que el usuario pulsa al botón de comando "selecciona objetos" y se marca en el dibujo aquellas entidades que quiere consultar. El programa accede a la información almacenada de cada entidad y muestra un listado con los diferentes valores que aparecen en el conjunto de objetos para cada plantilla. Por defecto aparecen los valores enlazados a la plantilla UC pero quedan almacenados para todas las plantillas disponibles. El usuario no tiene más que cambiar la plantilla consultada en la casilla desplegable y automáticamente se actualizarán los valores de la lista sin tener que volver a realizar otra selección. Además existe la posibilidad de realizar la consulta sobre todos los elementos del dibujo actual pulsando el botón de comando "Consulta dibujo actual". Con esta opción el usuario no debe realizar ninguna selección en el dibujo.

En lo referente al código el programa utiliza tres subrutinas principalmente:

a) *Abreformu.* Es la misma subrutina explicada en el apartado anterior
b) *Grupolink* (objcode2). Esta subrutina crea un fichero de texto que se llama como la plantilla de vinculo definida como actual y cuyos registros son los distintos valores enlazados con los elementos seleccionados. La subrutina utiliza el identificador único de entidades (ID) para crear una matriz que contiene los identificadores ID únicos para cada

elemento (matriz objcode2). Para cada elemento de esta matriz se realiza un proceso similar a la subrutina Editlink explicada anteriormente pero realizando la tarea de recuperar enlaces de la plantilla actual para todos los elementos seleccionadas (en vez de utilizarse para un único elemento). Cada enlace recuperado es cotejado con los ya obtenidos y si está repetido se desecha pero si es nuevo se añade a la matriz de valores diferentes. Finalmente se traspasan los valores almacenados en la matriz al fichero creado.

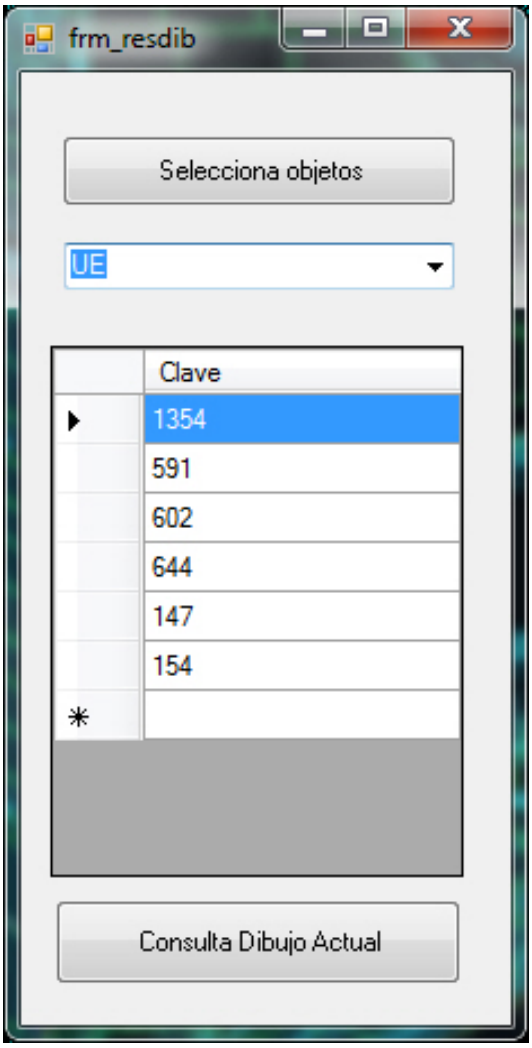
c) *Llenadatagrid*. Esta subrutina se encarga de leer el archivo de texto referido a la plantilla de vínculo actual generado anteriormente y de rellenar con esos valores el datagrid del formulario principal.

El programa empieza cuando el usuario pulsa el botón “selecciona objetos” o “Consulta dibujo actual”. En ambos casos el proceso seguido es similar y la única diferencia es que en la primera opción el usuario elige las entidades que desea entre todas las existentes en el dibujo y con la segunda el programa selecciona automáticamente todas las entidades del dibujo.

Una vez realizada la selección el programa comprueba que es válida y se crea una matriz con tantos elementos como entidades seleccionadas. En esta matriz se almacenan los códigos ID únicos para cada entidad asignados por el Autocad (esta matriz se utilizará en la subrutina *GrupoLink*) y posteriormente se accede a las plantillas de vínculo que existen en el dibujo.

Para cada plantilla de vínculo el programa ejecuta la *subrutina Grupolink* y en esta subrutina se crea un archivo de texto que se llama igual que la plantilla de vínculo y que contiene una lista de los diferentes valores asignados a la selección para esa plantilla.

El siguiente paso es actualizar el listado de valores del formulario y para ello el programa lee la plantilla de vinculo seleccionada (por defecto toma UC), accede al fichero creado para esa plantilla de vínculo e incorpora los valores de dicho fichero. Estos valores son refrescados cada vez que el usuario cambia el valor de la lista desplegable (que contiene las plantillas de vinculo con al menos un valor enlazado en los elementos seleccionados).



Formulario del comando RESDIB

Videotutorial: http://www.youtube.com/watch?v=FdAe0cylC8A

Comando datex.

Los datos extendidos son caracteres alfanuméricos que se añaden a cada entidad de dibujo y que permanecen como una característica más de esta entidad (como el color, capa,....). En este sentido recuerdan a los metadatos que se incluyen en las cabeceras de los archivos de fotografía y que permiten al usuario obtener información de la imagen.

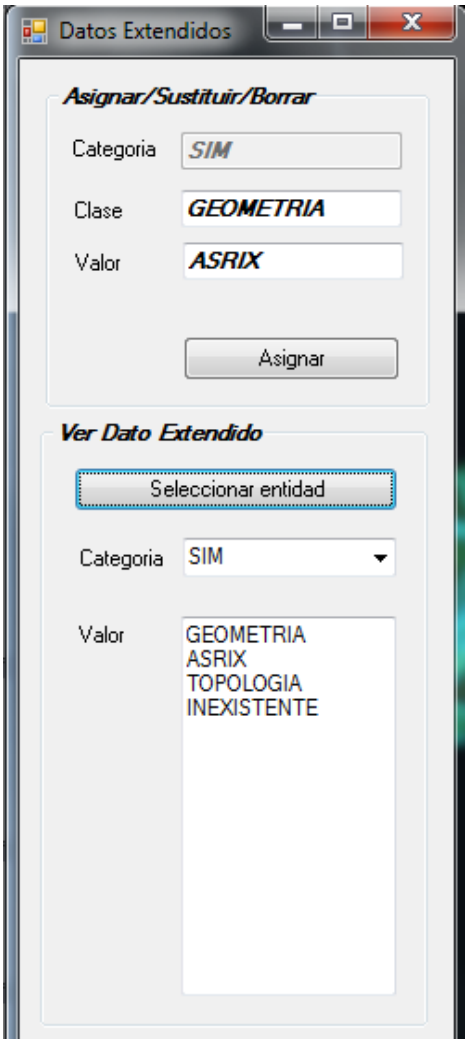
Para asignar un dato extendido a una entidad hay que definir una categoría (SIM, _JV_) asociada al valor numérico 1001 y una lista de valores asociados al valor numérico 1000. Cuando se realizó el modelo tridimensional de las fábricas de la catedral se utilizó la etiqueta _JV_ para añadir aspectos particulares a cada entidad. En esta nueva fase del SIM se plantea esta herramienta para que el usuario pueda añadir información de interés a cada entidad. Por ejemplo se podría añadir a cada polilínea el método utilizado para la adquisición de datos (p.e. topografia clásica, fotogrametría, rectificación con el programa ASRIX,…) de tal manera que cualquier usuario que copie esos elementos en otro dibujo pueda acceder de modo inmediato a este tipo de características. En el modelo de la catedral nos encontramos con información restituida por fotogrametría (95%) pero también entidades documentadas mediante topografía clásica (últimas campañas de excavaciones) o polilíneas creadas a partir de rectificación de fotos (para alzados inaccesibles en el proceso de documentación de pares fotogramétricos). Todos los métodos no tienen la misma precisión y no son discernibles a primera vista pero mediante la aplicación de datos extendidos podrían estar al alcance de cualquier usuario sin tener que analizar un informe específico sobre el tema.

El formulario del programa consta de dos partes:

+ En la parte superior el usuario puede indicar la clase y el valor del dato extendido que quiere asignar a un conjunto de entidades. Por defecto el programa asigna la categoría SIM. En función de los valores de las casillas de texto el programa realizará una tarea u otra:

  - Si la clase no está asignada a las entidades y el valor es distinto de “” el programa añade la clase y el valor indicados.
  - Si la clase ya está asignada pero el valor es diferente el programa sustituye el valor .
  - Si la clase ya está asignada y el valor es “” el programa borrará la clase y el valor que tenga asociado cada entidad.

De esta forma podremos añadir, modificar y eliminar datos extendidos. Conviene recordar que para una misma categoría (SIM en nuestro caso) podemos añadir varios elementos de texto (no se limita a un único valor) y en nuestro caso se añaden en parejas de tal forma que el primer valor define la clase de datos que incorporamos y el segundo su contenido concreto, p.e “GEOMETRIA , FOTOGRAMETRIA” ó “GEOMETRIA , TOPOGRAFIA CLASICA”.



Formulario del comando datex

+ En la parte inferior el programa nos pide seleccionar una entidad y acceder a los datos extendidos que tiene almacenados, indicando el valor de la categoría que deseemos. En nuestro caso aparece por defecto la categoría SIM y en el desplegable aparece también como predefinida la categoría "_JV_" utilizada en el desarrollo del modelo tridimensional realizado por fotogrametría. Una vez realizada la selección los valores son añadidos al cuadro de texto valor y son actualizados cada vez que el usuario cambia de categoría.

En lo referente al código, el programa muestra el formulario de inicio y si el usuario quiere editar los datos extendidos completará los campos de textos definidos para ello (clase y valor). Posteriormente pulsará el botón "Asignar" y el programa leerá el valor de las casillas de clase y valor e interpretará si el usuario quiere añadir/modificar (clase distinto de "" y valor distinto de "") o si quiere borrar datos (clase distinto de "" y valor igual a ""). El programa realiza la misma tarea con cada entidad, esto es, crea un nuevo tipo de datos extendidos para la categoría SIM (asignada por defecto) y va asignando los que ya tenía la entidad de tal forma:

- Si queremos sustituir un valor el programa localiza la clase definida y actualiza su valor con el contenido de la casilla de texto.
- Si queremos añadir un valor, éste se añadirá al final de la serie
- Si queremos borrar un elemento de la lista (clase y valor) se recorre toda la lista y se elimina el campo indicado y su valor correspondiente.

Finalmente se añade el nuevo dato extendido o se sustituye en caso de que exista uno anterior para la misma categoría. Si lo que se ha seleccionado es borrar algún dato concreto de la lista el proceso es similar ya que el nuevo dato extendido se ha generado sin el par de valores indicados.

En la parte que corresponde a mostrar los datos extendidos de una entidad, el código es relativamente sencillo ya que primero se selecciona la entidad y el programa consulta sus datos extendidos y los va añadiendo a la casilla de texto correspondiente. Además si el usuario actualiza el desplegable se repite la operación y se actualizan los valores mostrados. Para este programa los valores del desplegable se han limitado a _JV_ y a SIM que son las categorías predefinidas para los elementos del SIM.

Videotutorial: http://www.youtube.com/watch?v=r4lP7agPbZE

Comando Eti .

Este comando inserta un texto en el dibujo en el lugar y con el tamaño indicado por el usuario y el valor del texto lo toma de la información enlazada a la entidad seleccionada.

Las diferentes entidades del dibujo están enlazadas a varias plantillas de vínculo. Estos vínculos pueden ser UE (unidades estratigráficas), UC (unidades constructivas), MATERIALES, POLICRO (policromías). A su vez (y dado el carácter relacional de la base de datos) existe una gran cantidad de información asociada a estos vínculos. Por ejemplo una entidad estará asociada a una Unidad Estratigráfica (UE) concreta pero esta UE pertenecerá a una actividad concreta que a su vez pertenecerá a un grupo de actividades que se encuadrarán en una fase del monumento y se incluirán en un Periodo. Mediante esta herramienta el usuario puede incluir una etiqueta con el valor de la fase a la que pertenece la entidad del dibujo seleccionada. Para conocer este valor el programa debe conocer una serie de características técnicas de la base de datos diseñada (nombre de campo de consulta, nombre de la tabla,….). El programa diseñado permite ampliar y modificar estas características sin tocar ni una línea de código ya que se basa en la información que aparece en la base de datos SOPORTE.MDB ya que contiene la información base para este programa.

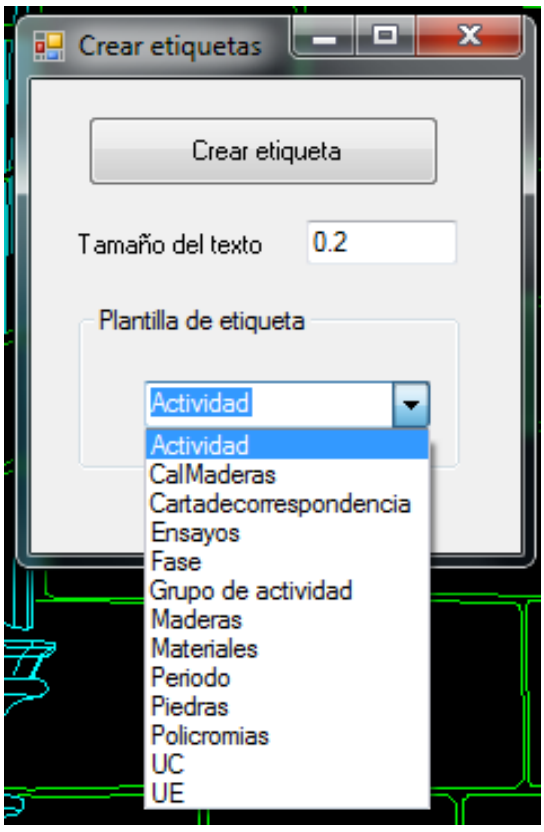
El usuario simplemente selecciona el tamaño del texto, el campo que quiere utilizar para la etiqueta y pulsará el botón “Crear etiqueta”. Posteriormente seleccionará la entidad consultada e indicará el punto de inserción del texto correspondiente. Este sencillo proceso pone en marcha un mecanismo más complejo que describe a continuación:

+ Se elige el valor de la etiqueta del desplegable. El desplegable muestra las diferentes temáticas que podemos elegir. Toma sus datos del campo *Indice* de la tabla *conIndice* de *Soporte.mdb*

+ Se selecciona la entidad del dibujo y si es correcta se ejecuta la *subrutina selekval* utilizando el valor del desplegable seleccionado

  - La *subrutina selekval* realiza una consulta sobre la tabla conIndice para ver qué plantilla de vinculo (campo cplantilla), en qué campo está la información de la etiqueta (campo ccampoclave) y el formato Texto/Numérico del campo clave (campo textonum)

  - Se obtiene el valor del enlace con la plantilla de vínculo que corresponde al campo seleccionado utilizando la *subrutina editlink* (similar a la descrita en el comando OBJINF)



Formulario del comando eti

- Se realiza una consulta sobre la tabla enlacesBD para ver en qué base de datos hay que consultar (cBaseDeDatos), en qué tabla/consulta (cTablaBase) y qué campo clave (cCampoClave) utiliza la etiqueta.

- Se extrae el valor del texto de la etiqueta con la información obtenida en el punto anterior.

+ Se ejecuta la *subrutina createxto* para el valor correspondiente
  - Se selecciona al punto de inserción
  - Se convierten las coordenadas del punto de inserción a coordenadas para el sistema de coordenadas universal
  - Se inserta el texto en la capa actual, con la orientación dada por el Sistema de Coordenadas Actual y con el estilo de texto actual.
  - Se inserta el texto en la capa actual, en el estilo de texto actual y con el tamaño indicado en el letrero de diálogo.

+ Se devuelven todas los valores de las variables a su estado inicial

+ Se vuelve a mostrar el formulario de inicio

Videotutorial: http://www.youtube.com/watch?v=hIXHArU2N54

Comando vermapa.

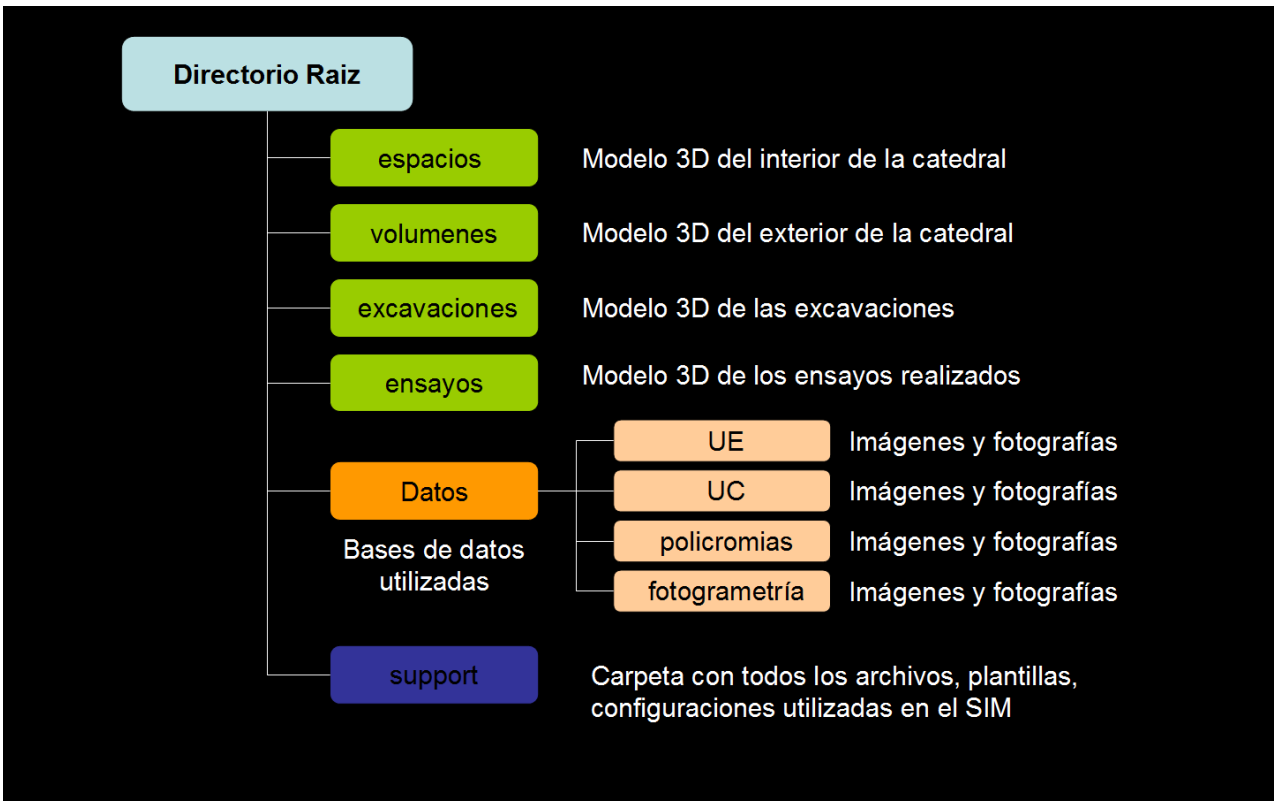
Mediante este comando se muestra un formulario con un acceso directo a todos los comandos programados para el manejo del Sistema de Información Monumental de la Catedral de Santa María de Vitoria-Gasteiz.

Formulario del comando vermapa

4.7 Instalación del programa SIM.

Todo el SIM está estructurado en un conjunto organizado de carpetas y subcarpetas, por lo tanto si conocemos dónde está instalada la carpeta inicial podremos conocer el camino concreto para cada elemento.

Esquemáticamente la estructura de ficheros carpetas del SIM es la siguiente:



Actualmente toda esta estructura de datos ocupa algo más de 35 Gb y más de 50.000 ficheros con diversa documentación. Esta gran cantidad de información desaconseja crear un archivo de instalación “todo en uno”.

Es por ello que el proceso de instalación se plantea tres pasos definido:

a) Copiar los archivos correspondientes en la carpeta de instalación indicada,

b) Modificar parámetros de la aplicación de Autodesk Map. Los formularios y comandos desarrollados en este proyecto utilizan recursos (dibujos, plantillas, fotografías, bases de datos,..) que se han cargado en el primer paso. Autocad permite la personalización de una serie de preferencias que utilizarán posteriormente los programas realizados para su correcto funcionamiento. Para realizar esta actualización de parámetros se deberá cargar el archivo SIM.DLL (subcarpeta Support) desde el comando NETLOAD. Una vez cargado se ejecutará el comando “**instalaSIM**”. Este programa solicita al usuario que seleccione el archivo *config.path* (situado en el directorio raíz del SIM) y una vez seleccionado este archivo calcula los nuevos parámetros, los sustituye en la aplicación y guarda los originales en el archivo config.path. Mediante la ejecución del comando “**quitaSIM**” este proceso se puede revertir y volver a la configuración original.

El comando “instalaSIM” realizará las siguientes tareas:

+ Actualización de los valores para el archivo de soporte, camino para plantillas, localización del fichero de autoguardado y el camino para buscar los orígenes de datos.

+ Actualización de los alias definidos para enlazar los archivos de consulta en el Autocad Map. Estos alias se utilizan en los comandos ASOEXT, ASOINT y ASOEXC.

+ Cargar el menú SIM.CUIX. Como ya se ha descrito anteriormente toda la colección de comandos desarrollados tiene una barra de herramientas específica que se guarda como un archivo de configuración parcial. Mediante el comando InstalaSIM este menú es cargado en el dibujo para que aparezca la barra de herramientas correspondiente. A continuación se puede ver la barra de herramientas diseñada para el manejo de los comandos desarrollados en este proyecto.

VerMapa

Asoexc

Asoext

Asoint

Consim

Conmul

Contem

Objinf

ResDib

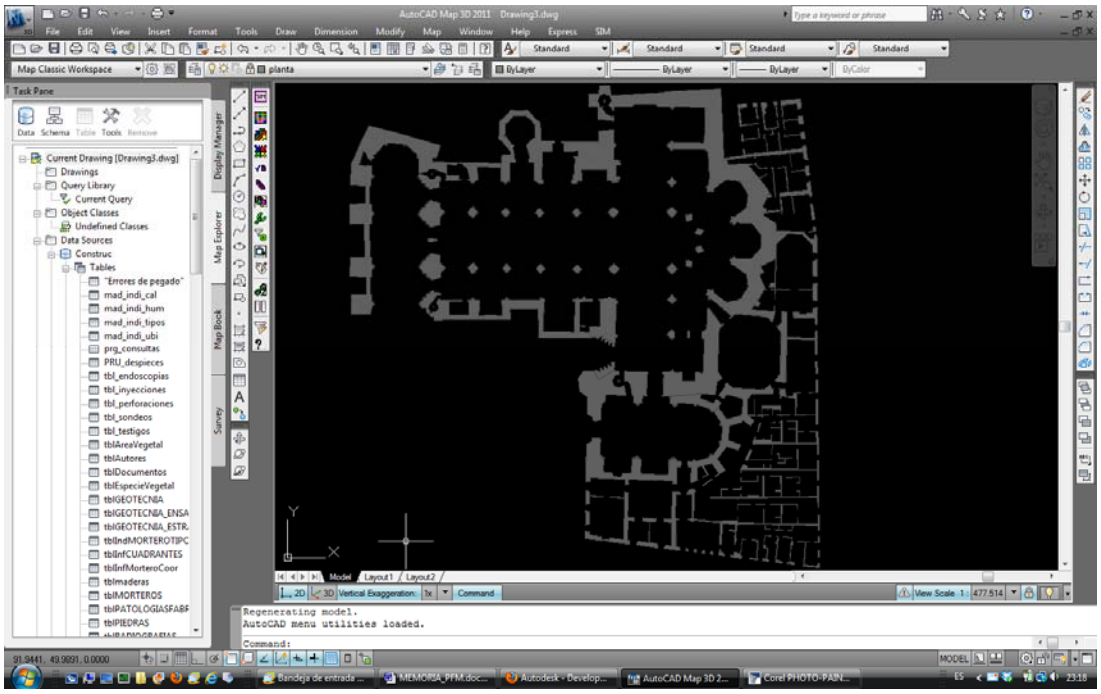
Datex

Eti

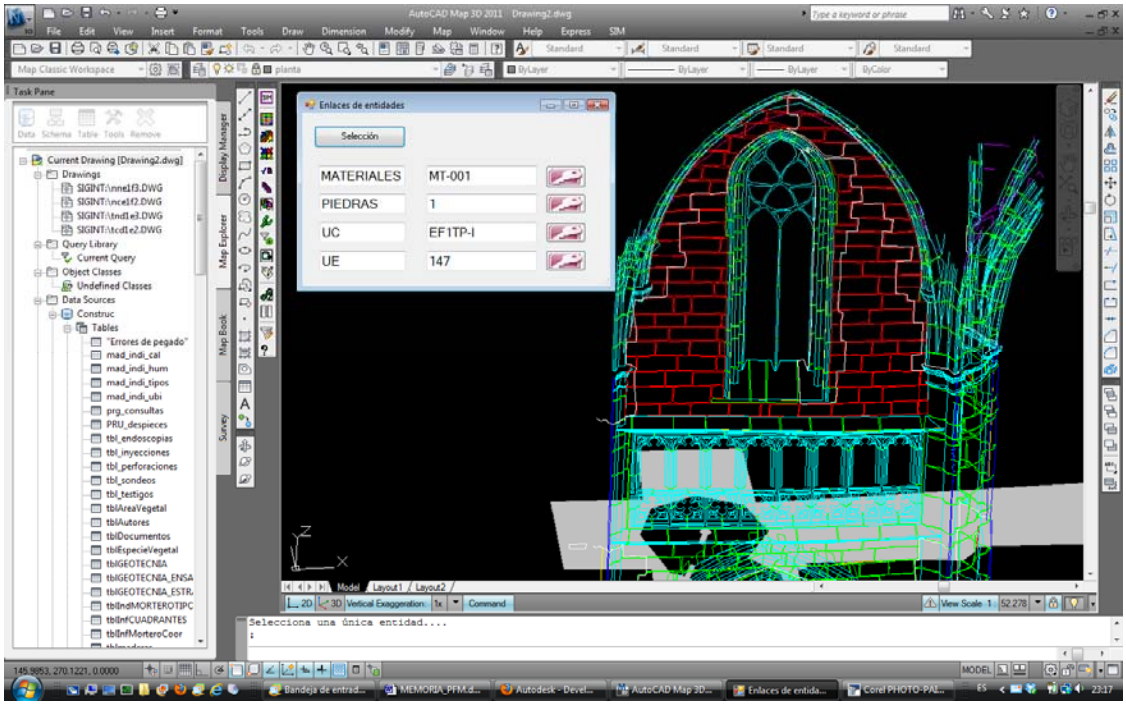
+ Creación de los archivos de vinculo de datos UDL. La plantilla principal de la aplicación se diferencia de la que ofrece Autocad por defecto en que existe una planta de la catedral y están asociados a su origen de datos las bases de datos de HISTORIA, CONSTRUC y POLICROMÍAS. En el proceso normal el usuario debe asociar el origen de datos seleccionando el fichero MDB correspondiente y el sistema crea automáticamente un archivo con extensión UDL que es el que utiliza Autocad. Para ahorrar al usuario tener que asociar 3 bases de datos el programa configura directamente los archivos UDL

c) Añadir entradas al registro para que el programa de autocad cargue automáticamente las librerías dll creadas. Para ello hay que acceder al registro y añadir la siguiente entrada:

[HKEY_CURRENT_USER\Software\Autodesk\AutoCAD\R18.1\ACAD-9002:40A\Applications\SIMNET]
"DESCRIPTION"="Carga el archivo SIM.DLL"
"LOADCTRLS"=dword:00000002
"LOADER"="E:\\DIRECTORIO DE INSTALACION\\support\\SIM.dll"
"MANAGED"=dword:00000001



Nuevo dibujo de autocad basado en la plantilla del SIM.



Pantalla del programa Autodesk Map mientras se consulta el SIM. En la imagen se puede observar los archivos asociados, la barra de herramientas del SIM, el formulario del comando objinf y parte del modelo tridimensional de la Catedral.

5.- Conclusiones

La utilización del lenguaje de programación de VB.NET en entornos de Autocad proporciona una serie de soluciones personalizadas con un enorme potencial. En el Sistema de Información Monumental (SIM) de la Catedral de Santa María ha permitido simplificar y hacer accesibles las tareas más habituales en la gestión del SIM, en este sentido el proyecto tiene una aplicación directa con la realidad ya que parte de una necesidad concreta en el ámbito de la geomática y aporta una solución aplicada.

Durante el proyecto se han utilizado especialmente los conocimientos correspondiente a la asignatura Herramientas Informáticas para el Geoprocesado. Sin embargo no se ha limitando a los contenidos de esta asignatura ya que se han reforzado mediante un curso en programación en *Microsoft Visual Basic .NET* y *ADO.NET* y se ha profundizado en su uso específico para Autodesk Map utilizando diferentes recursos disponibles en internet.

Se ha comprobado que el lenguaje VB.NET puede trabajar con todos los datos que forman parte del SIM, para ello se han desarrollado programas para realizar aplicaciones Windows, se ha accedido a bases de datos mediante la tecnología ADO.NET y se han generado comandos que realizan determinadas tareas dentro de las aplicaciones de Autocad y Autodesk Map.

El resultado del proyecto es una metodología que permite a los diferentes profesionales que trabajan en la restauración de la Catedral de Santa María utilizar y gestionar toda la información registrada en el SIM. El usuario deberá tener nociones a nivel de usuario tanto del manejo de dibujos tridimensionales como de los conceptos asociados a los Sistemas de Información Geográfica para poder utilizar los diferentes comandos programados. Sobre esta base una lectura explicativa de los cometidos de los diferentes comandos será suficiente para empezar a gestionar el sistema. Además la disponibilidad de videotutoriales para cada herramienta desarrolladas para el SIM permite al usuario conocer el funcionamiento del comando sobre un ejemplo real y en un entorno configurado correctamente, resolviendo muchas de las dudas que persisten tras leer una descripción de las tareas que realiza el comando en cuestión.

El proyecto realizado simplifica el trabajo de gestión del SIM y además de conseguir una mayor eficiencia en el análisis, permite utilizar herramientas de gestión a usuarios no avanzados en Autodesk Map.

No existe gran cantidad de información en internet referida a la utilización de VB.NET en el entorno de Autodesk Map y mucho menos para dibujos en tres dimensiones. Sin embargo la información disponible ha sido de vital importancia en el desarrollo de este proyecto y en consecuencia todo el código aportado en este proyecto se puede reproducir con la condición de que el proyecto resultante se ponga a disposición de la comunidad de desarrolladores.

Una línea de investigación interesante consiste en el desarrollo de las posibilidades que ofrece el llamado *Dato Extendido,* ya que al ser un campo definido por el usuario que se añade a la descripción de la entidad geométrica podría ser utilizado para relacionar la entidad gráfica y la base de datos. De esta forma se podrían programar una serie de comandos que permitirían relacionar bases de datos y entidades de Autocad sin tener que recurrir al programa Autodesk Map (más caro y menos habitual).

6.- Bibliografía consultada

Libros sobre programación:

- VB.NET for AutoCAD 2010, Jerry Winters publicado por VB CAD.
- AutoCAD Database Connectivity, Scott McFarlane publicado por Thomson Learning 2000

Información de Autodesk:

- ObjectARX for Autodesk Map 3D 2011

http://usa.autodesk.com/adsk/servlet/item?siteID=123112&id=868205

- ObjectARX for AutoCAD 2011

http://usa.autodesk.com/adsk/servlet/item?siteID=123112&id=785550

ObjectARX (AutoCAD Runtime Extension) es un entorno de programación de lenguaje compilado para desarrollar aplicaciones de AutoCAD. Se puede acceder a este entorno mediante C++, C#, VB.NET o cualquier otro lenguaje .NET. También incluyen una guia de desarrollador, información de referencia y código de ejemplo para los lenguajes de programación soportados.

Las bibliotecas de ObjectARX permiten aprovechar la arquitectura abierta de AutoCAD y Autodesk Map, proporcionando un acceso directo a las estructuras de bases de datos de AutoCAD, al sistema de gráficos, y al motor de geometría de AutoCAD para ampliar las clases y capacidades de AutoCAD en tiempo de ejecución. Además, puede definir comandos nuevos que funcionen exactamente igual que los comandos originales de AutoCAD.

El entorno de programación ObjectARX se describe en ObjectARX Developer's Guide. La documentación forma parte de ObjectARX Software Development Kit (SDK), que se puede descargar de la sección Development Tools del sitio Web de Autodesk.

Foros y blogs en la red

- Autodesk Discussion Groups > AutoCAD > AutoCAD Customization > .NET (http://forums.autodesk.com/t5/NET/bd-p/152)
- Autodesk Discussion Groups > AutoCAD > AutoCAD Customization >Autodesk Object ARX. http://forums.autodesk.com/t5/Autodesk-ObjectARX/bd-p/34
- Autodesk Discussion Groups > AutoCAD Map 3D > AutoCAD Map 3D Developer http://forums.autodesk.com/t5/AutoCAD-Map-3D-Developer/bd-p/84
- The Swamp - TheSwamp is a friendly community of programmers, designers and drafters who share ideas, resources and information in a friendly, helpful context. http://www.theswamp.org/index.php?board=27.0
- Through the interface, blog moderado por Kean Walmsley http://through-the-interface.typepad.com/through_the_interface/
- Map 3d and Murphs Law http://map3d.wordpress.com/

7.- Anexos

7.1 Certificado de realización de curso de Microsoft Visual Basic .NET.

Ayuntamiento de Vitoria-Gasteiz
Vitoria-Gasteizko Udala
www.vitoria-gasteiz.org

E-LEARNING PRESTAKUNTZA - FORMACIÓN E-LEARNING

Luis Ortiz Tudanca,
Prestakuntza Atalaren buruak

ADIERAZTEN DU

Jarraian aipatzen den pertsonak **"Microsoft Visual Basic.NET"** izeneko programan parte hartu duela eta gaitasun maila lortu duela.

Luis Ortiz Tudanca,
Jefe de la Unidad de Formación

ACREDITA

Que la persona que se cita a continuación ha participado en el programa denominado **"Microsoft Visual Basic.NET"** habiendo alcanzado la cualificación prevista.

Ikaslearen izen-deiturak Nombre y apellidos del alumno/a	**IÑAKI KOROSO ARRIAGA**
NAN DNI	**18595621-Y**
Iraupena Duración	**74 ordu / horas**
Hasiera-bukaera datak Fechas inicio-fin	**20/01/2011 - 21/04/2011**
Lekua Lugar	**CETIC, Castro Urdiales, 10. Vitoria-Gasteiz**
Antolatzailea Organizado por	**Vitoria-Gasteizko Udala**, Ekonomia Sustapen eta Estrategia Plangintzaren Saileko Prestakuntza eta Enplegu Sustapenerako Zerbitzuaren bitartez **Ayuntamiento de Vitoria-Gasteiz**, a través del Servicio de Formación-Promoción Empleo del Departamento de Promoción Económica y Planificación Estrategica

Vitoria-Gasteiz, 5 de mayo de 2011

Izp./Fdo.: Luis Ortiz Tudanca
Prestakuntza Atalaren buruak
Jefe de la Unidad de Formación

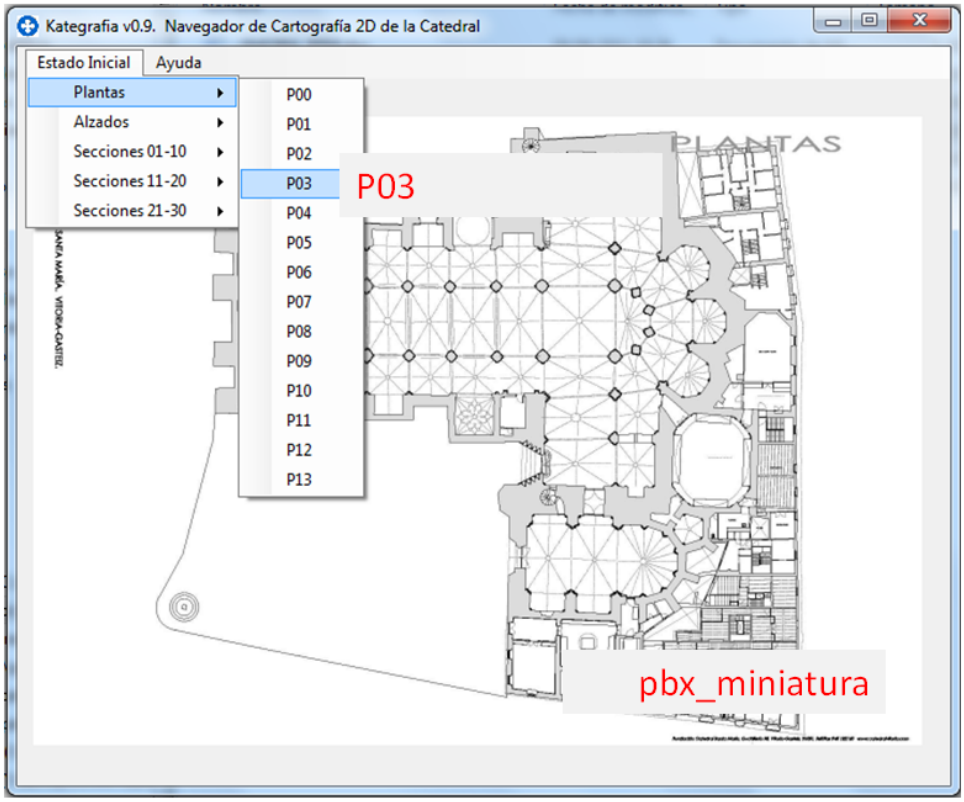
Ekonomia Sustapen eta Estrategia Plangintzaren Saila
Departamento de Promoción Económica y Planificación Estratégica

Prestakuntza eta Enplegu Sustapenerako Zerbitzua
Servicio de Formación-Promoción de Empleo

Castro Urdiales, 10
01006 Vitoria-Gasteiz
Tel.: 945 16 15 05
Fax: 945 16 15 04
promocioneconomica@vitoria-gasteiz.org
www.vitoria-gasteiz.org

7.2 Código fuente comentado del programa Kategrafia

SIM_2D.vb




```
Public Class SIM_2D

    Public Function App_Path() As String
        Return System.AppDomain.CurrentDomain.BaseDirectory()
    End Function
    Dim actual As String
    Dim caminoacad As String


    Private Sub pbxMiniatura_MouseDown(ByVal sender As Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles pbxMiniatura.MouseDown
‘ Eventos a realizar al pulsar el puntero sobre la imagen de fondo

        If e.Button = Windows.Forms.MouseButtons.Right Then
            Dim proceso As Process = New Process()
            ‘ Si se pulsa el botón derecho se abre una instancia de Adobe
      Acrobat o de Acrobat Reader
                Dim camino, fichero As String
            camino = App_Path()
            Dim plano As String
            plano = sender.ToString
            fichero = actual & ".pdf"
            Try
                proceso.StartInfo.FileName = "acrobat.exe"
                proceso.StartInfo.Arguments = fichero
                proceso.Start()
            Catch
                proceso.StartInfo.FileName = "AcroRd32.exe"
                proceso.StartInfo.Arguments = fichero
                proceso.Start()
```

```
            End Try
        End If
        If e.Button = Windows.Forms.MouseButtons.Left Then
            Dim proceso As Process = New Process()
            Dim camino, fichero As String
            ‘ Si se pulsa el botón izquierdo se abre una instancia del
      explorador situado en la carpeta donde se aloja la información mostrada
      en la miniatura

            camino = App_Path()
            Dim plano As String
            plano = sender.ToString

            Try
                Dim caracter As String
                Dim posi, lona As Single
                fichero = actual
                lona = Len(fichero)
                For x = lona To 1 Step -1
                    caracter = Mid(fichero, x, 1)
                    If caracter = "\" Then posi = x : Exit For
                Next x
                fichero = Mid(fichero, 1, posi)
                proceso.StartInfo.FileName = "explorer.exe"
                proceso.StartInfo.Arguments = fichero
            Catch
                Dim caracter As String
                Dim posi, lona As Single
                fichero = actual
                lona = Len(fichero)
                For x = lona To 1 Step -1
                    caracter = Mid(fichero, x, 1)
                    If caracter = "\" Then posi = x : Exit For
                Next x
                fichero = Mid(fichero, 1, posi)
                proceso.StartInfo.FileName = "explorer.exe"
                proceso.StartInfo.Arguments = fichero
            End Try
            proceso.Start()
        End If
    End Sub


    Private Sub P00_MouseDown(ByVal sender As Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles P00.MouseDown
‘ Eventos a realizar al pasar el puntero sobre los elementos del menu

        If e.Button = Windows.Forms.MouseButtons.Right Then
            ‘ Si se pulsa el botón derecho se abre una instancia de Adobe
      Acrobat o de Acrobat Reader

            Dim proceso As Process = New Process()
            Dim camino, fichero As String
            Dim plano As String
            plano = sender.ToString
            camino = App_Path()
            fichero = camino & "2D\01PLANTA\" & plano & "\" & plano & ".pdf"
            Try
                proceso.StartInfo.FileName = "acrobat.exe"
                proceso.StartInfo.Arguments = fichero
                proceso.Start()
            Catch
                proceso.StartInfo.FileName = "AcroRd32.exe"
                proceso.StartInfo.Arguments = fichero
```

```
                proceso.Start()
            End Try
        End If
        If e.Button = Windows.Forms.MouseButtons.Left Then
            ‘ Si se pulsa el botón izquierdo se abre una instancia del
      explorador situado en la carpeta donde se aloja la información mostrada
      en la miniatura

            Dim proceso As Process = New Process()
            Dim camino, fichero As String
            Dim plano As String
            plano = sender.ToString
            camino = App_Path()
            Try
                    fichero = camino & "2D\01PLANTA\" & plano
                proceso.StartInfo.FileName = "explorer.exe"
                proceso.StartInfo.Arguments = fichero
                proceso.Start()
            Catch
                MsgBox("Problema al abrir el explorer...")
            End Try
        End If
    End Sub

      .......
similar a lo anterior para las 13 Planos de Planta
      .......


    Private Sub A01_MouseDown(ByVal sender As Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles A01.MouseDown
        If e.Button = Windows.Forms.MouseButtons.Right Then
            Dim proceso As Process = New Process()
            Dim camino, fichero As String

            camino = App_Path()
            Dim plano As String
            plano = sender.ToString
            fichero = camino & "2D\02ALZADO\" & plano & "\" & plano & ".pdf"
            Try
                proceso.StartInfo.FileName = "acrobat.exe"
                proceso.StartInfo.Arguments = fichero
                proceso.Start()
            Catch
                proceso.StartInfo.FileName = "AcroRd32.exe"
                proceso.StartInfo.Arguments = fichero
                proceso.Start()
            End Try
        End If
        If e.Button = Windows.Forms.MouseButtons.Left Then
            Dim proceso As Process = New Process()
            Dim camino, fichero As String
            proceso.StartInfo.FileName = "explorer.exe"
            camino = App_Path()
            Dim plano As String
            plano = sender.ToString
            fichero = camino & "2D\02ALZADO\" & plano
            proceso.StartInfo.Arguments = fichero
            proceso.Start()
        End If
    End Sub

      .......
similar a lo anterior para las 4 Planos de Alzado
```

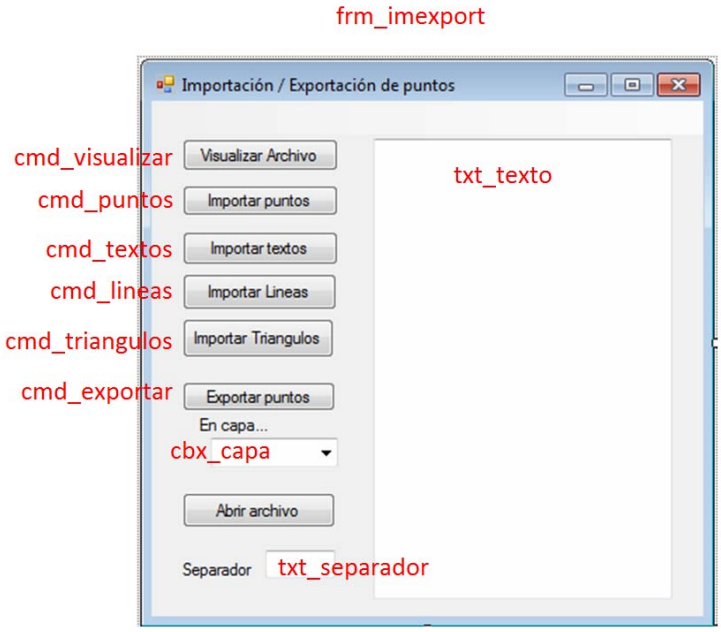
```
      .......


     Private Sub S01_MouseDown(ByVal sender As Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles S01.MouseDown
        If e.Button = Windows.Forms.MouseButtons.Right Then
            Dim proceso As Process = New Process()
            Dim camino, fichero As String
            camino = App_Path()
            Dim plano As String
            plano = sender.ToString
            fichero = camino & "2D\03SECCION\" & plano & "\" & plano & ".pdf"
            Try
                proceso.StartInfo.FileName = "acrobat.exe"
                proceso.StartInfo.Arguments = fichero
                proceso.Start()
            Catch
                proceso.StartInfo.FileName = "AcroRd32.exe"
                proceso.StartInfo.Arguments = fichero
                proceso.Start()
            End Try
        End If
        If e.Button = Windows.Forms.MouseButtons.Left Then
            Dim proceso As Process = New Process()
            Dim camino, fichero As String
            proceso.StartInfo.FileName = "explorer.exe"
            camino = App_Path()
            Dim plano As String
            plano = sender.ToString
            fichero = camino & "2D\03SECCION\" & plano
            proceso.StartInfo.Arguments = fichero
            proceso.Start()
        End If
    End Sub

      .......
similar a lo anterior para las 33 Planos de Sección
      .......



    Private Sub P00_MouseMove(ByVal sender As Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles P00.MouseMove
‘ Eventos a realizar al pasar el puntero sobre los elementos del menu

        Dim camino, fichero As String
        camino = App_Path()
        Dim plano As String
        plano = sender.ToString
        fichero = camino & "2D\01PLANTA\" & plano & "\" & plano & ".jpg"
        pbxMiniatura.Image = System.Drawing.Bitmap.FromFile(fichero)
        actual = Mid(fichero, 1, Len(fichero) - 4)
    End Sub
    Private Sub P01_MouseMove(ByVal sender As Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles P01.MouseMove
‘ Actualiza la imagen mostrada en el fondo para que coincida con la mostrada en
el nombre del menu
        Dim camino, fichero As String
        camino = App_Path()
        Dim plano As String
        plano = sender.ToString
        fichero = camino & "2D\01PLANTA\" & plano & "\" & plano & ".jpg"
        pbxMiniatura.Image = System.Drawing.Bitmap.FromFile(fichero)
        actual = Mid(fichero, 1, Len(fichero) - 4)
```

```
    End Sub

      .......
similar a lo anterior para las 13 Planos de Planta
      .......

    Private Sub A01_MouseMove(ByVal sender As Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles A01.MouseMove
        Dim camino, fichero As String
        camino = App_Path()
        Dim plano As String
        plano = sender.ToString
        fichero = camino & "2D\02ALZADO\" & plano & "\" & plano & ".jpg"
        pbxMiniatura.Image = System.Drawing.Bitmap.FromFile(fichero)
        actual = Mid(fichero, 1, Len(fichero) - 4)
    End Sub

      .......
similar a lo anterior para las 4 Planos de Alzado
      .......


    Private Sub S01_MouseMove(ByVal sender As Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles S01.MouseMove
        Dim camino, fichero As String
        camino = App_Path()
        Dim plano As String
        plano = sender.ToString
        fichero = camino & "2D\03SECCION\" & plano & "\" & plano & ".jpg"
        pbxMiniatura.Image = System.Drawing.Bitmap.FromFile(fichero)
        actual = Mid(fichero, 1, Len(fichero) - 4)
    End Sub
      .......
similar a lo anterior para las 33 Planos de Sección
      .......


    Private Sub ToolStripMenuItem_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles ÇaToolStripMenuItem.Click
        Dim frmlaguntza As New frm_ayuda
        frmlaguntza.Show()
    End Sub



End Class
```

7.3 Código fuente comentado del comando DesactivaCapa

```
    <CommandMethod("DesactivaCapa")> _
    Public Sub DesactivaCapa()

        'Definición y asignación de variables
        Dim mydwg As ApplicationServices.Document
        Dim mydb As DatabaseServices.Database
        Dim mytransman As DatabaseServices.TransactionManager
        Dim mytrans As DatabaseServices.Transaction

        mydwg =
ApplicationServices.Application.DocumentManager.MdiActiveDocument
        mydb = mydwg.Database
        mytransman = mydwg.TransactionManager
        mytrans = mytransman.StartTransaction

        'Definición de variables relacionadas con la biblioteca de capas
        Dim myLT As DatabaseServices.LayerTable
        Dim mylayer As DatabaseServices.LayerTableRecord
        Dim myste As DatabaseServices.SymbolTableEnumerator

        'Creación de una selección
        Dim myEd As Editor = DocumentManager.MdiActiveDocument.Editor
        Dim myPSR As PromptSelectionResult = myEd.GetSelection

        ' Si la selección es correcta se recorren todos sus elementos
        ' leyendo la capa en la que están y se analiza de tal forma
        ' que si es la capa actual o está congelada no se hace nada
        ' y si está activada se congela.
        Dim nomcapa As String
        mylayer = mydwg.Database.LayerZero.GetObject(OpenMode.ForRead)
'initializes the layer
        myLT = mydb.LayerTableId.GetObject(DatabaseServices.OpenMode.ForRead)
        myste = myLT.GetEnumerator
        If myPSR.Status = PromptStatus.OK Then
            If IsNothing(myPSR.Value) = False Then
                For Each myObjID As ObjectId In myPSR.Value.GetObjectIds
                    myste.Reset()
                    Dim myAcadEnt As Entity =
myObjID.GetObject(OpenMode.ForRead)
                    nomcapa = myAcadEnt.Layer
                    While myste.MoveNext
                        mylayer = myste.Current.GetObject(OpenMode.ForWrite)
                        If nomcapa = mylayer.Name Then
                            If Not mylayer.IsFrozen Then
                                If Not mylayer.Id = mydb.Clayer Then
                                    mylayer.IsFrozen = True
                                    Exit While
                                End If
                            End If
                        End If
                    End While
                Next
            End If
        End If
        mytrans.Commit()

    End Sub
```

7.4 Código fuente comentado del comando Imexport




```
Imports Autodesk.AutoCAD.Runtime
Imports Autodesk.AutoCAD
Imports Autodesk.AutoCAD.DatabaseServices
Imports Autodesk.AutoCAD.EditorInput
Imports Autodesk.AutoCAD.Geometry
Imports Autodesk.AutoCAD.ApplicationServices
Imports Autodesk.AutoCAD.ApplicationServices.Application
Imports Autodesk.AutoCAD.LayerManager
Imports Autodesk.AutoCAD.Windows
Imports Autodesk.AutoCAD.LayerManager.LayerCollection
Imports Autodesk.AutoCAD.DatabaseServices.SymbolTableRecord
Public Class frm_imexport
    Dim nombrefichero As String


    Private Sub cmd_visualizar_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cmd_visualizar.Click
        OpenFileDialog1.Filter = "Archivos de Texto(*.txt)|*.txt|Todos los
Archivos|*.*"

        'Le indicamos los filtros que aparecerán al seleccionar los archivos.
asc, txt, o todos.
        OpenFileDialog1.ShowDialog() 'Mostramos diálogo de Abrir
        nombrefichero = OpenFileDialog1.FileName 'Colocamos en el cuadro de
texto la ruta y nombre del fichero seleccionado
        Dim sr As New IO.StreamReader(nombrefichero)
        Dim linea As String
        Dim contenido As String
        Dim ind As Integer
        contenido = ""
        For ind = 1 To 20
            linea = sr.ReadLine()
```

```
            If Not linea Is Nothing Then
                If linea <> "" Then
                    contenido = (contenido & linea & vbCrLf) 'Añado la linea
leída al visor. vbCrLf es un salto de linea
                End If
            Else
                Exit For
            End If
        Next ind
        MsgBox(contenido)
        txt_texto.Text = contenido
        sr.Close()
        sr = Nothing
    End Sub

    Private Sub cmd_puntos_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cmd_puntos.Click
        Dim myDWG As ApplicationServices.Document
        Dim myDB As DatabaseServices.Database
        Dim myEd As EditorInput.Editor
        Dim mytransman As DatabaseServices.TransactionManager
        Dim mytrans As DatabaseServices.Transaction
        Dim myBT As DatabaseServices.BlockTable
        Dim myBTR As DatabaseServices.BlockTableRecord

        myDWG =
ApplicationServices.Application.DocumentManager.MdiActiveDocument
        myDWG.LockDocument(DocumentLockMode.AutoWrite, Nothing, Nothing, True)
        myDB = myDWG.Database
        myEd = myDWG.Editor
        mytransman = myDWG.TransactionManager
        mytrans = mytransman.StartTransaction
        myBT =
myDWG.Database.BlockTableId.GetObject(DatabaseServices.OpenMode.ForRead)
        myBTR =
myBT(DatabaseServices.BlockTableRecord.ModelSpace).GetObject(DatabaseServices.O
penMode.ForWrite)

        Dim x, y, z As Double

        Dim nombrefichero1 As String
        OpenFileDialog1.Filter = "Archivos de Texto(*.txt)|*.txt|Todos los
Archivos|*.*"
        OpenFileDialog1.ShowDialog() 'Mostramos diálogo de Abrir
        nombrefichero1 = OpenFileDialog1.FileName 'Colocamos en el cuadro de
texto la ruta y nombre del fichero seleccionado
        Dim sr As New IO.StreamReader(nombrefichero1)
        Dim linea As String
        Dim contenido As String
        Dim separador As String
        Dim capa As String
        capa = ""
        contenido = ""
        txt_separador.Text = vbTab
        separador = txt_separador.Text
        If separador = Nothing Then separador = " "
        Do
            linea = sr.ReadLine()
            If Not linea Is Nothing Then
                If linea <> "" Then
                    Dim ptoArray() As String = Split(linea, separador)
                    If ptoArray.Length = 3 Then
                        x = ptoArray(0)
                        y = ptoArray(1)
```

```
                        z = ptoArray(2)
                    End If
                    If ptoArray.Length = 4 Or ptoArray.Length = 5 Then
                        x = ptoArray(0)
                        y = ptoArray(1)
                        z = ptoArray(2)
                        capa = ptoArray(3)
                    End If
                    Dim punto As New Geometry.Point3d(x, y, z)
                    Dim punto2 As New DBPoint(punto)
                    myBTR.AppendEntity(punto2)
                    Try
                        If ptoArray.Length = 4 Or ptoArray.Length = 5 Then
punto2.Layer = capa
                    Catch
                        MsgBox("No existe la capa " & capa & vbCr & "El punto
se importará en la capa actual")
                    End Try
                    mytrans.AddNewlyCreatedDBObject(punto2, True)

                End If
            End If
        Loop Until linea Is Nothing
        sr.Close()
        sr = Nothing

        myTrans.Commit()
        myTrans.Dispose()
        myTransMan.Dispose()

        MsgBox("Proceso finalizado...")
    End Sub

    Private Sub cmd_exportar_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cmd_exportar.Click
        Dim myDWG As ApplicationServices.Document
        Dim myDB As DatabaseServices.Database
        Dim myEd As EditorInput.Editor
        Dim mytransman As DatabaseServices.TransactionManager
        Dim mytrans As DatabaseServices.Transaction
        Dim cam As String

        Dim myfilter1(0) As DatabaseServices.TypedValue
        Dim myfilter2(1) As DatabaseServices.TypedValue
        If cbx_capa.Text = "--Todas--" Then
            myfilter1(0) = New
DatabaseServices.TypedValue(DatabaseServices.DxfCode.Start, "POINT")
        Else
            myfilter2(0) = New
DatabaseServices.TypedValue(DatabaseServices.DxfCode.Start, "POINT")
            myfilter2(1) = New
DatabaseServices.TypedValue(DatabaseServices.DxfCode.LayerName, cbx_capa.Text)
        End If
        Dim mysf1 As New EditorInput.SelectionFilter(myfilter1)
        Dim mysf2 As New EditorInput.SelectionFilter(myfilter2)

        Dim mypsr As EditorInput.PromptSelectionResult
        Dim mySS As EditorInput.SelectionSet
        Dim myobjids As DatabaseServices.ObjectIdCollection
        Dim myent As DatabaseServices.Entity
        myDWG =
ApplicationServices.Application.DocumentManager.MdiActiveDocument
        myDB = myDWG.Database
        myEd = myDWG.Editor
```

```
        'Seleccionar archivo a crear
        SaveFileDialog1.Filter = "Archivos de Texto(*.txt)|*.txt|Todos los
Archivos|*.*"
        SaveFileDialog1.ShowDialog() 'Mostramos diálogo de Abrir
        cam = SaveFileDialog1.FileName 'Colocamos en el cuadro de texto la ruta
y nombre del fichero seleccionado

        Dim myfsw As New IO.StreamWriter(cam)
        Dim separador As String
        separador = txt_separador.Text
        If separador = Nothing Then separador = vbTab
        'Seleccionar puntos
        Dim i As Long
        Dim xx, yy, zz As Double
        Dim capa As String
        capa = ""
        If cbx_capa.Text = "--Todas--" Then
            mypsr = myEd.SelectAll(mysf1)
        Else
            mypsr = myEd.SelectAll(mysf2)
        End If
        mytransman = myDWG.TransactionManager
        mytrans = mytransman.StartTransaction
        If Not IsNothing(mypsr.Value) Then
            mySS = mypsr.Value
            MsgBox(mySS.Count)
            myobjids = New
DatabaseServices.ObjectIdCollection(mySS.GetObjectIds)
            For i = 1 To mySS.Count
                myent = mySS(i -
1).ObjectId.GetObject(DatabaseServices.OpenMode.ForRead)
                xx = Math.Round(CDbl(myent.GeometricExtents.MaxPoint.X), 3)
                yy = Math.Round(CDbl(myent.GeometricExtents.MaxPoint.Y), 3)
                zz = Math.Round(CDbl(myent.GeometricExtents.MaxPoint.Z), 3)
                capa = myent.Layer
                myfsw.WriteLine(xx & separador & yy & separador & zz &
separador & capa & vbCr)
            Next i
        End If
        mytrans.Dispose()
        mytransman.Dispose()

        'Cerrar ficheros
        myfsw.Close()
        myfsw.Dispose()

        MsgBox("Puntos exportados en ..." & cam)
    End Sub

    Private Sub cmd_Abrir_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cmd_abrir.Click
        Dim proceso As Process = New Process()

        'Indicamos el Programa
        proceso.StartInfo.FileName = "Notepad.exe"

        'Indicamos la ruta del archivo
        OpenFileDialog2.Filter = "Archivos de Texto(*.txt)|*.txt|Todos los
Archivos|*.*"
        OpenFileDialog2.ShowDialog() 'Mostramos diálogo de Abrir
        nombrefichero = OpenFileDialog2.FileName 'Colocamos en el cuadro de
texto la ruta y nombre del fichero seleccionad
        proceso.StartInfo.Arguments = nombrefichero
```

```
        'Ejecutamos el proceso
        proceso.Start()
    End Sub

    Private Sub frm_imexport_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        Dim myDWG As ApplicationServices.Document
        Dim myDB As DatabaseServices.Database
        Dim myEd As EditorInput.Editor
        Dim mytransman As DatabaseServices.TransactionManager
        Dim mytrans As DatabaseServices.Transaction
        Dim myBT As DatabaseServices.BlockTable
        Dim myBTR As DatabaseServices.BlockTableRecord

        myDWG =
ApplicationServices.Application.DocumentManager.MdiActiveDocument
        myDWG.LockDocument(DocumentLockMode.AutoWrite, Nothing, Nothing, True)
        myDB = myDWG.Database
        myEd = myDWG.Editor
        mytransman = myDWG.TransactionManager
        mytrans = mytransman.StartTransaction
        myBT =
myDWG.Database.BlockTableId.GetObject(DatabaseServices.OpenMode.ForRead)
        myBTR =
myBT(DatabaseServices.BlockTableRecord.ModelSpace).GetObject(DatabaseServices.O
penMode.ForWrite)

        Dim myLT As DatabaseServices.LayerTable
        Dim myLTR As DatabaseServices.LayerTableRecord
        Dim mySTE As DatabaseServices.SymbolTableEnumerator
        cbx_capa.Items.Clear()
        myLT = myDB.LayerTableId.GetObject(DatabaseServices.OpenMode.ForRead)
        mySTE = myLT.GetEnumerator
        cbx_capa.Items.Add("--Todas--")
        While mySTE.MoveNext
            myLTR = mySTE.Current.GetObject(DatabaseServices.OpenMode.ForRead)
            cbx_capa.Items.Add(myLTR.Name)
        End While
        cbx_capa.Text = "--Todas--"
        myTrans.Dispose()
        myTransMan.Dispose()

    End Sub

    Private Sub cmd_lineas_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cmd_lineas.Click

        Dim myDWG As ApplicationServices.Document
        Dim myDB As DatabaseServices.Database
        Dim myEd As EditorInput.Editor
        Dim mytransman As DatabaseServices.TransactionManager
        Dim mytrans As DatabaseServices.Transaction
        Dim myBT As DatabaseServices.BlockTable
        Dim myBTR As DatabaseServices.BlockTableRecord

        myDWG =
ApplicationServices.Application.DocumentManager.MdiActiveDocument
        myDWG.LockDocument(DocumentLockMode.AutoWrite, Nothing, Nothing, True)
        myDB = myDWG.Database
        myEd = myDWG.Editor
        mytransman = myDWG.TransactionManager
        mytrans = mytransman.StartTransaction
```

```
        myBT =
myDWG.Database.BlockTableId.GetObject(DatabaseServices.OpenMode.ForRead)
        myBTR =
myBT(DatabaseServices.BlockTableRecord.ModelSpace).GetObject(DatabaseServices.O
penMode.ForWrite)

        Dim x1, y1, z1 As Double
        Dim x2, y2, z2 As Double

        Dim nombrefichero1 As String
        OpenFileDialog1.Filter = "Archivos de texto(*.txt)|*.txt|Archivos
XYZ(*.xyz)|*.xyz|Todos los Archivos|*.*"
        OpenFileDialog1.ShowDialog() 'Mostramos diálogo de Abrir
        nombrefichero1 = OpenFileDialog1.FileName 'Colocamos en el cuadro de
texto la ruta y nombre del fichero seleccionado
        Dim sr As New IO.StreamReader(nombrefichero1)
        Dim linea As String
        Dim separador As String
        Dim capa As String
        capa = ""
        'tx_separador.Text = vbTab
        separador = txt_separador.Text
        If separador = Nothing Then separador = vbTab
        Do
            linea = sr.ReadLine()
            If Not linea Is Nothing Then
                If linea <> "" Then
                    Dim ptoArray() As String = Split(linea, separador)
                    If ptoArray.Length = 6 Then
                        x1 = ptoArray(0)
                        y1 = ptoArray(1)
                        z1 = ptoArray(2)
                        x2 = ptoArray(3)
                        y2 = ptoArray(4)
                        z2 = ptoArray(5)
                    End If
                    If ptoArray.Length = 7 Then
                        x1 = ptoArray(0)
                        y1 = ptoArray(1)
                        z1 = ptoArray(2)
                        x2 = ptoArray(3)
                        y2 = ptoArray(4)
                        z2 = ptoArray(5)
                        capa = ptoArray(6)
                    End If

                    Dim punto1 As New Geometry.Point3d(x1, y1, z1)
                    Dim punto2 As New Geometry.Point3d(x2, y2, z2)
                    Dim myline As New DatabaseServices.Line(punto1, punto2)
                    myBTR.AppendEntity(myline)

                    Try
                        If ptoArray.Length = 7 Then myline.Layer = capa
                    Catch
                        MsgBox("No existe la capa " & capa & vbCr & "La linea
se importará en la capa actual")
                    End Try

                    mytrans.AddNewlyCreatedDBObject(myline, True)
                End If
            End If
        Loop Until linea Is Nothing

        'Cerrar archivos
```

```
        sr.Close()
        sr = Nothing

        'Commit the Transaction
        mytrans.Commit()
        mytrans.Dispose()
        mytransman.Dispose()

        MsgBox("Proceso finalizado...")
    End Sub

    Private Sub cmd_triangulos_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cmd_triangulos.Click
        Dim myDWG As ApplicationServices.Document
        Dim myDB As DatabaseServices.Database
        Dim myEd As EditorInput.Editor
        Dim mytransman As DatabaseServices.TransactionManager
        Dim mytrans As DatabaseServices.Transaction
        Dim myBT As DatabaseServices.BlockTable
        Dim myBTR As DatabaseServices.BlockTableRecord

        myDWG =
ApplicationServices.Application.DocumentManager.MdiActiveDocument
        myDWG.LockDocument(DocumentLockMode.AutoWrite, Nothing, Nothing, True)
        myDB = myDWG.Database
        myEd = myDWG.Editor
        mytransman = myDWG.TransactionManager
        mytrans = mytransman.StartTransaction
        myBT =
myDWG.Database.BlockTableId.GetObject(DatabaseServices.OpenMode.ForRead)
        myBTR =
myBT(DatabaseServices.BlockTableRecord.ModelSpace).GetObject(DatabaseServices.O
penMode.ForWrite)


        Dim x1, y1, z1 As Double
        Dim x2, y2, z2 As Double
        Dim x3, y3, z3 As Double

        Dim nombrefichero1 As String
        OpenFileDialog1.Filter = "Archivos de texto(*.txt)|*.txt|Archivos
XYZ(*.xyz)|*.xyz|Todos los Archivos|*.*"
        OpenFileDialog1.ShowDialog() 'Mostramos diálogo de Abrir
        nombrefichero1 = OpenFileDialog1.FileName 'Colocamos en el cuadro de
texto la ruta y nombre del fichero seleccionado
        Dim sr As New IO.StreamReader(nombrefichero1)
        Dim linea As String
        Dim separador As String
        Dim capa As String
        capa = ""
        separador = txt_separador.Text
        If separador = Nothing Then separador = vbTab
        Do
            linea = sr.ReadLine()
            If Not linea Is Nothing Then
                If linea <> "" Then
                    Dim ptoArray() As String = Split(linea, separador)
                    If ptoArray.Length = 9 Then
                        x1 = ptoArray(0)
                        y1 = ptoArray(1)
                        z1 = ptoArray(2)
                        x2 = ptoArray(3)
                        y2 = ptoArray(4)
                        z2 = ptoArray(5)
```

```
                        x3 = ptoArray(6)
                        y3 = ptoArray(7)
                        z3 = ptoArray(8)
                    End If
                    If ptoArray.Length = 10 Then
                        x1 = ptoArray(0)
                        y1 = ptoArray(1)
                        z1 = ptoArray(2)
                        x2 = ptoArray(3)
                        y2 = ptoArray(4)
                        z2 = ptoArray(5)
                        x3 = ptoArray(6)
                        y3 = ptoArray(7)
                        z3 = ptoArray(8)
                        capa = ptoArray(9)
                    End If
                    Dim mypoints As New Geometry.Point3dCollection
                    mypoints.Add(New Geometry.Point3d(x1, y1, z1))
                    mypoints.Add(New Geometry.Point3d(x2, y2, z2))
                    mypoints.Add(New Geometry.Point3d(x3, y3, z3))
                    Dim my3dobj As New
DatabaseServices.Polyline3d(Poly3dType.SimplePoly, mypoints, True)
                    myBTR.AppendEntity(my3dobj)
                    Try
                        If ptoArray.Length = 10 Then my3dobj.Layer = capa
                    Catch
                        MsgBox("No existe la capa " & capa & vbCr & "El
triangulo se importará en la capa actual")
                    End Try
                    mytrans.AddNewlyCreatedDBObject(my3dobj, True)

                End If
            End If
        Loop Until linea Is Nothing

        'Cerrar archivos
        sr.Close()
        sr = Nothing

        'Commit the Transaction
        mytrans.Commit()
        mytrans.Dispose()
        mytransman.Dispose()

        MsgBox("Proceso finalizado...")
    End Sub

    Private Sub cmd_textos_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cmd_textos.Click
        Dim myDWG As ApplicationServices.Document
        Dim myDB As DatabaseServices.Database
        Dim myEd As EditorInput.Editor
        Dim mytransman As DatabaseServices.TransactionManager
        Dim mytrans As DatabaseServices.Transaction
        Dim myBT As DatabaseServices.BlockTable
        Dim myBTR As DatabaseServices.BlockTableRecord

        myDWG =
ApplicationServices.Application.DocumentManager.MdiActiveDocument
        myDWG.LockDocument(DocumentLockMode.AutoWrite, Nothing, Nothing, True)
        myDB = myDWG.Database
        myEd = myDWG.Editor
        mytransman = myDWG.TransactionManager
        mytrans = mytransman.StartTransaction
```

```
        myBT =
myDWG.Database.BlockTableId.GetObject(DatabaseServices.OpenMode.ForRead)
        myBTR =
myBT(DatabaseServices.BlockTableRecord.ModelSpace).GetObject(DatabaseServices.O
penMode.ForWrite)

        Dim x, y, z As Double

        Dim nombrefichero1 As String
        OpenFileDialog1.Filter = "Archivos de Texto(*.txt)|*.txt|Todos los
Archivos|*.*"
        OpenFileDialog1.ShowDialog() 'Mostramos diálogo de Abrir
        nombrefichero1 = OpenFileDialog1.FileName 'Colocamos en el cuadro de
texto la ruta y nombre del fichero seleccionado
        Dim sr As New IO.StreamReader(nombrefichero1)
        Dim linea As String
        Dim contenido As String
        Dim separador As String
        Dim capa As String
        Dim texto As String
        Dim myModelSpace As BlockTableRecord =
myBT(BlockTableRecord.ModelSpace).GetObject(OpenMode.ForWrite)
        Dim tamano As String
        Dim pos_text As String
        pos_text = ""
        tamano = ""
        capa = "" : texto = ""
        contenido = ""
        txt_separador.Text = vbTab
        separador = txt_separador.Text
        If separador = Nothing Then separador = " "
        Do
            linea = sr.ReadLine()
            If Not linea Is Nothing Then
                If linea <> "" Then
                    Dim ptoArray() As String = Split(linea, separador)
                    x = ptoArray(0)
                    y = ptoArray(1)
                    z = ptoArray(2)
                    If pos_text = "" Then
                        pos_text = InputBox("Elegir la posición del texto a
partir del 0" & vbCrLf & linea)
                    End If
                    texto = ptoArray(CInt(pos_text))
                    Using myText As New DBText()
                        myText.TextString = texto
                        If tamano = "" Then tamano = InputBox("Tamaño del texto
a insertar ...")
                        If CSng(tamano) = 0 Then tamano = "1"
                        myText.Height = CSng(tamano)
                        Dim puntoIns As New Geometry.Point3d(x, y, z)
                        myText.Position = puntoIns
                        myModelSpace.AppendEntity(myText)
                        mytrans.AddNewlyCreatedDBObject(myText, True)
                    End Using
                End If
            End If
        Loop Until linea Is Nothing
        myModelSpace.Dispose()
        'Cerrar archivos
        sr.Close()
        sr = Nothing

        mytrans.Commit()
```

```
        mytrans.Dispose()
        mytransman.Dispose()

        MsgBox("Proceso finalizado...")
    End Sub
End Class
```

7.5 Código fuente comentado del comando vbLote

```
Imports Autodesk.AutoCAD.Runtime
Imports Autodesk.AutoCAD
Imports Autodesk.AutoCAD.DatabaseServices
Imports Autodesk.AutoCAD.EditorInput
Imports Autodesk.AutoCAD.Geometry
Imports Autodesk.AutoCAD.ApplicationServices
Imports Autodesk.AutoCAD.ApplicationServices.Application
Imports Autodesk.AutoCAD.LayerManager
Imports Autodesk.AutoCAD.PlottingServices
Imports System.Data.OleDb
Imports System.Console
Imports System.IO


Public Class vb_lote
    Public linea As String

    <CommandMethod("lote", CommandFlags.Session)> _
Public Sub lote()
        'Indicamos el txt que contiene las direcciones de los dibujos
        Dim nombrefichero As String
        nombrefichero = "E:\inaki\1102SIM_programaLOTE\texto.txt" 'Colocamos en
el cuadro de texto la ruta y nombre del fichero seleccionado"

        'Indicamos el dwg que contiene el cajetin y el nombre del bloque
        Dim nombrecajetin, nombreDwg As String
        nombrecajetin = "Kajetin"
        nombreDwg =
"C:\inaki\0308SIG\support\plantillasCajetin\110217_CAS_H_A2.dwg"
        Dim sr As New IO.StreamReader(nombrefichero)
        Do
            linea = sr.ReadLine()
            If Not linea Is Nothing Then
                Dim newdwg As Autodesk.AutoCAD.ApplicationServices.Document
                newdwg =
ApplicationServices.Application.DocumentManager.Open(linea, False)

                'Subrutinas(ejecutadas)
                definecajetin(nombrecajetin, nombreDwg)
                borraelementoscapa_espaciopapel()
                insertaCajetin(nombrecajetin)
                explotavaloresBloque()
                cambiavalorestextos()
                imprimePDF()
                newdwg.CloseAndSave(newdwg.Name)
            End If
        Loop Until linea Is Nothing
        sr.Close()
        sr = Nothing
    End Sub

    Public Sub definecajetin(ByVal nombrecajetin As String, ByVal nombreDwg As
String)
        Dim myDWG As Autodesk.AutoCAD.ApplicationServices.Document
        Dim myDB As Database = HostApplicationServices.WorkingDatabase
        Dim mytransman As DatabaseServices.TransactionManager
        myDWG = Application.DocumentManager.MdiActiveDocument
        mytransman = myDWG.TransactionManager
        Using DocLoc As DocumentLock = myDWG.LockDocument
            Using mytrans = mytransman.StartTransaction
                Try
```

```
                    Dim extDB As New Database(False, True)
                    extDB.ReadDwgFile(nombreDwg, IO.FileShare.Read, True, "")
                    Dim id As ObjectId = myDB.Insert(nombrecajetin, extDB,
False)
                    extDB.CloseInput(True)
                    extDB.Dispose()
                    mytrans.Commit()
                    mytrans.Dispose()
                    mytransman.Dispose()
                Catch Ex As System.Exception
                End Try
            End Using
        End Using
    End Sub

    Sub borraelementoscapa_espaciopapel()
        'Coge el bloque que está en el espacio papel en la capa papel_cajetin y
lo explota
        Dim myEd As Editor = DocumentManager.MdiActiveDocument.Editor
        Dim myPSR As PromptSelectionResult = myEd.SelectAll
        Dim myDWG As Autodesk.AutoCAD.ApplicationServices.Document
        myDWG = Application.DocumentManager.MdiActiveDocument
        Dim myDB As Database = HostApplicationServices.WorkingDatabase
        Dim mytransman As DatabaseServices.TransactionManager
        mytransman = myDWG.TransactionManager

        Dim myTrans As Transaction =
myEd.Document.TransactionManager.StartTransaction

        Dim myBT As BlockTable =
myDWG.Database.BlockTableId.GetObject(OpenMode.ForRead)
        Dim myBTR As BlockTableRecord
        Using DocLoc As DocumentLock = myDWG.LockDocument
            myBTR =
myBT(BlockTableRecord.PaperSpace).GetObject(OpenMode.ForWrite)
            mytransman = myDWG.TransactionManager
            If myPSR.Status = PromptStatus.OK Then
                Dim mySS As SelectionSet = myPSR.Value
                For Each mySelObj As SelectedObject In mySS
                    Dim myEnt As Entity =
mySelObj.ObjectId.GetObject(OpenMode.ForWrite)
                    If myEnt.Layer = "CAS_PAPEL_CAJETIN" Then
                        myEnt.Erase()
                        myEnt.Dispose()
                    End If
                Next
                myTrans.Commit()
                myTrans.Dispose()
                mytransman.Dispose()
            End If
        End Using
    End Sub

    Public Sub insertaCajetin(ByVal nombrecajetin As String)
        Dim myDWG As Autodesk.AutoCAD.ApplicationServices.Document
        Dim myDB As Database = HostApplicationServices.WorkingDatabase
        Dim mytransman As DatabaseServices.TransactionManager
        myDWG = Application.DocumentManager.MdiActiveDocument
        mytransman = myDWG.TransactionManager
        Using DocLoc As DocumentLock = myDWG.LockDocument
            Using mytrans = mytransman.StartTransaction
                Try
                    'insertar bloque en el dibujo
```

```
                    Dim myBT As BlockTable =
myDWG.Database.BlockTableId.GetObject(OpenMode.ForRead)
                    Dim myBTR As BlockTableRecord
                    myBTR =
myBT(BlockTableRecord.PaperSpace).GetObject(OpenMode.ForWrite)

                    Dim myBlockDef As BlockTableRecord =
myBT(nombrecajetin).GetObject(OpenMode.ForRead)
                    Dim MyBlockRef As New DatabaseServices.BlockReference(New
Geometry.Point3d(0, 0, 0), myBT(nombrecajetin))
                    MyBlockRef.Layer = "CAS_PAPEL_CAJETIN"
                    myBTR.AppendEntity(MyBlockRef)
                    mytrans.AddNewlyCreatedDBObject(MyBlockRef, True)

                    mytrans.Commit()
                    mytrans.Dispose()
                    mytransman.Dispose()

                Catch Ex As System.Exception
                End Try
            End Using
        End Using
    End Sub

    Public Sub explotavaloresBloque()
        'Coge el bloque que está en el espacio papel en la capa papel_cajetin y
lo explota
        Dim myEd As Editor = DocumentManager.MdiActiveDocument.Editor
        Dim myPSR As PromptSelectionResult = myEd.SelectAll
        Dim myDWG As Autodesk.AutoCAD.ApplicationServices.Document
        myDWG = Application.DocumentManager.MdiActiveDocument
        Dim myDB As Database = HostApplicationServices.WorkingDatabase
        Dim mytransman As DatabaseServices.TransactionManager
        mytransman = myDWG.TransactionManager

        Dim myTrans As Transaction =
myEd.Document.TransactionManager.StartTransaction

        Dim myBT As BlockTable =
myDWG.Database.BlockTableId.GetObject(OpenMode.ForRead)
        Dim myBTR As BlockTableRecord
        Using DocLoc As DocumentLock = myDWG.LockDocument
            myBTR =
myBT(BlockTableRecord.PaperSpace).GetObject(OpenMode.ForWrite)
            mytransman = myDWG.TransactionManager
            If myPSR.Status = PromptStatus.OK Then
                Dim mySS As SelectionSet = myPSR.Value
                For Each mySelObj As SelectedObject In mySS
                    Dim myEnt As Entity =
mySelObj.ObjectId.GetObject(OpenMode.ForWrite)
                    If mySelObj.ObjectId.ObjectClass.DxfName = "INSERT" And
myEnt.Layer = "CAS_PAPEL_CAJETIN" Then
                        Dim acDBObjColl As DBObjectCollection = New
DBObjectCollection()
                        myEnt.Explode(acDBObjColl)
                        For Each acEnt As Entity In acDBObjColl
                            myBTR.AppendEntity(acEnt)
                            myTrans.AddNewlyCreatedDBObject(acEnt, True)
                        Next
                        myEnt.Erase()
                        myEnt.Dispose()
                        myTrans.Commit()
                        myTrans.Dispose()
                        mytransman.Dispose()
```

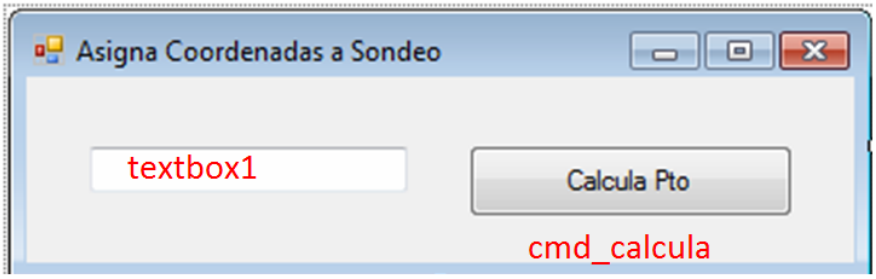
```
                        Exit For
                    End If
                Next
            End If
        End Using
    End Sub

    Public Sub cambiavalorestextos()
        Dim myDWG As Autodesk.AutoCAD.ApplicationServices.Document
        myDWG = Application.DocumentManager.MdiActiveDocument

        Dim nactual, nomplano, caracter As String
        nactual = myDWG.Name
        nomplano = ""
        For x = 1 To Len(nactual)
            caracter = Mid(nactual, x, 1)
            If caracter = "\" Then
                nomplano = ""
            Else
                nomplano = nomplano & caracter
            End If
        Next
        nomplano = Mid(nomplano, 1, Len(nomplano) - 4)

        Dim myDB As Database = HostApplicationServices.WorkingDatabase
        Dim mytransman As DatabaseServices.TransactionManager
        mytransman = myDWG.TransactionManager

        'Se establecen los valores que habrá que sustituir en los elementos de
texto actuales
        Dim dbconnection As OleDbConnection
        Dim dbdataset As DataSet
        Dim dbdataadapter As OleDbDataAdapter
        Dim cadenaconexion As String
        Dim archivodatos As String
        'Dim nombretabla As String
        Dim camino As String
        Dim sentencia As String

        archivodatos = "Cajetines2d.mdb"
        camino = "C:\inaki\0308SIG\SUPPORT"
        cadenaconexion = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" &
camino & "\" & archivodatos & ";"
        dbconnection = New OleDbConnection(cadenaconexion)
        dbconnection.Open()


        sentencia = "SELECT * FROM con_cajetin WHERE nombre='" & nomplano & "'"

        dbdataset = New DataSet
        dbdataadapter = New OleDbDataAdapter(sentencia, dbconnection)
        Dim commandbuilder As New OleDbCommandBuilder(dbdataadapter)
        dbdataadapter.MissingSchemaAction = MissingSchemaAction.AddWithKey
        dbdataadapter.Fill(dbdataset, "tabla1")
        Dim nivel1, nivel2, titulo As String
        nivel1 = "" : nivel2 = "" : titulo = ""
        If dbdataset.Tables(0).Rows.Count = 1 Then
            Using mytrans As Transaction =
HostApplicationServices.WorkingDatabase.TransactionManager.StartTransaction
                nivel1 = dbdataset.Tables("tabla1").Rows(0).Item("nivel1")
                nivel2 = dbdataset.Tables("tabla1").Rows(0).Item("nivel2")
                titulo = dbdataset.Tables("tabla1").Rows(0).Item("titulo")
                mytrans.Commit()
                dbconnection.Close()
```

```
                mytrans.Dispose()
                mytransman.Dispose()
            End Using
        End If

        Dim acDocEd As Editor
        Dim acTypValAr(3) As TypedValue
        Dim acSelFtr As SelectionFilter
        Dim acSSPrompt As PromptSelectionResult
        Dim acSSet As SelectionSet
        Dim selObj As SelectedObject

        acDocEd = Application.DocumentManager.MdiActiveDocument.Editor

        acTypValAr.SetValue(New TypedValue(DxfCode.Operator, "<or"), 0)
        acTypValAr.SetValue(New TypedValue(DxfCode.Start, "TEXT"), 1)
        acTypValAr.SetValue(New TypedValue(DxfCode.Start, "MTEXT"), 2)
        acTypValAr.SetValue(New TypedValue(DxfCode.Operator, "or>"), 3)

        acSelFtr = New SelectionFilter(acTypValAr)
        acSSPrompt = acDocEd.SelectAll(acSelFtr)

        If acSSPrompt.Status = PromptStatus.OK Then
            acSSet = acSSPrompt.Value
            Using DocLoc As DocumentLock = myDWG.LockDocument
                Using trans As Transaction =
HostApplicationServices.WorkingDatabase.TransactionManager.StartTransaction
                    Try
                        For Each selObj In acSSet
                            Select Case selObj.ObjectId.ObjectClass.Name
                                Case "AcDbText"
                                    Dim textObj As DBText =
trans.GetObject(selObj.ObjectId, OpenMode.ForWrite, False, True)
                                  If textObj.TextString = "11111" Then
textObj.TextString = nivel1
                                    If textObj.TextString = "22222" Then
textObj.TextString = nomplano
                                    If textObj.TextString = "33333" Then
textObj.TextString = titulo
                                    If textObj.TextString = "44444" Then
textObj.TextString = nomplano
                                    If textObj.TextString = "55555" Then
textObj.TextString = titulo
                                    If textObj.TextString = "66666" Then
textObj.TextString = nivel2
                                    textObj.Dispose()
                                Case "AcDbMText"
                                    Dim mtextObj As MText =
trans.GetObject(selObj.ObjectId, OpenMode.ForRead, False, True)
                                mtextObj.Dispose()
                            End Select
                        Next selObj
                    Catch ex As Exception
                    Finally
                        trans.Commit()
                    End Try
                End Using
            End Using
        End If
    End Sub

    Sub imprimePDF()
        Dim myDWG As Autodesk.AutoCAD.ApplicationServices.Document
        myDWG = Application.DocumentManager.MdiActiveDocument
```

```
        Application.SetSystemVariable("BACKGROUNDPLOT", 0)
        Using tr As Transaction =
Application.DocumentManager.MdiActiveDocument.TransactionManager.StartTransacti
on()
            Try
                Dim layMgr As LayoutManager = LayoutManager.Current
                Dim loObjId As ObjectId =
layMgr.GetLayoutId(layMgr.CurrentLayout)
                Dim lo As Layout = DirectCast(tr.GetObject(loObjId,
OpenMode.ForRead), Layout)
                Dim ps As New PlotSettings(False)

                ps.CopyFrom(lo)
                Dim pi As New PlotInfo()
                pi.Layout = loObjId
                Dim psv As PlotSettingsValidator =
Autodesk.AutoCAD.DatabaseServices.PlotSettingsValidator.Current
                psv.SetPlotConfigurationName(ps, "DWG To PDF.pc3",
"ISO_full_bleed_A2_(420.00_x_594.00_MM)")
                psv.RefreshLists(ps)
                psv.SetPlotType(ps,
Autodesk.AutoCAD.DatabaseServices.PlotType.Layout)
                psv.SetUseStandardScale(ps, True)
                psv.SetStdScaleType(ps, StdScaleType.StdScale1To1)
                psv.SetCurrentStyleSheet(ps, "universal.ctb")
                pi.OverrideSettings = ps
                Dim pe As PlotEngine = PlotFactory.CreatePublishEngine()
                Try
                    pe.BeginPlot(Nothing, Nothing)
                    Dim validator As New PlotInfoValidator()
                    validator.MediaMatchingPolicy =
Autodesk.AutoCAD.PlottingServices.MatchingPolicy.MatchEnabled
                    validator.Validate(pi)
                    Dim plotFile As String =
Path.GetFullPath(Application.DocumentManager.MdiActiveDocument.Database.Filenam
e)
                    plotFile = plotFile.Remove(plotFile.Length - 4)
                    pe.BeginDocument(pi,
Application.DocumentManager.MdiActiveDocument.Database.Filename, Nothing, 1,
True, plotFile & ".pdf")
                    Dim pageInfo As New PlotPageInfo()
                    pe.BeginPage(pageInfo, pi, True, Nothing)
                    pe.BeginGenerateGraphics(Nothing)
                    pe.EndGenerateGraphics(Nothing)
                    pe.EndPage(Nothing)
                    pe.EndDocument(Nothing)
                    pe.EndPlot(Nothing)
                Catch ex As System.Exception
                    System.Windows.Forms.MessageBox.Show(ex.Message)
                End Try
                pe.Destroy()
                tr.Commit()
            Catch ex As System.Exception
                System.Windows.Forms.MessageBox.Show(ex.Message)
            End Try
        End Using
    End Sub
    End Class
```

7.6 Código fuente comentado del programa Asignar Coordenadas a Sondeo

frm_asigcoorsondeo

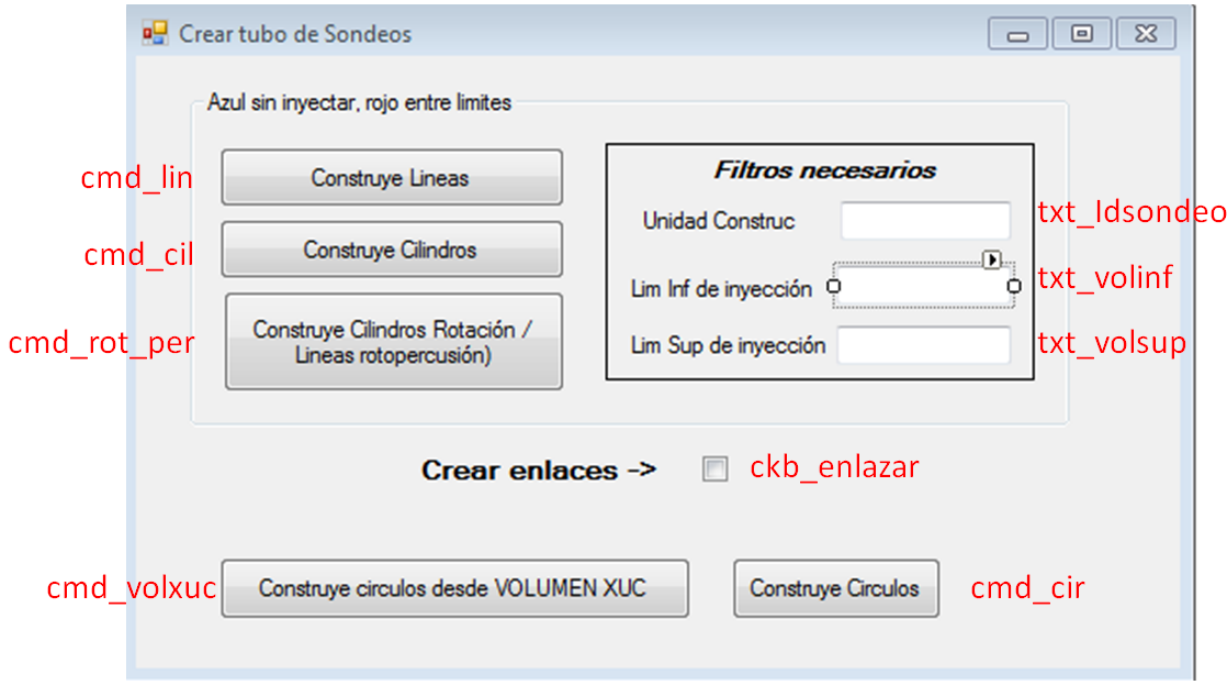



```
Imports Autodesk.AutoCAD.Runtime
Imports Autodesk.AutoCAD
Imports Autodesk.AutoCAD.DatabaseServices
Imports Autodesk.AutoCAD.EditorInput
Imports Autodesk.AutoCAD.Geometry
Imports Autodesk.AutoCAD.ApplicationServices
Imports Autodesk.AutoCAD.ApplicationServices.Application
Imports Autodesk.AutoCAD.LayerManager
Imports Autodesk.AutoCAD.Windows

Imports System.Data.OleDb
Imports System.Data.OleDb.OleDbDataAdapter
Imports System.Data.OleDb.OleDbTransaction

Imports System.Console

Public Class frm_asigcoorsondeo
    Private Sub cmd_calcula_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cmd_calcula.Click
        Me.Hide()
        ' Definición de variables
        Dim pi As Double
        pi = 3.14159265
        Dim myEd As Editor = DocumentManager.MdiActiveDocument.Editor
        Dim p1 As PromptPointResult
        Dim corX, corY, corZ As Single

        'Selecciona puntos en el dibujo
        p1 = myEd.GetPoint("Selecciona un punto:" & vbCr)
        corX = Math.Round(p1.Value(0), 3) : corY = Math.Round(p1.Value(1), 3) :
corZ = Math.Round(p1.Value(2), 3)
        Dim cn As OleDbConnection
        Dim cmd As OleDbCommand
        Dim icount As Integer
        icount = 0

        'Asignación de sentencia y caminos
        Dim autoguardado, camino, sentencia As String
        autoguardado =
Autodesk.AutoCAD.ApplicationServices.Application.Preferences.Files.autosavepath
        camino = autoguardado & "\DATOS"
        sentencia = "SELECT * FROM tbl_Sondeos"
```

```
        'Conexión con la base de datos e inserción mediante código SQL del
registro correspondiente
        Try
            cn = New OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=" & camino & "\SONDEOS.mdb;")
            cn.Open()
            sentencia = "insert into tbl_sondeos (IDSONDEO,x,y,z) VALUES ('" &
LTrim(RTrim(TextBox1.Text)) & "'," & corX & "," & corY & "," & corZ & ")"
            cmd = New OleDbCommand(sentencia, cn)
            icount = cmd.ExecuteNonQuery()
            cn.Close()
        Catch
        End Try
        If icount = 0 Then MsgBox("No se ha añadido el registro. Revise los
valores")
        Me.Show()

    End Sub
End Class
```

7.7 Código fuente comentado del programa Crear Sondeos

frm_lineasondeos.vb




```
Imports Autodesk.AutoCAD.Runtime
Imports Autodesk.AutoCAD
Imports Autodesk.AutoCAD.DatabaseServices
Imports Autodesk.AutoCAD.EditorInput
Imports Autodesk.AutoCAD.Geometry
Imports Autodesk.AutoCAD.ApplicationServices
Imports Autodesk.AutoCAD.ApplicationServices.Application
Imports Autodesk.AutoCAD.LayerManager
Imports Autodesk.AutoCAD.Windows

Imports System.Data.OleDb
Imports System.Data.OleDb.OleDbDataAdapter
Imports System.Data.OleDb.OleDbTransaction

Imports System.Console
Public Class frm_lineasondeos
    Dim p1(0 To 2) As Double
    Dim IDSONDEO As String
    Dim txtvolinff As Double
    Dim txtvolsupp As Double


    Private Sub cmd_lin_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cmd_lin.Click

        Dim pto(0 To 2) As Double
        Dim pi As Double
        pi = 3.14159265

        'Asignar los limites superior e inferior para dar el color 1 o 4
        If txtvolinf.Text = "" Then
            txtvolinff = 0
        Else
            txtvolinff = CDbl(txtvolinf.Text)
        End If
```

```
        If txtvolsup.Text = "" Then
            txtvolsupp = 50000
        Else
            txtvolsupp = txtvolsup.Text
        End If

        'Se asigna la variable que determina si se enlazarán con plantilla de
vinculo
        Dim enlace As String
        If ckb_enlazar.Checked = True Then enlace = "BAI" Else enlace = "EZ"

        'Declaración y enlace de datos
        Dim dbconnection As OleDbConnection
        Dim dbdataset As DataSet
        Dim dbdataadapter As OleDbDataAdapter
        Dim cadenaconexion As String
        Dim archivodatos As String
        Dim nombretabla As String

        'Comunicación con base de datos de sondeos
        Dim autoguardado, camino, sentencia As String
        autoguardado =
Autodesk.AutoCAD.ApplicationServices.Application.Preferences.Files.autosavepath
        camino = autoguardado & "\DATOS"
        sentencia = "SELECT * FROM tbl_Sondeos"
        archivodatos = "SONDEOS.MDB"
        nombretabla = "Sondeos_tabla"
        cadenaconexion = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" &
camino & "\" & archivodatos & ";"
        dbconnection = New OleDbConnection(cadenaconexion)
        dbconnection.Open()
        Dim form1 As New frm_lineasondeos

        sentencia = "SELECT * FROM tbl_sondeos WHERE UC='" &
LCase(txtIdsondeo.Text) & "'"

        dbdataset = New DataSet
        dbdataadapter = New OleDbDataAdapter(sentencia, dbconnection)
        Dim commandbuilder As New OleDbCommandBuilder(dbdataadapter)
        dbdataadapter.MissingSchemaAction = MissingSchemaAction.AddWithKey
        dbdataadapter.Fill(dbdataset, nombretabla)
        Dim fila As DataRow
        Dim p1(0 To 2) As Double
        Dim dis, angHH, angVV As Double

        If dbdataset.Tables.Item(0).Rows.Count > 0 Then
            For Each fila In dbdataset.Tables(nombretabla).Rows
                IDSONDEO = fila("idsondeo").ToString
                p1(0) = fila("X").ToString
                p1(1) = fila("Y").ToString
                p1(2) = fila("Z").ToString
                dis = fila("PROFUNDIDAD").ToString
                angHH = fila("anguloH").ToString
                angVV = fila("anguloV").ToString
                construirsondeos(p1, dis, angHH, angVV, IDSONDEO, txtvolinff,
txtvolsupp, enlace)
            Next
        End If
        Dim mensaje As String
        mensaje = "Recuerda: " & vbCrLf & " Azul indica que la inyección está
entre los límites" & _
        vbCrLf & " Rojo indica que la inyección está fuera de los límites"
```

```
        If enlace = "BAI" Then mensaje = mensaje & vbCrLf & vbCrLf & " Las
entidades generadas se han enlazado"
        MsgBox(mensaje)
    End Sub

    Private Sub cmd_rot_per_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cmd_rot_per.Click
        Dim pto(0 To 2) As Double
        Dim pi As Double
        pi = 3.14159265

        'Asignar los limites superior e inferior para dar el color 1 o 4
        If txtvolinf.Text = "" Then
            txtvolinff = 0
        Else
            txtvolinff = CDbl(txtvolinf.Text)
        End If

        If txtvolsup.Text = "" Then
            txtvolsupp = 50000
        Else
            txtvolsupp = txtvolsup.Text
        End If

        Dim enlace As String
        If ckb_enlazar.Checked = True Then enlace = "BAI" Else enlace = "EZ"

        'Declaración y enlace de datos
        Dim dbconnection As OleDbConnection
        Dim dbdataset, dbdataset1 As DataSet
        Dim dbdataadapter, dbdataadapter1 As OleDbDataAdapter
        Dim cadenaconexion As String
        Dim archivodatos As String
        Dim nombretabla As String

        'Comunicación con base de datos de sondeos
        Dim autoguardado, camino, sentencia As String
        autoguardado =
Autodesk.AutoCAD.ApplicationServices.Application.Preferences.Files.autosavepath
        camino = autoguardado & "\DATOS"
        sentencia = "SELECT * FROM tbl_Sondeos"
        archivodatos = "SONDEOS.MDB"
        nombretabla = "tabla1"
        cadenaconexion = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" &
camino & "\" & archivodatos & ";"
        dbconnection = New OleDbConnection(cadenaconexion)
        dbconnection.Open()
        Dim form1 As New frm_lineasondeos

        sentencia = "SELECT * FROM tbl_sondeos WHERE UC='" &
LCase(txtIdsondeo.Text) & "'"

        dbdataset = New DataSet
        dbdataadapter = New OleDbDataAdapter(sentencia, dbconnection)
        Dim commandbuilder As New OleDbCommandBuilder(dbdataadapter)
        dbdataadapter.MissingSchemaAction = MissingSchemaAction.AddWithKey
        dbdataadapter.Fill(dbdataset, nombretabla)
        Dim fila, fila1 As DataRow
        Dim p1(0 To 2) As Double
        Dim dis, angHH, angVV As Double
        Dim sist As String
        If dbdataset.Tables.Item(0).Rows.Count > 0 Then
            For Each fila In dbdataset.Tables(nombretabla).Rows
                IDSONDEO = fila("idsondeo").ToString
```

```
                p1(0) = fila("X").ToString
                p1(1) = fila("Y").ToString
                p1(2) = fila("Z").ToString
                dis = fila("PROFUNDIDAD").ToString
                angHH = fila("anguloH").ToString
                angVV = fila("anguloV").ToString
                sentencia = "SELECT SISTEMA FROM tbl_perforaciones WHERE
idperforacion='" & IDSONDEO & "'"
                dbdataset1 = New DataSet
                dbdataadapter1 = New OleDbDataAdapter(sentencia, dbconnection)
                Dim commandbuilder1 As New OleDbCommandBuilder(dbdataadapter1)
                dbdataadapter1.MissingSchemaAction =
MissingSchemaAction.AddWithKey
                nombretabla = "sonding"
                dbdataadapter1.Fill(dbdataset1, nombretabla)
                If dbdataset1.Tables(0).Rows.Count > 0 Then
                    For Each fila1 In dbdataset1.Tables(nombretabla).Rows
                        sist = fila1("sistema")
                        If sist = "Rotación" Then construircilindros(p1, dis,
angHH, angVV, IDSONDEO, txtvolinff, txtvolsupp, enlace)
                        If sist = "Rotopercusión" Then construirsondeos(p1,
dis, angHH, angVV, IDSONDEO, txtvolinff, txtvolsupp, enlace)
                    Next
                End If
            Next
        End If
        Dim mensaje As String
        mensaje = "Recuerda: " & vbCrLf & " Azul indica que la inyección está
entre los límites" & _
        vbCrLf & " Rojo indica que la inyección está fuera de los límites" &
vbCrLf & _
        vbCrLf & " Cilindro indica perforación por rotación" & vbCrLf & _
        " Linea indica perforación por rotopercusión"
        If enlace = "BAI" Then mensaje = mensaje & vbCrLf & vbCrLf & " Las
entidades generadas se han enlazado"
        MsgBox(mensaje)

    End Sub


    Private Sub cmd_cil_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cmd_cil.Click
        Dim pto(0 To 2) As Double
        Dim pi As Double
        pi = 3.14159265

        'Asignar los limites superior e inferior para dar el color 1 o 4
        If txtvolinf.Text = "" Then
            txtvolinff = 0
        Else
            txtvolinff = CDbl(txtvolinf.Text)
        End If

        If txtvolsup.Text = "" Then
            txtvolsupp = 50000
        Else
            txtvolsupp = txtvolsup.Text
        End If

        Dim enlace As String
        If ckb_enlazar.Checked = True Then enlace = "BAI" Else enlace = "EZ"

        'Declaración y enlace de datos
        Dim dbconnection As OleDbConnection
```

```
        Dim dbdataset As DataSet
        Dim dbdataadapter As OleDbDataAdapter
        Dim cadenaconexion As String
        Dim archivodatos As String
        Dim nombretabla As String

        'Comunicación con base de datos de sondeos
        Dim autoguardado, camino, sentencia As String
        autoguardado =
Autodesk.AutoCAD.ApplicationServices.Application.Preferences.Files.autosavepath
        camino = autoguardado & "\DATOS"
        sentencia = "SELECT * FROM tbl_Sondeos"
        archivodatos = "SONDEOS.MDB"
        nombretabla = "Sondeos_tabla"
        cadenaconexion = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" &
camino & "\" & archivodatos & ";"
        dbconnection = New OleDbConnection(cadenaconexion)
        dbconnection.Open()
        Dim form1 As New frm_lineasondeos

        sentencia = "SELECT * FROM tbl_sondeos WHERE UC='" &
LCase(txtIdsondeo.Text) & "'"

        dbdataset = New DataSet
        dbdataadapter = New OleDbDataAdapter(sentencia, dbconnection)
        Dim commandbuilder As New OleDbCommandBuilder(dbdataadapter)
        dbdataadapter.MissingSchemaAction = MissingSchemaAction.AddWithKey
        dbdataadapter.Fill(dbdataset, nombretabla)
        Dim fila As DataRow
        Dim p1(0 To 2) As Double
        Dim dis, angHH, angVV As Double

        If dbdataset.Tables.Item(0).Rows.Count > 0 Then
            For Each fila In dbdataset.Tables(nombretabla).Rows
                IDSONDEO = fila("idsondeo").ToString
                p1(0) = fila("X").ToString
                p1(1) = fila("Y").ToString
                p1(2) = fila("Z").ToString
                dis = fila("PROFUNDIDAD").ToString
                angHH = fila("anguloH").ToString
                angVV = fila("anguloV").ToString
                construircilindros(p1, dis, angHH, angVV, IDSONDEO, txtvolinff,
txtvolsupp, enlace)
            Next
        End If

        Dim mensaje As String
        mensaje = "Recuerda: " & vbCrLf & " Azul indica que la inyección está
entre los límites" & _
        vbCrLf & " Rojo indica que la inyección está fuera de los límites"
        If enlace = "BAI" Then mensaje = mensaje & vbCrLf & vbCrLf & " Las
entidades generadas se han enlazado"
        MsgBox(mensaje)

    End Sub

    Private Sub cmd_cir_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cmd_cir.Click
        Dim pto(0 To 2) As Double
        Dim pi As Double
        pi = 3.14159265

        'Asignar los limites superior e inferior para dar el color 1 o 4
        If txtvolinf.Text = "" Then
```

```
            txtvolinff = 0
        Else
            txtvolinff = CDbl(txtvolinf.Text)
        End If

        If txtvolsup.Text = "" Then
            txtvolsupp = 50000
        Else
            txtvolsupp = txtvolsup.Text
        End If

        Dim enlace As String
        If ckb_enlazar.Checked = True Then enlace = "BAI" Else enlace = "EZ"

        'Declaración y enlace de datos
        Dim dbconnection As OleDbConnection
        Dim dbdataset As DataSet
        Dim dbdataadapter As OleDbDataAdapter
        Dim cadenaconexion As String
        Dim archivodatos As String
        Dim nombretabla As String

        'Comunicación con base de datos de sondeos
        Dim autoguardado, camino, sentencia As String
        autoguardado =
Autodesk.AutoCAD.ApplicationServices.Application.Preferences.Files.autosavepath
        camino = autoguardado & "\DATOS"
        sentencia = "SELECT * FROM tbl_Sondeos"
        archivodatos = "SONDEOS.MDB"
        nombretabla = "Sondeos_tabla"
        cadenaconexion = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" &
camino & "\" & archivodatos & ";"
        dbconnection = New OleDbConnection(cadenaconexion)
        dbconnection.Open()
        Dim form1 As New frm_lineasondeos

        sentencia = "SELECT * FROM tbl_sondeos WHERE UC='" &
LCase(txtIdsondeo.Text) & "'"

        dbdataset = New DataSet
        dbdataadapter = New OleDbDataAdapter(sentencia, dbconnection)
        Dim commandbuilder As New OleDbCommandBuilder(dbdataadapter)
        dbdataadapter.MissingSchemaAction = MissingSchemaAction.AddWithKey
        dbdataadapter.Fill(dbdataset, nombretabla)
        Dim fila As DataRow
        Dim p1(0 To 2) As Double
        Dim dis, angHH, angVV As Double

        If dbdataset.Tables.Item(0).Rows.Count > 0 Then
            For Each fila In dbdataset.Tables(nombretabla).Rows
                IDSONDEO = fila("idsondeo").ToString
                p1(0) = fila("X").ToString
                p1(1) = fila("Y").ToString
                p1(2) = fila("Z").ToString
                dis = fila("PROFUNDIDAD").ToString
                angHH = fila("anguloH").ToString
                angVV = fila("anguloV").ToString
                construircirculos(p1, dis, angHH, angVV, IDSONDEO, txtvolinff,
txtvolsupp, enlace)
            Next
        End If

        Dim mensaje As String
```

```
        mensaje = "Recuerda: " & vbCrLf & " Azul indica que la inyección está
entre los límites" & _
        vbCrLf & " Rojo indica que la inyección está fuera de los límites"
        If enlace = "BAI" Then mensaje = mensaje & vbCrLf & vbCrLf & " Las
entidades generadas se han enlazado"
        MsgBox(mensaje)
    End Sub

    Private Sub frm_lineasondeos_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        txtvolinf.Text = "5"
        txtvolsup.Text = "50000"
    End Sub

End Class


Module md_sondeos
    Public z_xy As Double, y_xz As Double, x_yz As Double
    Sub construirsondeos(ByVal p1() As Double, ByVal dis As Double, ByVal angH
As Double, ByVal angV As Double, ByVal idsondeo As String, ByVal txtvolinf As
Double, ByVal txtvolsup As Double, ByVal enlace As String)
        Dim myDWG As ApplicationServices.Document
        Dim myDB As DatabaseServices.Database
        Dim myEd As EditorInput.Editor
        Dim mytransman As DatabaseServices.TransactionManager
        Dim mytrans As DatabaseServices.Transaction
        Dim myBT As DatabaseServices.BlockTable
        Dim myBTR As DatabaseServices.BlockTableRecord
        Dim retIDColl As New ObjectIdCollection
        myDWG =
ApplicationServices.Application.DocumentManager.MdiActiveDocument
        myDWG.LockDocument(DocumentLockMode.AutoWrite, Nothing, Nothing, True)
        myDB = myDWG.Database
        myEd = myDWG.Editor
        mytransman = myDWG.TransactionManager
        mytrans = mytransman.StartTransaction
        myBT =
myDWG.Database.BlockTableId.GetObject(DatabaseServices.OpenMode.ForRead)
        myBTR =
myBT(DatabaseServices.BlockTableRecord.ModelSpace).GetObject(DatabaseServices.O
penMode.ForWrite)

        Dim pto1(0 To 2) As Double
        Dim pto2(0 To 2) As Double
        Dim pto3(0 To 2) As Double
        Dim pto4(0 To 2) As Double
        Dim pto5(0 To 2) As Double
        Dim scpejex(0 To 2), scpejey(0 To 2) As Double
        Dim dr As Double
        Dim L1 As Autodesk.AutoCAD.DatabaseServices.Line

        Dim pi As Double
        pi = 3.14159265

        angH = angH * pi / 180
        angV = angV * pi / 180

        dr = dis * Math.Cos(angV) / 100
        pto2(0) = Math.Sin(angH) * dr + p1(0)
        pto2(1) = Math.Cos(angH) * dr + p1(1)
        pto2(2) = p1(2) - (Math.Sin(angV) * dis) / 100
        pto3(0) = p1(0) : pto3(1) = p1(1) : pto3(2) = p1(2)
        pto4(0) = pto2(0) : pto4(1) = pto2(1) : pto4(2) = pto2(2)
```

```
        pto5(0) = p1(0) : pto5(1) = p1(1) : pto5(2) = p1(2)

        L1 = New DatabaseServices.Line(New Point3d(p1(0), p1(1), p1(2)), New
Point3d(pto2(0), pto2(1), pto2(2)))

        myBTR.AppendEntity(L1)
        mytrans.AddNewlyCreatedDBObject(L1, True)

        'asignar color AZUL a sin inyectar
        'asigna color ROJO a inyectado
        'Comprobar si tiene inyección y un volumen > 5
        Dim dbconnection As OleDbConnection
        Dim dbdataset As DataSet
        Dim dbdataadapter As OleDbDataAdapter
        Dim cadenaconexion As String
        Dim archivodatos As String
        Dim nombretabla As String
        Dim camino As String
        Dim sentencia As String

        archivodatos = "SONDEOS.MDB"
        camino = "C:\inaki\0308SIG\DATOS"
        sentencia = "SELECT * FROM tbl_Sondeos"
        nombretabla = "Con_Sondeos"
        cadenaconexion = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" &
camino & "\" & archivodatos & ";"
        dbconnection = New OleDbConnection(cadenaconexion)
        dbconnection.Open()

        sentencia = "SELECT * FROM tbl_inyecciones WHERE idinyeccion='" &
idsondeo & _
        "' and VOLUMEN>=" & txtvolinf & " and VOLUMEN<" & txtvolsup

        dbdataset = New DataSet
        dbdataadapter = New OleDbDataAdapter(sentencia, dbconnection)
        Dim commandbuilder As New OleDbCommandBuilder(dbdataadapter)
        dbdataadapter.MissingSchemaAction = MissingSchemaAction.AddWithKey
        dbdataadapter.Fill(dbdataset, nombretabla)

        If dbdataset.Tables(0).Rows.Count = 1 Then L1.ColorIndex = 4 Else
L1.ColorIndex = 1

        'crear enlace con la plantilla ENSAYOS, idensayo.
        If enlace = "BAI" Then

            Dim idensayo As String
            idensayo = LTrim(RTrim(idsondeo))

            'enlazar valor
            Dim conjplanvin As CAO.LinkTemplates
            Dim planvinculo As CAO.LinkTemplate
            Dim conex As New CAO.DbConnect
            Dim keyvalor As CAO.KeyValues
            Dim kvs As CAO.KeyValue
            Dim alink As CAO.Link
            Dim objcode As Autodesk.AutoCAD.DatabaseServices.ObjectId
            'Establecer plantilla de vínculo
            conjplanvin = conex.GetLinkTemplates(0)
            planvinculo = conjplanvin.Item("ENSAYOS")

            'Establecer el registro para vincular

            keyvalor = AcadApplication.GetInterfaceObject("CAO.KeyValues.16")
            kvs = keyvalor.Add("IDENSAYO", idensayo)
```

```
            'Enlaces a los objetos seleccionados
            objcode = L1.ObjectId

            'Crear un vinculo al objeto seleccionado
            alink = planvinculo.CreateLink(objcode.OldId, keyvalor)
        End If

        mytrans.Commit()
        dbconnection.Close()
        mytrans.Dispose()
        mytransman.Dispose()

    End Sub

    Sub construircirculos(ByVal p1() As Double, ByVal dis As Double, ByVal angH
As Double, ByVal angV As Double, ByVal idsondeo As String, ByVal txtvolinf As
Double, ByVal txtvolsup As Double, ByVal enlace As String)
        Dim myDWG As ApplicationServices.Document
        Dim myDB As DatabaseServices.Database
        Dim myEd As EditorInput.Editor
        Dim mytransman As DatabaseServices.TransactionManager
        Dim mytrans As DatabaseServices.Transaction
        Dim myBT As DatabaseServices.BlockTable
        Dim myBTR As DatabaseServices.BlockTableRecord
        Dim retIDColl As New ObjectIdCollection
        myDWG =
ApplicationServices.Application.DocumentManager.MdiActiveDocument
        myDWG.LockDocument(DocumentLockMode.AutoWrite, Nothing, Nothing, True)
        myDB = myDWG.Database
        myEd = myDWG.Editor
        mytransman = myDWG.TransactionManager
        mytrans = mytransman.StartTransaction
        myBT =
myDWG.Database.BlockTableId.GetObject(DatabaseServices.OpenMode.ForRead)
        myBTR =
myBT(DatabaseServices.BlockTableRecord.ModelSpace).GetObject(DatabaseServices.O
penMode.ForWrite)

        Dim pto1(0 To 2) As Double
        Dim pto2(0 To 2) As Double
        Dim pto3(0 To 2) As Double
        Dim pto4(0 To 2) As Double
        Dim pto5(0 To 2) As Double
        Dim scpejex(0 To 2), scpejey(0 To 2) As Double
        Dim dr As Double
        Dim c1 As Autodesk.AutoCAD.DatabaseServices.Circle
        Dim pi As Double
        pi = 3.14159265

        angH = angH * pi / 180
        angV = angV * pi / 180

        dr = dis * Math.Cos(angV) / 100
        pto2(0) = Math.Sin(angH) * dr + p1(0)
        pto2(1) = Math.Cos(angH) * dr + p1(1)
        pto2(2) = p1(2) - (Math.Sin(angV) * dis) / 100
        pto3(0) = p1(0) : pto3(1) = p1(1) : pto3(2) = p1(2)
        pto4(0) = pto2(0) : pto4(1) = pto2(1) : pto4(2) = pto2(2)
        pto5(0) = p1(0) : pto5(1) = p1(1) : pto5(2) = p1(2)
```

```
        'asignar color AZUL a sin inyectar
        'asigna color ROJO a inyectado
        'Comprobar si tiene inyección y un volumen > 5
        Dim dbconnection As OleDbConnection
        Dim dbdataset As DataSet
        Dim dbdataadapter As OleDbDataAdapter
        Dim cadenaconexion As String
        Dim archivodatos As String
        Dim nombretabla As String
        Dim camino As String
        Dim sentencia As String

        archivodatos = "SONDEOS.MDB"
        camino = "C:\inaki\0308SIG\DATOS"
        sentencia = "SELECT * FROM tbl_Sondeos"
        nombretabla = "Con_Sondeos"
        cadenaconexion = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" &
camino & "\" & archivodatos & ";"
        dbconnection = New OleDbConnection(cadenaconexion)
        dbconnection.Open()

        sentencia = "SELECT * FROM tbl_inyecciones WHERE idinyeccion='" &
idsondeo & _
        "' and VOLUMEN>=" & txtvolinf & " and VOLUMEN<" & txtvolsup

        dbdataset = New DataSet
        dbdataadapter = New OleDbDataAdapter(sentencia, dbconnection)
        Dim commandbuilder As New OleDbCommandBuilder(dbdataadapter)
        dbdataadapter.MissingSchemaAction = MissingSchemaAction.AddWithKey
        dbdataadapter.Fill(dbdataset, nombretabla)
        Dim volum, radio As Single
        If dbdataset.Tables(0).Rows.Count = 1 Then
            volum = CSng(dbdataset.Tables(0).Rows(0).Item("Volumen").ToString)
            radio = ((volum / pi) ^ 0.5) / 7
            'radio = 0.03
            If radio < 0.2 Then radio = 0.2
            c1 = New DatabaseServices.Circle(New Point3d(p1(0), p1(1), p1(2)),
Vector3d.XAxis, radio)
            myBTR.AppendEntity(c1)
            mytrans.AddNewlyCreatedDBObject(c1, True)
            c1.ColorIndex = 4
        End If

        'crear enlace con la plantilla ENSAYOS, idensayo.
        If enlace = "BAI" Then

            Dim idensayo As String
            idensayo = LTrim(RTrim(idsondeo))

            'enlazar valor
            Dim conjplanvin As CAO.LinkTemplates
            Dim planvinculo As CAO.LinkTemplate
            Dim conex As New CAO.DbConnect
            Dim keyvalor As CAO.KeyValues
            Dim kvs As CAO.KeyValue
            Dim alink As CAO.Link
            Dim objcode As Autodesk.AutoCAD.DatabaseServices.ObjectId
            'Establecer plantilla de vínculo
            conjplanvin = conex.GetLinkTemplates(0)
            planvinculo = conjplanvin.Item("ENSAYOS")

            'Establecer el registro para vincular

            keyvalor = AcadApplication.GetInterfaceObject("CAO.KeyValues.16")
```

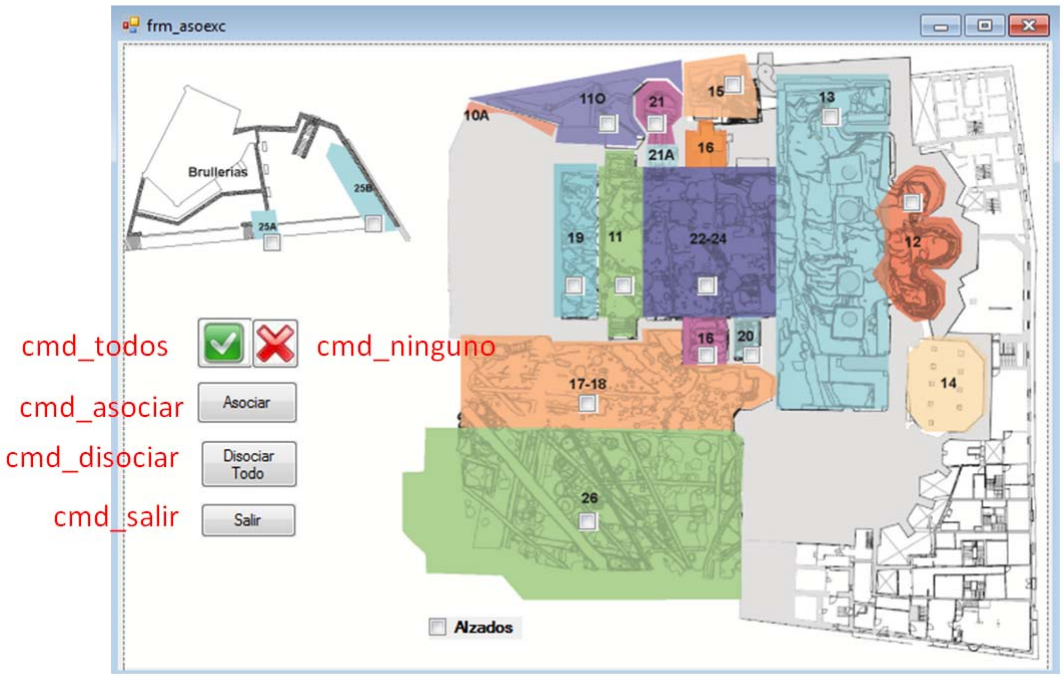
```
            kvs = keyvalor.Add("IDENSAYO", idensayo)

            'Enlaces a los objetos seleccionados
            objcode = c1.ObjectId

            'Crear un vinculo al objeto seleccionado
            alink = planvinculo.CreateLink(objcode.OldId, keyvalor)
        End If

        mytrans.Commit()
        dbconnection.Close()
        mytrans.Dispose()
        mytransman.Dispose()

    End Sub

    Sub calculoangulos(ByVal p1 As Point3d, ByVal p2 As Point3d)
        Dim vx12, vy12, vz12 As Double
        Dim disxy, disxz, disyz As Double
        disxy = Math.Sqrt(((p2(0) - p1(0)) ^ 2) + ((p2(1) - p1(1)) ^ 2))
        disxz = Math.Sqrt(((p2(0) - p1(0)) ^ 2) + ((p2(2) - p1(2)) ^ 2))
        disyz = Math.Sqrt(((p2(1) - p1(1)) ^ 2) + ((p2(2) - p1(2)) ^ 2))
        vx12 = p2.X - p1.X
        vy12 = p2.Y - p1.Y
        vz12 = p2.Z - p1.Z

        z_xy = (Math.Atan(vy12 / vx12))
        x_yz = (Math.Atan(vz12 / vy12))
        y_xz = Math.Atan(vz12 / disxy)

    End Sub

    Public Sub construircilindros(ByVal p1() As Double, ByVal dis As Double,
ByVal angH As Double, ByVal angV As Double, ByVal idsondeo As String, ByVal
txtvolinf As Double, ByVal txtvolsup As Double, ByVal enlace As String)
        Dim pi As Double
        pi = 3.1415926535
        Dim myDWG As ApplicationServices.Document
        Dim myDB As DatabaseServices.Database
        Dim myEd As EditorInput.Editor
        Dim mytransman As DatabaseServices.TransactionManager
        Dim mytrans As DatabaseServices.Transaction
        Dim myBT As DatabaseServices.BlockTable
        Dim myBTR As DatabaseServices.BlockTableRecord
        Dim retIDColl As New ObjectIdCollection
        myDWG =
ApplicationServices.Application.DocumentManager.MdiActiveDocument
        myDWG.LockDocument(DocumentLockMode.AutoWrite, Nothing, Nothing, True)
        myDB = myDWG.Database
        myEd = myDWG.Editor
        mytransman = myDWG.TransactionManager
        mytrans = mytransman.StartTransaction
        myBT =
myDWG.Database.BlockTableId.GetObject(DatabaseServices.OpenMode.ForRead)
        myBTR =
myBT(DatabaseServices.BlockTableRecord.ModelSpace).GetObject(DatabaseServices.O
penMode.ForWrite)


        Dim pto1(0 To 2) As Double
        pto1(0) = p1(0)
        pto1(1) = p1(1)
        pto1(2) = p1(2)
```

```
        Dim p2 As New Point3d
        x_yz = 1 : y_xz = 1 : z_xy = 1

        Dim pto2(0 To 2) As Double
        Dim dr As Double

        angH = angH * pi / 180
        angV = angV * pi / 180
        dr = dis * Math.Cos(angV) / 100
        pto2(0) = Math.Sin(angH) * dr + pto1(0)
        pto2(1) = Math.Cos(angH) * dr + pto1(1)
        pto2(2) = pto1(2) - (Math.Sin(angV) * dis) / 100

        Dim pmedio(0 To 2) As Double
        pmedio(0) = (pto1(0) + pto2(0)) / 2
        pmedio(1) = (pto1(1) + pto2(1)) / 2
        pmedio(2) = (pto1(2) + pto2(2)) / 2


        Dim acSol3DCyl1 As Solid3d = New Solid3d()
        acSol3DCyl1.SetDatabaseDefaults()

        acSol3DCyl1.CreateFrustum(dis / 100, 0.03, 0.03, 0.03)
        Dim vvv As New Vector3d(-(pto2(1) - pto1(1)), (pto2(0) - pto1(0)), 0)

        acSol3DCyl1.ColorIndex = 3
        acSol3DCyl1.TransformBy(Matrix3d.Rotation(pi / 2, Vector3d.XAxis, New
Point3d(0, 0, 0)))
        acSol3DCyl1.TransformBy(Matrix3d.Rotation(2 * pi - angH,
Vector3d.ZAxis, New Point3d(0, 0, 0)))
        acSol3DCyl1.TransformBy(Matrix3d.Rotation(angV, vvv, New Point3d(0, 0,
0)))
        acSol3DCyl1.TransformBy(Matrix3d.Displacement(New Point3d(pmedio(0),
pmedio(1), pmedio(2)) - Point3d.Origin))


        myBTR.AppendEntity(acSol3DCyl1)
        mytrans.AddNewlyCreatedDBObject(acSol3DCyl1, True)


        'asignar color AZUL a sin inyectar
        'asigna color ROJO a inyectado
        'Comprobar si tiene inyección y un volumen > 5
        Dim dbconnection As OleDbConnection
        Dim dbdataset As DataSet
        Dim dbdataadapter As OleDbDataAdapter
        Dim cadenaconexion As String
        Dim archivodatos As String
        Dim nombretabla As String
        Dim camino As String
        Dim sentencia As String

        archivodatos = "SONDEOS.MDB"
        camino = "C:\inaki\0308SIG\DATOS"
        sentencia = "SELECT * FROM tbl_Sondeos"
        nombretabla = "Con_Sondeos"
        cadenaconexion = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" &
camino & "\" & archivodatos & ";"
        dbconnection = New OleDbConnection(cadenaconexion)
        dbconnection.Open()

        sentencia = "SELECT * FROM tbl_inyecciones WHERE idinyeccion='" &
idsondeo & _
        "' and VOLUMEN>=" & txtvolinf & " and VOLUMEN<" & txtvolsup
```

```
        dbdataset = New DataSet
        dbdataadapter = New OleDbDataAdapter(sentencia, dbconnection)
        Dim commandbuilder As New OleDbCommandBuilder(dbdataadapter)
        dbdataadapter.MissingSchemaAction = MissingSchemaAction.AddWithKey
        dbdataadapter.Fill(dbdataset, nombretabla)

        If dbdataset.Tables(0).Rows.Count = 1 Then acSol3DCyl1.ColorIndex = 4
Else acSol3DCyl1.ColorIndex = 1

        'crear enlace con la plantilla ENSAYOS, idensayo.
        If enlace = "BAI" Then

            Dim idensayo As String
            idensayo = LTrim(RTrim(idsondeo))

            'enlazar valor
            Dim conjplanvin As CAO.LinkTemplates
            Dim planvinculo As CAO.LinkTemplate
            Dim conex As New CAO.DbConnect
            Dim keyvalor As CAO.KeyValues
            Dim kvs As CAO.KeyValue
            Dim alink As CAO.Link
            Dim objcode As Autodesk.AutoCAD.DatabaseServices.ObjectId
            'Establecer plantilla de vínculo
            conjplanvin = conex.GetLinkTemplates(0)
            planvinculo = conjplanvin.Item("ENSAYOS")

            'Establecer el registro para vincular

            keyvalor = AcadApplication.GetInterfaceObject("CAO.KeyValues.16")
            kvs = keyvalor.Add("IDENSAYO", idensayo)

            'Enlaces a los objetos seleccionados
            objcode = acSol3DCyl1.ObjectId

            'Crear un vinculo al objeto seleccionado
            alink = planvinculo.CreateLink(objcode.OldId, keyvalor)
        End If



        mytrans.Commit()
        mytrans.Dispose()
        mytransman.Dispose()

    End Sub

End Module
```

7.8 Código fuente comentado del programa AsoExc

frm_asoexc



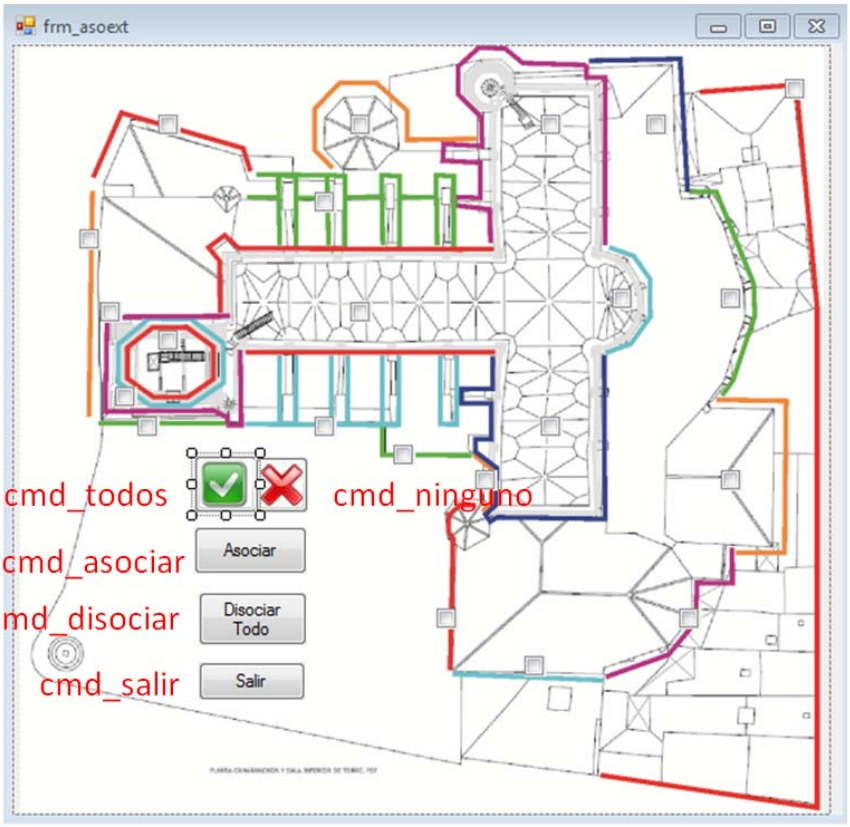

```
Imports Autodesk.Gis.Map
Imports Autodesk.Gis.Map.DisplayManagement
Imports Autodesk.Gis.Map.Project
Imports Autodesk.AutoCAD.Publishing

Imports Autodesk.AutoCAD.Runtime
Imports Autodesk.AutoCAD
Imports Autodesk.AutoCAD.DatabaseServices
Imports Autodesk.AutoCAD.EditorInput
Imports Autodesk.AutoCAD.Geometry
Imports Autodesk.AutoCAD.ApplicationServices
Imports Autodesk.AutoCAD.ApplicationServices.Application
Imports Autodesk.AutoCAD.LayerManager
Imports Autodesk.AutoCAD.Windows

Imports System.Data.OleDb
Imports System.Console
Imports System.Collections
Imports System.Windows
Imports System
Imports Autodesk.AutoCAD.Colors
Public Class frm_asoexc

    'Este codigo ha sido diseñado por Iñaki Koroso para el Sistema de Información Monumental (SIM)
    'de la Catedral de Santa María de Vitoria-Gasteiz.
    'El autor agradece a la comunidad de desarrolladores (foros, publicaciones, blogs) el soporte
    'suministrado en el desarrollo del programa. En sintonía con este planteamiento se permite la
    'reproducción de una parte o el total del código siempre que el resultado se comparta de igual
    'manera con el resto de la comunidad de desarrolladores


    Private Sub cmd_todos_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cmd_todos.Click
        c_11.Checked = True
```

```
        c_11O.Checked = True
        c_12.Checked = True
        c_13.Checked = True
        c_14.Checked = True
        c_15.Checked = True
        c_16.Checked = True
        c_1718.Checked = True
        c_19.Checked = True
        c_20.Checked = True
        c_21.Checked = True
        c_2224.Checked = True
        c_25a.Checked = True
        c_25.Checked = True
        c_26.Checked = True
        c_alzados.Checked = True
    End Sub

    Private Sub cmd_ninguno_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cmd_ninguno.Click
        c_11.Checked = False
        c_11O.Checked = False
        c_12.Checked = False
        c_13.Checked = False
        c_14.Checked = False
        c_15.Checked = False
        c_16.Checked = False
        c_1718.Checked = False
        c_19.Checked = False
        c_20.Checked = False
        c_21.Checked = False
        c_2224.Checked = False
        c_25a.Checked = False
        c_25.Checked = False
        c_26.Checked = False
        c_alzados.Checked = False
    End Sub

    Private Sub cmd_asociar_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cmd_asociar.Click
        Dim dibujo As Document
        dibujo = Application.DocumentManager.MdiActiveDocument
        ' Declaracion de elementos necesarios
        Dim mapApi As MapApplication =
Autodesk.Gis.Map.HostMapApplicationServices.Application
        Dim proj As ProjectModel = mapApi.ActiveProject
        Dim drawingSet As DrawingSet = proj.DrawingSet
        Dim dwgdrawing As AttachedDrawing
        ' Recorrido por todas las casillas de verificación y asociar dibujos
correspondientes
        If c_11.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGEXC:\zona11.DWG")
        If c_11O.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGEXC:\zona11o.DWG")
        If c_12.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGEXC:\zona12.DWG")
        If c_13.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGEXC:\zona13.DWG")
        If c_15.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGEXC:\zona15.DWG")
        If c_16.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGEXC:\z16.DWG")
        If c_1718.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGEXC:\zona17_18.DWG")
```

```
        If c_19.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGEXC:\zona19.DWG")
        If c_20.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGEXC:\z20.DWG")
        If c_21.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGEXC:\zona21.DWG")
        If c_2224.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGEXC:\zona22.DWG")
        If c_25a.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGEXC:\z25a.DWG")
        If c_25.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGEXC:\z25b.DWG")
        If c_26.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGEXC:\sector26.DWG")
        If c_alzados.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGEXC:\Alzado_subsuelo.DWG")
        dibujo.Editor.Regen()
    End Sub

    Private Sub cmd_disociar_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cmd_disociar.Click
        Dim dibujo As Document
        dibujo = Application.DocumentManager.MdiActiveDocument

        Dim mapApi As MapApplication =
Autodesk.Gis.Map.HostMapApplicationServices.Application
        Dim proj As ProjectModel = mapApi.ActiveProject
        Dim drawingSet As DrawingSet = proj.DrawingSet
        Dim attdraw As AttachedDrawing

        For x = 0 To drawingSet.AllDrawingsCount - 1
            attdraw = drawingSet.AllAttachedDrawings.Item(0)
            drawingSet.DetachDrawing(attdraw.ObjectId)
        Next x
        dibujo.Editor.Regen()
    End Sub

    Private Sub cmd_salir_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cdm_salir.Click
        Me.Close()
    End Sub

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

    End Sub
End Class
```

7.9 Código fuente comentado del programa AsoExt

frm_asoext



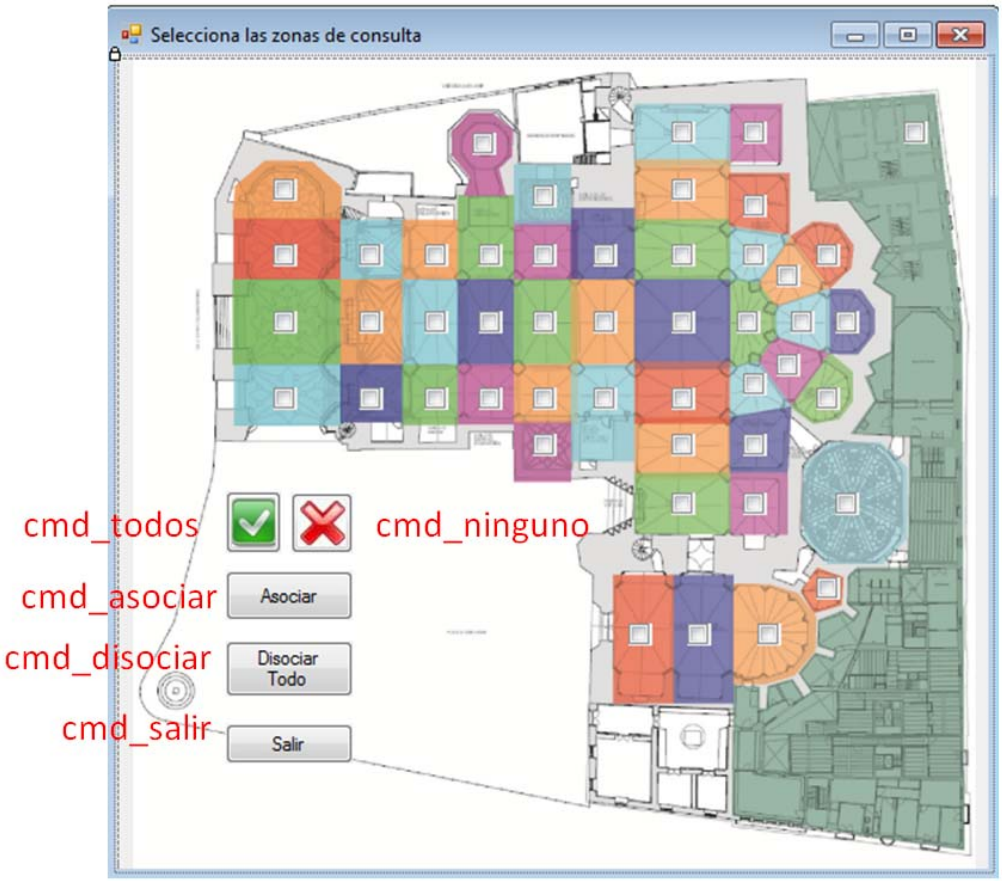

```
Imports Autodesk.Gis.Map
Imports Autodesk.Gis.Map.DisplayManagement
Imports Autodesk.Gis.Map.Project
Imports Autodesk.AutoCAD.Publishing
Imports Autodesk.AutoCAD.Runtime
Imports Autodesk.AutoCAD
Imports Autodesk.AutoCAD.DatabaseServices
Imports Autodesk.AutoCAD.EditorInput
Imports Autodesk.AutoCAD.Geometry
Imports Autodesk.AutoCAD.ApplicationServices
Imports Autodesk.AutoCAD.ApplicationServices.Application
Imports Autodesk.AutoCAD.LayerManager
Imports Autodesk.AutoCAD.Windows
Imports System.Data.OleDb
Imports System.Console
Imports System.Collections
Imports System.Windows
Imports System
Imports Autodesk.AutoCAD.Colors
Public Class frm_asoext

    Private Sub cmd_todos_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cmd_todos.Click
        c_ptn.Checked = True
        c_ptw.Checked = True
        c_pts.Checked = True
```

```
        c_trft.Checked = True
        c_trch.Checked = True
        c_trov.Checked = True
        c_cngh.Checked = True
        c_nn.Checked = True
        c_nc.Checked = True
        c_ns.Checked = True
        c_csfg.Checked = True
        c_ana.Checked = True
        c_syw.Checked = True
        c_sys.Checked = True
        c_syab.Checked = True
        c_tns.Checked = True
        c_tnn.Checked = True
        c_tcn.Checked = True
        c_tpb.Checked = True
        c_abs.Checked = True
        c_scr.Checked = True
        c_kutx.Checked = True
    End Sub

    Private Sub cmd_ninguno_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cmd_ninguno.Click
        c_ptn.Checked = False
        c_ptw.Checked = False
        c_pts.Checked = False
        c_trft.Checked = False
        c_trch.Checked = False
        c_trov.Checked = False
        c_cngh.Checked = False
        c_nn.Checked = False
        c_nc.Checked = False
        c_ns.Checked = False
        c_csfg.Checked = False
        c_ana.Checked = False
        c_syw.Checked = False
        c_sys.Checked = False
        c_syab.Checked = False
        c_tns.Checked = False
        c_tnn.Checked = False
        c_tcn.Checked = False
        c_tpb.Checked = False
        c_abs.Checked = False
        c_scr.Checked = False
        c_kutx.Checked = False
    End Sub

    Private Sub cmd_asociar_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cmd_asociar.Click
        ' Recorrido por todas las casillas de verificación y asociar dibujos
correspondientes
        Dim dibujo As Document
        dibujo = Application.DocumentManager.MdiActiveDocument
        Dim mapApi As MapApplication =
Autodesk.Gis.Map.HostMapApplicationServices.Application
        Dim proj As ProjectModel = mapApi.ActiveProject
        Dim drawingSet As DrawingSet = proj.DrawingSet
        Dim dwgdrawing As AttachedDrawing
        If c_ptn.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGEXT:\PTN-E.DWG")
        If c_ptw.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGEXT:\PTW-E.DWG")
        If c_pts.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGEXT:\PTS-E.DWG")
```

```
        If c_trft.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGEXT:\TRFT.DWG")
        If c_trch.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGEXT:\TRCH.DWG")
        If c_trov.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGEXT:\TROV.DWG")
        If c_cngh.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGEXT:\CNGH-E.DWG")
        If c_nn.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGEXT:\NN-E.DWG")
        If c_nc.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGEXT:\NC-E.DWG")
        If c_ns.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGEXT:\NS-E.DWG")
        If c_csfg.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGEXT:\CSFG-E.DWG")
        If c_ana.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGEXT:\ANA-E.DWG")
        If c_syw.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGEXT:\SYW-E.DWG")
        If c_sys.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGEXT:\SYS-E.DWG")
        If c_syab.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGEXT:\SYAB-E.DWG")
        If c_tns.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGEXT:\TNS-E.DWG")
        If c_tnn.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGEXT:\TNN-E.DWG")
        If c_tcn.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGEXT:\TCN-E.DWG")
        If c_tpb.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGEXT:\TPB-E.DWG")
        If c_abs.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGEXT:\ABS-E.DWG")
        If c_scr.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGEXT:\SCR-E.DWG")
        If c_kutx.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGEXT:\KUTX.DWG")
        dibujo.Editor.Regen()
    End Sub

    Private Sub cmd_disociar_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cmd_disociar.Click
        'Se disocian todos los dibujos enlazados
        Dim dibujo As Document
        dibujo = Application.DocumentManager.MdiActiveDocument
        Dim mapApi As MapApplication =
Autodesk.Gis.Map.HostMapApplicationServices.Application
        Dim proj As ProjectModel = mapApi.ActiveProject
        Dim drawingSet As DrawingSet = proj.DrawingSet
        Dim attdraw As AttachedDrawing
        For x = 0 To drawingSet.AllDrawingsCount - 1
            attdraw = drawingSet.AllAttachedDrawings.Item(0)
            drawingSet.DetachDrawing(attdraw.ObjectId)
        Next x
        dibujo.Editor.Regen()
    End Sub

    Private Sub cmd_salir_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cmd_salir.Click
        Me.Close()
    End Sub
End Class
```

7.10 Código fuente comentado del programa Asoint

frm_asoint




```
 Private Sub cmd_todos_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cmd_todos.Click
        c_ptp2q4.Checked = True
        c_ptp3q7.Checked = True
        c_ptp1q3.Checked = True
        c_ptp1q2.Checked = True
        c_ptp2q4.Checked = True
        c_nni1j3.Checked = True
        c_nci1j2.Checked = True
        c_nsi2j4.Checked = True
        c_nnh1i3.Checked = True
        c_nch1i2.Checked = True
        c_nsh2i4.Checked = True
        c_xto.Checked = True
        c_nng1h3.Checked = True
        c_ncg1h2.Checked = True
        c_nsg2h4.Checked = True
        c_cnf3g5.Checked = True
        c_nnf1g3.Checked = True
        c_ncf1g2.Checked = True
        c_nsf2g4.Checked = True
        c_csf4g6.Checked = True
        c_nne1f3.Checked = True
        c_nce1f2.Checked = True
        c_nse2f4.Checked = True
        c_tnd5e7.Checked = True
```

```
        c_tnd3e5.Checked = True
        c_tnd1e3.Checked = True
        c_tcd1e2.Checked = True
        c_tsd2e4.Checked = True
        c_tsd4e6.Checked = True
        c_tsd6e8.Checked = True
        c_ctb5d7.Checked = True
        c_ctb3d5.Checked = True
        c_dbb3d3.Checked = True
        c_pbc1d2.Checked = True
        c_dbb4d4.Checked = True
        c_ctb4d6.Checked = True
        c_ctb6d8.Checked = True
        c_dbb1c3.Checked = True
        c_dbb1c2.Checked = True
        c_dbb2c4.Checked = True
        c_aba7b3.Checked = True
        c_aba1b2.Checked = True
        c_aba8b4.Checked = True
        c_scr.Checked = True
        c_9ay0b.Checked = True
        c_79y80.Checked = True
        c_17y28.Checked = True
        c_kutxi.Checked = True

    End Sub
    Friend WithEvents c_pbc1d2 As System.Windows.Forms.CheckBox

    Private Sub cmd_asociar_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cmd_Asociar.Click
        ' Recorrido por todas las casillas de verificación y asociar dibujos
correspondientes

        Dim dibujo As Document
        dibujo = Application.DocumentManager.MdiActiveDocument
        Dim mapApi As MapApplication =
Autodesk.Gis.Map.HostMapApplicationServices.Application
        Dim proj As ProjectModel = mapApi.ActiveProject
        Dim drawingSet As DrawingSet = proj.DrawingSet
        Dim dwgdrawing As AttachedDrawing
        If c_ptp2q4.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGINT:\ptp2q4.DWG")
        If c_ptp3q7.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGINT:\ptp3q7.DWG")
        If c_ptp1q3.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGINT:\ptp1q3.DWG")
        If c_ptp1q2.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGINT:\ptp1q2.DWG")
        If c_ptp2q4.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGINT:\ptp2q4.DWG")
        If c_nni1j3.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGINT:\nni1j3.DWG")
        If c_nci1j2.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGINT:\nci1j2.DWG")
        If c_nsi2j4.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGINT:\nsi2j4.DWG")
        If c_nnh1i3.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGINT:\nnh1i3.DWG")
        If c_nch1i2.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGINT:\nch1i2.DWG")
        If c_nsh2i4.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGINT:\nsh2i4.DWG")
        If c_xto.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGINT:\xto.DWG")
```

```
        If c_nng1h3.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGINT:\nng1h3.DWG")
        If c_ncg1h2.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGINT:\ncg1h2.DWG")
        If c_nsg2h4.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGINT:\nsg2h4.DWG")
        If c_cnf3g5.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGINT:\cnf3g5.DWG")
        If c_nnf1g3.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGINT:\nnf1g3.DWG")
        If c_ncf1g2.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGINT:\ncf1g2.DWG")
        If c_nsf2g4.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGINT:\nsf2g4.DWG")
        If c_csf4g6.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGINT:\csf4g6.DWG")
        If c_nne1f3.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGINT:\nne1f3.DWG")
        If c_nce1f2.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGINT:\nce1f2.DWG")
        If c_nse2f4.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGINT:\nse2f4.DWG")
        If c_tnd5e7.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGINT:\tnd5e7.DWG")
        If c_tnd3e5.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGINT:\tnd3e5.DWG")
        If c_tnd1e3.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGINT:\tnd1e3.DWG")
        If c_tcd1e2.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGINT:\tcd1e2.DWG")
        If c_tsd2e4.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGINT:\tsd2e4.DWG")
        If c_tsd4e6.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGINT:\tsd4e6.DWG")
        If c_tsd6e8.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGINT:\tsd6e8.DWG")
        If c_ctb5d7.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGINT:\ctb5d7.DWG")
        If c_ctb3d5.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGINT:\ctb3d5.DWG")
        If c_dbb3d3.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGINT:\dbb3d3.DWG")
        If c_pbc1d2.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGINT:\pbc1d2.DWG")
        If c_dbb4d4.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGINT:\dbb4d4.DWG")
        If c_ctb4d6.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGINT:\ctb4d6.DWG")
        If c_ctb6d8.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGINT:\ctb6d8.DWG")
        If c_dbb1c3.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGINT:\dbb1c3.DWG")
        If c_dbb1c2.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGINT:\dbb1c2.DWG")
        If c_dbb2c4.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGINT:\dbb2c4.DWG")
        If c_aba7b3.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGINT:\aba7b3.DWG")
        If c_aba1b2.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGINT:\aba1b2.DWG")
        If c_aba8b4.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGINT:\aba8b4.DWG")
        If c_scr.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGINT:\scr.DWG")
```

```
        If c_9ay0b.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGINT:\9ay0b.DWG")
        If c_79y80.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGINT:\79y80.DWG")
        If c_17y28.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGINT:\17y28.DWG")
        'If c_kutxi.Checked = True Then dwgdrawing =
drawingSet.AttachDrawing("SIGINT:\kutxi.DWG")

    End Sub

    Private Sub cmd_ninguno_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cmd_ninguno.Click
        c_ptp2q4.Checked = False
        c_ptp3q7.Checked = False
        c_ptp1q3.Checked = False
        c_ptp1q2.Checked = False
        c_ptp2q4.Checked = False
        c_nni1j3.Checked = False
        c_nci1j2.Checked = False
        c_nsi2j4.Checked = False
        c_nnh1i3.Checked = False
        c_nch1i2.Checked = False
        c_nsh2i4.Checked = False
        c_xto.Checked = False
        c_nng1h3.Checked = False
        c_ncg1h2.Checked = False
        c_nsg2h4.Checked = False
        c_cnf3g5.Checked = False
        c_nnf1g3.Checked = False
        c_ncf1g2.Checked = False
        c_nsf2g4.Checked = False
        c_csf4g6.Checked = False
        c_nne1f3.Checked = False
        c_nce1f2.Checked = False
        c_nse2f4.Checked = False
        c_tnd5e7.Checked = False
        c_tnd3e5.Checked = False
        c_tnd1e3.Checked = False
        c_tcd1e2.Checked = False
        c_tsd2e4.Checked = False
        c_tsd4e6.Checked = False
        c_tsd6e8.Checked = False
        c_ctb5d7.Checked = False
        c_ctb3d5.Checked = False
        c_dbb3d3.Checked = False
        c_pbc1d2.Checked = False
        c_dbb4d4.Checked = False
        c_ctb4d6.Checked = False
        c_ctb6d8.Checked = False
        c_dbb1c3.Checked = False
        c_dbb1c2.Checked = False
        c_dbb2c4.Checked = False
        c_aba7b3.Checked = False
        c_aba1b2.Checked = False
        c_aba8b4.Checked = False
        c_scr.Checked = False
        c_9ay0b.Checked = False
        c_79y80.Checked = False
        c_17y28.Checked = False
    End Sub

    Private Sub cmd_disociar_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cmd_disociar.Click
```

```
        'Se disocian todos los dibujos enlazados
        Dim dibujo As Document
        dibujo = Application.DocumentManager.MdiActiveDocument
        Dim mapApi As MapApplication =
Autodesk.Gis.Map.HostMapApplicationServices.Application
        Dim proj As ProjectModel = mapApi.ActiveProject
        Dim drawingSet As DrawingSet = proj.DrawingSet
        Dim attdraw As AttachedDrawing
        For x = 0 To drawingSet.AllDrawingsCount - 1
            attdraw = drawingSet.AllAttachedDrawings.Item(0)
            drawingSet.DetachDrawing(attdraw.ObjectId)
        Next x
    End Sub

    Protected Overrides Sub Finalize()
        MyBase.Finalize()
    End Sub

    Private Sub frm_asoint_Deactivate(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.Deactivate
        Dim dibujo As Document
        dibujo = Application.DocumentManager.MdiActiveDocument
        dibujo.Editor.UpdateScreen()
    End Sub
End Class
```

7.11 Código fuente comentado del programa ConSim


```
Imports Autodesk.Gis.Map
Imports Autodesk.Gis.Map.DisplayManagement
Imports Autodesk.Gis.Map.Project
Imports Autodesk.AutoCAD.Publishing
Imports Autodesk.AutoCAD.Runtime
Imports Autodesk.AutoCAD
Imports Autodesk.AutoCAD.DatabaseServices
Imports Autodesk.AutoCAD.EditorInput
Imports Autodesk.AutoCAD.Geometry
Imports Autodesk.AutoCAD.ApplicationServices
Imports Autodesk.AutoCAD.ApplicationServices.Application
Imports Autodesk.AutoCAD.LayerManager
Imports Autodesk.AutoCAD.Windows
Imports System.Data.OleDb
Imports System.Console
Imports System.Collections
Imports System.Windows
Imports System
Imports Autodesk.AutoCAD.Colors


Public Class frm_consim
    Public were As String
    Public planti As String
    Public camino_base As String


    Sub valorwere()
        'Determinar el valor1 para la sentencia where y la plantilla
        Dim dbconnection As OleDbConnection
        Dim dbdataset As DataSet
        Dim dbdataadapter As OleDbDataAdapter
        Dim cadenaconexion As String
        Dim archivodatos As String
        Dim nombretabla As String
        Dim sentencia As String

        'Consulta de valores a BD de soporte
        archivodatos = "SOPORTE.MDB"
        Dim camGEN, camSUP As String
```

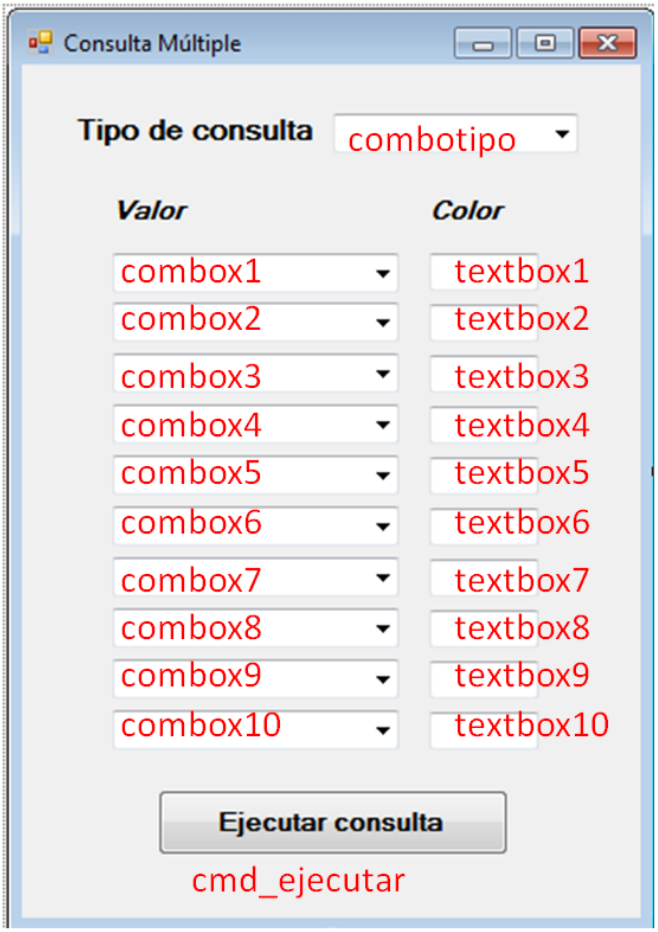
```
        camGEN =
Autodesk.AutoCAD.ApplicationServices.Application.Preferences.Files.AutoSavePath
.ToString
        camSUP =
Autodesk.AutoCAD.ApplicationServices.Application.Preferences.Files.TemplateDwgP
ath.ToString

        sentencia = "SELECT indice,ccampoclave,textonum,cplantilla FROM
conIndice"
        nombretabla = "Consulta"
        cadenaconexion = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" &
camSUP & "\" & archivodatos & ";"
        dbconnection = New OleDbConnection(cadenaconexion)
        dbconnection.Open()

        'Consulta de valores a la tabla correspondiente al objeto seleccionado
        dbdataset = New DataSet
        dbdataadapter = New OleDbDataAdapter(sentencia, dbconnection)
        dbdataadapter.Fill(dbdataset, "Consulta")
        dbconnection.Close()
        Dim tabla As System.Data.DataTable
        tabla = dbdataset.Tables("Consulta")
        Dim fila As DataRow
        Dim valor1 As String
        Dim tn As String
        Dim valor2 As String
        tn = "" : valor1 = ""
        For Each fila In tabla.Rows
            If fila.Item("indice") = cbx_tipo.Text Then
                valor1 = fila.Item("ccampoclave")
                tn = fila.Item("textonum")
                planti = fila.Item("cplantilla")
                Exit For
            End If
        Next
        'Creación de la consulta WHERE
        valor2 = cbx_valor.Text
        Dim lonvalor2 As Integer
        Dim valact, caracter As String
        were = "" : valact = ""
        lonvalor2 = Len(valor2)
        For x = 1 To lonvalor2
            caracter = Mid(valor2, x, 1)
            If caracter = "," Then
                If tn = "N" Then were = were & valor1 & " = " & valact & " OR "
                If tn = "T" Then were = were & valor1 & " = '" & valact & "' OR
"
                valact = ""
            Else
                valact = valact + caracter
            End If
        Next x
        If valact > "" Then
            If tn = "T" Then were = were & valor1 & " = '" & valact & "'"
            If tn = "N" Then were = were & valor1 & " = " & valact
        End If

    End Sub

    Private Sub frm_consim_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        'Asignación de valores a las listas desplegables
        Dim dbconnection As OleDbConnection
        Dim dbdataset As DataSet
```

```
        Dim dbdataadapter As OleDbDataAdapter
        Dim cadenaconexion As String
        Dim archivodatos As String
        Dim nombretabla As String
        Dim sentencia As String

        archivodatos = "SOPORTE.MDB"
        Dim camGEN, camSUP As String
        camGEN =
Autodesk.AutoCAD.ApplicationServices.Application.Preferences.Files.AutoSavePath
.ToString
        camSUP =
Autodesk.AutoCAD.ApplicationServices.Application.Preferences.Files.TemplateDwgP
ath.ToString
        sentencia = "SELECT * FROM conIndice ORDER BY indice"
        nombretabla = "Consulta"
        cadenaconexion = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" &
camSUP & "\" & archivodatos & ";"
        dbconnection = New OleDbConnection(cadenaconexion)
        dbconnection.Open()

        dbdataset = New DataSet
        dbdataadapter = New OleDbDataAdapter(sentencia, dbconnection)
        dbdataadapter.Fill(dbdataset, "Consulta")
        dbconnection.Close()
        Dim tabla As System.Data.DataTable
        tabla = dbdataset.Tables("Consulta")
        Dim fila As DataRow
        For Each fila In tabla.Rows
            Me.cbx_tipo.Items.Add(fila.Item("indice"))
        Next

    End Sub

    Private Sub cbx_tipo_SelectedIndexChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles cbx_tipo.SelectedIndexChanged
        'Actualización de valores en función del elemento de lista seleccionado
        Dim dbconnection As OleDbConnection
        Dim dbdataset As DataSet
        Dim dbdataadapter As OleDbDataAdapter
        Dim cadenaconexion As String
        Dim archivodatos As String
        Dim nombretabla As String
        Dim sentencia As String

        Dim camGEN, camSUP As String
        camGEN =
Autodesk.AutoCAD.ApplicationServices.Application.Preferences.Files.AutoSavePath
.ToString
        camSUP =
Autodesk.AutoCAD.ApplicationServices.Application.Preferences.Files.TemplateDwgP
ath.ToString

        'Consulta de los nombres de campos a utilizar
        archivodatos = "SOPORTE.MDB"
        sentencia = "SELECT * FROM conSecundario WHERE indice= '" &
cbx_tipo.Text & "'"
        nombretabla = "Consulta"
        cadenaconexion = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" &
camSUP & "\" & archivodatos & ";"
        dbconnection = New OleDbConnection(cadenaconexion)
        dbconnection.Open()
        dbdataset = New DataSet
        dbdataadapter = New OleDbDataAdapter(sentencia, dbconnection)
```

```
        dbdataadapter.Fill(dbdataset, "Consulta")
        dbconnection.Close()
        Dim tabla As System.Data.DataTable
        tabla = dbdataset.Tables("Consulta")
        Dim fila As DataRow
        Me.cbx_valor.Items.Clear()
        For Each fila In tabla.Rows
            Me.cbx_valor.Items.Add(fila.Item("secundario"))
        Next
    End Sub

    Private Sub cmd_ejecutar_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cmd_ejecutar.Click
        'definición de las variables
        Dim dibujo As Document
        dibujo = Application.DocumentManager.MdiActiveDocument
        Dim amap As MapApplication =
Autodesk.Gis.Map.HostMapApplicationServices.Application
        Dim prj As ProjectModel = amap.ActiveProject
        Dim qry As Query.QueryModel
        qry = prj.CurQuery

        'Definir la sentencia where y la plantilla de vinculo
        planti = ""
        valorwere()
        Dim sqlcon As New Query.SqlCondition
        sqlcon.WhereCondition = were
        sqlcon.LinkTemplate = planti
        Dim mainqrybr As New
Query.QueryBranch(Constants.JoinOperator.OperatorAnd)
        mainqrybr.AppendOperand(sqlcon)

        'asignar color
        Dim alteracion As PropertyAlterationDefinition
        alteracion = qry.PropertyAlteration
        alteracion.Clear()
        alteracion.AddAlteration(Constants.AlterationType.AlterationColor)
        alteracion.Item(0).Expression = txt_color.Text
        qry.EnablePropertyAlteration(True)
        qry.Define(mainqrybr)
        qry.Run()
        Me.Close()
    End Sub
End Class
```

7.12 Código fuente comentado del programa ConMul

frm_conmul




```
Imports Autodesk.Gis.Map
Imports Autodesk.Gis.Map.DisplayManagement
Imports Autodesk.Gis.Map.Project
Imports Autodesk.AutoCAD.Publishing
Imports Autodesk.AutoCAD.Runtime
Imports Autodesk.AutoCAD
Imports Autodesk.AutoCAD.DatabaseServices
Imports Autodesk.AutoCAD.EditorInput
Imports Autodesk.AutoCAD.Geometry
Imports Autodesk.AutoCAD.ApplicationServices
Imports Autodesk.AutoCAD.ApplicationServices.Application
Imports Autodesk.AutoCAD.LayerManager
Imports Autodesk.AutoCAD.Windows
Imports System.Data.OleDb
Imports System.Console
Imports System.Collections
Imports System.Windows
Imports System
Imports Autodesk.AutoCAD.Colors
Public Class frm_conmul
    Public were As String
    Public planti As String
    Public comboClave As String
    Public comboColor As String
```

```
    Public camino_base As String

    Sub valorwere()
        'Determinar el valor1 para la sentencia where y la plantilla
        Dim dbconnection As OleDbConnection
        Dim dbdataset As DataSet
        Dim dbdataadapter As OleDbDataAdapter
        Dim cadenaconexion As String
        Dim archivodatos As String
        Dim nombretabla As String
        Dim sentencia As String

        'Consulta de valores a BD de soporte
        archivodatos = "SOPORTE.MDB"
        Dim camGEN, camSUP As String
        camGEN =
Autodesk.AutoCAD.ApplicationServices.Application.Preferences.Files.AutoSavePath
.ToString
        camSUP =
Autodesk.AutoCAD.ApplicationServices.Application.Preferences.Files.TemplateDwgP
ath.ToString
        sentencia = "SELECT indice,ccampoclave,textonum,cplantilla FROM
conIndice"
        nombretabla = "Consulta"
        cadenaconexion = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" &
camSUP & "\" & archivodatos & ";"
        dbconnection = New OleDbConnection(cadenaconexion)
        dbconnection.Open()

        'Consulta de valores a la tabla correspondiente al objeto seleccionado
        dbdataset = New DataSet
        dbdataadapter = New OleDbDataAdapter(sentencia, dbconnection)
        dbdataadapter.Fill(dbdataset, "Consulta")
        dbconnection.Close()
        Dim tabla As System.Data.DataTable
        tabla = dbdataset.Tables("Consulta")
        Dim fila As DataRow
        Dim valor1 As String
        Dim tn As String
        Dim valor2 As String
        tn = "" : valor1 = ""
        For Each fila In tabla.Rows
            If fila.Item("indice") = comboTipo.Text Then
                valor1 = fila.Item("ccampoclave")
                tn = fila.Item("textonum")
                planti = fila.Item("cplantilla")
                Exit For
            End If
        Next

        'Creación de la consulta WHERE
        valor2 = comboClave
        Dim lonvalor2 As Integer
        Dim valact, caracter As String
        were = "" : valact = ""
        lonvalor2 = Len(valor2)
        For x = 1 To lonvalor2
            caracter = Mid(valor2, x, 1)
            If caracter = "," Then
                If tn = "N" Then were = were & valor1 & " = " & valact & " OR "
                If tn = "T" Then were = were & valor1 & " = '" & valact & "' OR
"
                valact = ""
            Else
```

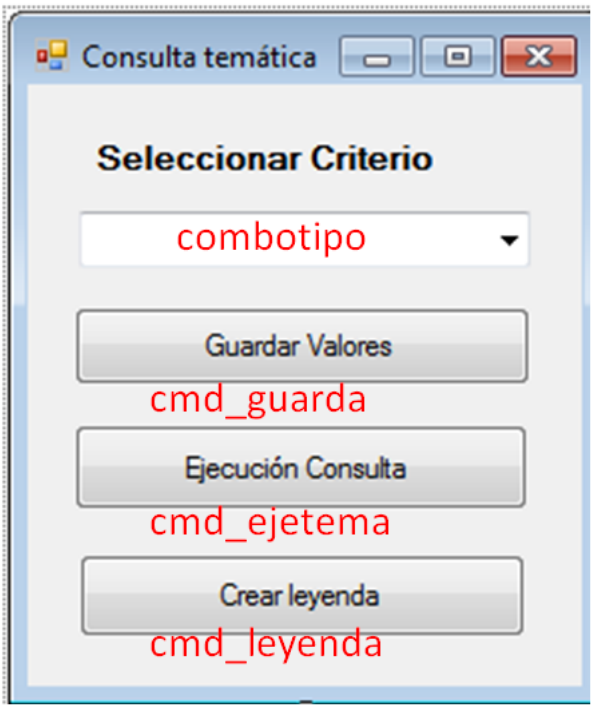
```
                valact = valact + caracter
            End If
        Next x
        If valact > "" Then
            If tn = "T" Then were = were & valor1 & " = '" & valact & "'"
            If tn = "N" Then were = were & valor1 & " = " & valact
        End If
    End Sub

    Private Sub frm_conmul_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        'Asignación de valores a las listas desplegables
        Dim dbconnection As OleDbConnection
        Dim dbdataset As DataSet
        Dim dbdataadapter As OleDbDataAdapter
        Dim cadenaconexion As String
        Dim archivodatos As String
        Dim nombretabla As String
        Dim sentencia As String

        Dim camGEN, camSUP As String
        camGEN =
Autodesk.AutoCAD.ApplicationServices.Application.Preferences.Files.AutoSavePath
.ToString
        camSUP =
Autodesk.AutoCAD.ApplicationServices.Application.Preferences.Files.TemplateDwgP
ath.ToString

        archivodatos = "SOPORTE.MDB"
        sentencia = "SELECT * FROM conIndice ORDER BY indice"
        nombretabla = "Consulta"
        cadenaconexion = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" &
camSUP & "\" & archivodatos & ";"
        dbconnection = New OleDbConnection(cadenaconexion)
        dbconnection.Open()

        dbdataset = New DataSet
        dbdataadapter = New OleDbDataAdapter(sentencia, dbconnection)
        dbdataadapter.Fill(dbdataset, "Consulta")
        dbconnection.Close()
        Dim tabla As System.Data.DataTable
        tabla = dbdataset.Tables("Consulta")
        Dim fila As DataRow
        For Each fila In tabla.Rows
            Me.comboTipo.Items.Add(fila.Item("indice"))
        Next
    End Sub

    Private Sub comboTipo_SelectedIndexChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles comboTipo.SelectedIndexChanged
        'Actualización de valores en función del elemento de lista seleccionado
        Dim dbconnection As OleDbConnection
        Dim dbdataset As DataSet
        Dim dbdataadapter As OleDbDataAdapter
        Dim cadenaconexion As String
        Dim archivodatos As String
        Dim nombretabla As String
        Dim sentencia As String

        'Consulta de los nombres de campos a utilizar
        archivodatos = "SOPORTE.MDB"
        Dim camGEN, camSUP As String
```

```
        camGEN =
Autodesk.AutoCAD.ApplicationServices.Application.Preferences.Files.AutoSavePath
.ToString
        camSUP =
Autodesk.AutoCAD.ApplicationServices.Application.Preferences.Files.TemplateDwgP
ath.ToString
        sentencia = "SELECT * FROM conSecundario WHERE indice= '" &
ComboBox1.Text & "'"
        nombretabla = "Consulta"
        cadenaconexion = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" &
camSUP & "\" & archivodatos & ";"
        dbconnection = New OleDbConnection(cadenaconexion)
        dbconnection.Open()
        dbdataset = New DataSet
        dbdataadapter = New OleDbDataAdapter(sentencia, dbconnection)
        dbdataadapter.Fill(dbdataset, "Consulta")
        dbconnection.Close()

        Dim tabla As System.Data.DataTable
        tabla = dbdataset.Tables("Consulta")
        Dim fila As DataRow

        Me.ComboBox1.Items.Clear()
        Me.ComboBox2.Items.Clear()
        Me.ComboBox3.Items.Clear()
        Me.ComboBox4.Items.Clear()
        Me.ComboBox5.Items.Clear()
        Me.ComboBox6.Items.Clear()
        Me.ComboBox7.Items.Clear()
        Me.ComboBox8.Items.Clear()
        Me.ComboBox9.Items.Clear()
        Me.ComboBox10.Items.Clear()
        For Each fila In tabla.Rows
            Me.ComboBox1.Items.Add(fila.Item("secundario"))
            Me.ComboBox2.Items.Add(fila.Item("secundario"))
            Me.ComboBox3.Items.Add(fila.Item("secundario"))
            Me.ComboBox4.Items.Add(fila.Item("secundario"))
            Me.ComboBox5.Items.Add(fila.Item("secundario"))
            Me.ComboBox6.Items.Add(fila.Item("secundario"))
            Me.ComboBox7.Items.Add(fila.Item("secundario"))
            Me.ComboBox8.Items.Add(fila.Item("secundario"))
            Me.ComboBox9.Items.Add(fila.Item("secundario"))
            Me.ComboBox10.Items.Add(fila.Item("secundario"))
        Next
    End Sub

    Private Sub cmd_ejecutar_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cmd_ejecutar.Click
        'definición de las variables
        Dim dibujo As Document
        dibujo = Application.DocumentManager.MdiActiveDocument
        Dim amap As MapApplication =
Autodesk.Gis.Map.HostMapApplicationServices.Application
        Dim prj As ProjectModel = amap.ActiveProject
        Dim qry As Query.QueryModel
        qry = prj.CurQuery


        'asignacion de los criterios de consulta
        Dim valores(0 To 10) As String
        Dim kontador As Integer
        kontador = 0
        valores(1) = ComboBox1.Text : valores(2) = ComboBox2.Text : valores(3)
= ComboBox3.Text
```

```
        valores(4) = ComboBox4.Text : valores(5) = ComboBox5.Text : valores(6)
= ComboBox6.Text
        valores(7) = ComboBox7.Text : valores(8) = ComboBox8.Text : valores(9)
= ComboBox9.Text
        valores(10) = ComboBox10.Text

        For x = 1 To 10
            If Len(valores(x)) > 0 Then
                kontador = kontador + 1
            Else
                Exit For
            End If
        Next x

        'asignación de colores
        Dim colores(0 To 10) As String
        colores(1) = TextBox1.Text : colores(2) = TextBox2.Text : colores(3) =
TextBox3.Text
        colores(4) = TextBox4.Text : colores(5) = TextBox5.Text : colores(6) =
TextBox6.Text
        colores(7) = TextBox7.Text : colores(8) = TextBox8.Text : colores(9) =
TextBox9.Text
        colores(10) = TextBox10.Text


        Dim sqlcon As New Query.SqlCondition
        Dim mainqrybr As New
Query.QueryBranch(Constants.JoinOperator.OperatorAnd)
        Dim alteracion As PropertyAlterationDefinition

        For kont = 1 To kontador
            qry.Clear()
            comboClave = valores(kont)
            comboColor = colores(kont)
            'Definir la sentencia where y la plantilla de vinculo
            planti = ""
            valorwere()

            sqlcon.WhereCondition = were
            sqlcon.LinkTemplate = planti

            mainqrybr.Clear()
            mainqrybr.AppendOperand(sqlcon)


            'asignar color
            alteracion = qry.PropertyAlteration
            alteracion.Clear()
            alteracion.AddAlteration(Constants.AlterationType.AlterationColor)
            alteracion.Item(0).Expression = colores(kont)
            qry.EnablePropertyAlteration(True)

            qry.Define(mainqrybr)

            qry.Run()

        Next kont
        Me.Refresh()
        Me.Close()
    End Sub
End Class
```

7.13 Código fuente comentado del programa ConTem

frm_contem




```
Imports Autodesk.Gis.Map
Imports Autodesk.Gis.Map.DisplayManagement
Imports Autodesk.Gis.Map.Project
Imports Autodesk.AutoCAD.Publishing
Imports Autodesk.AutoCAD.Runtime
Imports Autodesk.AutoCAD
Imports Autodesk.AutoCAD.DatabaseServices
Imports Autodesk.AutoCAD.EditorInput
Imports Autodesk.AutoCAD.Geometry
Imports Autodesk.AutoCAD.ApplicationServices
Imports Autodesk.AutoCAD.ApplicationServices.Application
Imports Autodesk.AutoCAD.LayerManager
Imports Autodesk.AutoCAD.Windows
Imports System.Data.OleDb
Imports System.Console
Imports System.Collections
Imports System.Windows
Imports System
Imports Autodesk.AutoCAD.Colors
Public Class frm_contem
    Public nombrefichero As String
    Public k_fin As Integer
    Public were As String
    Public planti As String
    Public Clave As String
    Public Color As String
    Public camino_base As String

    Function AModelSpace(ByVal DBIn As Database, ByVal EntityIn As Entity) As
ObjectId
        Dim myDWG As ApplicationServices.Document
        Dim myDB As DatabaseServices.Database
```

```
        Dim myEd As EditorInput.Editor
        Dim mytransman As DatabaseServices.TransactionManager
        Dim mytrans As DatabaseServices.Transaction
        Dim myBT As DatabaseServices.BlockTable
        Dim myBTR As DatabaseServices.BlockTableRecord
        Dim retIDColl As New ObjectIdCollection
        myDWG =
ApplicationServices.Application.DocumentManager.MdiActiveDocument
        myDWG.LockDocument(DocumentLockMode.AutoWrite, Nothing, Nothing, True)
        myDB = myDWG.Database
        myEd = myDWG.Editor
        mytransman = myDWG.TransactionManager
        mytrans = mytransman.StartTransaction
        myBT =
myDWG.Database.BlockTableId.GetObject(DatabaseServices.OpenMode.ForRead)
        myBTR =
myBT(DatabaseServices.BlockTableRecord.ModelSpace).GetObject(DatabaseServices.O
penMode.ForWrite)
        myBTR.AppendEntity(EntityIn)
        mytrans.AddNewlyCreatedDBObject(EntityIn, True)
        mytrans.Commit()

        Return EntityIn.ObjectId

    End Function

    Sub valorwere()
        'Determinar el valor1 para la sentencia where y la plantilla
        Dim dbconnection As OleDbConnection
        Dim dbdataset As DataSet
        Dim dbdataadapter As OleDbDataAdapter
        Dim cadenaconexion As String
        Dim archivodatos As String
        Dim nombretabla As String
        Dim sentencia As String

        'Consulta de valores a BD de soporte
        archivodatos = "SOPORTE.MDB"
        Dim camGEN, camSUP As String
        camGEN =
Autodesk.AutoCAD.ApplicationServices.Application.Preferences.Files.AutoSavePath
.ToString
        camSUP =
Autodesk.AutoCAD.ApplicationServices.Application.Preferences.Files.TemplateDwgP
ath.ToString
        sentencia = "SELECT indice,ccampoclave,textonum,cplantilla FROM
conIndice"
        nombretabla = "Consulta"
        cadenaconexion = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" &
camSUP & "\" & archivodatos & ";"
        dbconnection = New OleDbConnection(cadenaconexion)
        dbconnection.Open()

        'Consulta de valores a la tabla correspondiente al objeto seleccionado
        dbdataset = New DataSet
        dbdataadapter = New OleDbDataAdapter(sentencia, dbconnection)
        dbdataadapter.Fill(dbdataset, "Consulta")
        dbconnection.Close()
        Dim tabla As System.Data.DataTable
        tabla = dbdataset.Tables("Consulta")
        Dim fila As DataRow
        Dim valor1 As String
        Dim tn As String
        Dim valor2 As String
```

```
        tn = "" : valor1 = ""
        For Each fila In tabla.Rows
            If fila.Item("indice") = Combotipo.Text Then
                valor1 = fila.Item("ccampoclave")
                tn = fila.Item("textonum")
                planti = fila.Item("cplantilla")
                Exit For
            End If
        Next

        'Creación de la consulta WHERE
        valor2 = Clave
        Dim lonvalor2 As Integer
        Dim valact, caracter As String
        were = "" : valact = ""
        lonvalor2 = Len(valor2)
        For x = 1 To lonvalor2
            caracter = Mid(valor2, x, 1)
            If caracter = "," Then
                If tn = "N" Then were = were & valor1 & " = " & valact & " OR "
                If tn = "T" Then were = were & valor1 & " = '" & valact & "' OR
"
                valact = ""
            Else
                valact = valact + caracter
            End If
        Next x
        If valact > "" Then
            If tn = "T" Then were = were & valor1 & " = '" & valact & "'"
            If tn = "N" Then were = were & valor1 & " = " & valact
        End If
    End Sub

    Sub eliminarepetidos()
        Dim sr As New IO.StreamReader(nombrefichero)
        Dim linea As String
        Dim contenido As String

        'Calculo del número de registros iniciales
        Dim k_ini As Integer
        contenido = ""
        Do
            linea = sr.ReadLine()
            If Not linea Is Nothing Then
                k_ini = k_ini + 1
            End If
        Loop Until linea Is Nothing

        'Proceso para depurar valores iguales en los registros
        Dim val_fin(k_ini) As String
        Dim repetido As Integer
        sr.Close()
        Dim ssr As New IO.StreamReader(nombrefichero)
        For x = 1 To k_ini
            linea = ssr.ReadLine()
            repetido = 0
            If x > 1 Then
                For y = 1 To k_fin
                    If linea = val_fin(y) Then repetido = 1 : Exit For
                Next
            End If
            If repetido = 0 Then k_fin = k_fin + 1 : val_fin(k_fin) = linea
        Next
        ssr.Close()
```

```
        ssr = Nothing

        '*****************************ESCRIBIR FICHEROS
        Dim sw As New IO.StreamWriter(nombrefichero)
        For i = 1 To k_fin
            sw.WriteLine(val_fin(i))
        Next i
        'Cerrar ficheros
        sw.Close()
        sw = Nothing
    End Sub

    Private Sub frm_contem_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        cmd_leyen.Enabled = False
        cmd_guarda.Enabled = False
        cmd_ejetema.Enabled = False

        Dim dbconnection As OleDbConnection
        Dim dbdataset As DataSet
        Dim dbdataadapter As OleDbDataAdapter
        Dim cadenaconexion As String
        Dim archivodatos As String
        Dim nombretabla As String
        Dim sentencia As String

        '*********************************************************
        Dim camGEN, camSUP As String
        camGEN =
Autodesk.AutoCAD.ApplicationServices.Application.Preferences.Files.AutoSavePath
.ToString
        camSUP =
Autodesk.AutoCAD.ApplicationServices.Application.Preferences.Files.TemplateDwgP
ath.ToString
        '*********************************************************

        archivodatos = "SOPORTE.MDB"
        sentencia = "SELECT * FROM conIndice ORDER BY indice"
        nombretabla = "Consulta"
        cadenaconexion = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" &
camSUP & "\" & archivodatos & ";"
        dbconnection = New OleDbConnection(cadenaconexion)
        dbconnection.Open()

        dbdataset = New DataSet
        dbdataadapter = New OleDbDataAdapter(sentencia, dbconnection)
        dbdataadapter.Fill(dbdataset, "Consulta")
        dbconnection.Close()
        Dim tabla As System.Data.DataTable
        tabla = dbdataset.Tables("Consulta")
        Dim fila As DataRow
        For Each fila In tabla.Rows
            Me.Combotipo.Items.Add(fila.Item("indice"))
        Next
    End Sub

    Private Sub ComboTipo_SelectedIndexChanged(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles Combotipo.SelectedIndexChanged
        'Actualización del contenido de los desplegables al seleccionar el tipo
de consulta
        cmd_leyen.Enabled = True
        cmd_guarda.Enabled = False
        cmd_ejetema.Enabled = True
```

```
        Dim dbconnection As OleDbConnection
        Dim dbdataset As DataSet
        Dim dbdataadapter As OleDbDataAdapter
        Dim cadenaconexion As String
        Dim archivodatos As String
        Dim nombretabla As String
        Dim sentencia As String

        '*******************************************************
        Dim camGEN, camSUP As String
        camGEN =
Autodesk.AutoCAD.ApplicationServices.Application.Preferences.Files.AutoSavePath
.ToString
        camSUP =
Autodesk.AutoCAD.ApplicationServices.Application.Preferences.Files.TemplateDwgP
ath.ToString
        '*******************************************************

        archivodatos = "SOPORTE.MDB"
        sentencia = "SELECT * FROM conSecundario WHERE indice= '" &
Combotipo.Text & "'"
        nombretabla = "Consulta"
        cadenaconexion = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" &
camSUP & "\" & archivodatos & ";"
        dbconnection = New OleDbConnection(cadenaconexion)
        dbconnection.Open()

        dbdataset = New DataSet
        dbdataadapter = New OleDbDataAdapter(sentencia, dbconnection)
        dbdataadapter.Fill(dbdataset, "Consulta")
        dbconnection.Close()
        Dim tabla As System.Data.DataTable
        tabla = dbdataset.Tables("Consulta")

        If tabla.Rows.Count = 0 Then
            cmd_guarda.Enabled = True
        Else
            cmd_guarda.Enabled = False
        End If
    End Sub

    Private Sub cmd_guarda_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cmd_guarda.Click

        '*******************************************************
        Dim camGEN, camSUP As String
        camGEN =
Autodesk.AutoCAD.ApplicationServices.Application.Preferences.Files.AutoSavePath
.ToString
        camSUP =
Autodesk.AutoCAD.ApplicationServices.Application.Preferences.Files.TemplateDwgP
ath.ToString
        '*******************************************************
        Dim archisoporte As String
        Dim sentencia As String
        archisoporte = camSUP

        sentencia = archisoporte + "\notepad.EXE " + archisoporte +
"\consul.txt"

        Dim myappid As Object
        myappid = Shell(sentencia, 1)

    End Sub
```

```
    Private Sub cmd_ejetema_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cmd_ejetema.Click
        'Depurar y contar el número de registros del fichero consul.txt
        '*******************************************************
        Dim nombrecolores As String
        Dim camGEN, camSUP As String
        camGEN =
Autodesk.AutoCAD.ApplicationServices.Application.Preferences.Files.AutoSavePath
.ToString
        camSUP =
Autodesk.AutoCAD.ApplicationServices.Application.Preferences.Files.TemplateDwgP
ath.ToString
        nombrefichero = camSUP + "\consul.txt"
        nombrecolores = camSUP + "\colores.txt"
        '*******************************************************
        If Combotipo.Text = "Fase" Then nombrefichero = camSUP +
"\tem_fases.txt"
        If Combotipo.Text = "Maderas" Then nombrefichero = camSUP +
"\tem_maderas.txt"
        If Combotipo.Text = "Materiales" Then nombrefichero = camSUP +
"\tem_materiales.txt"
        If Combotipo.Text = "Periodo" Then nombrefichero = camSUP +
"\tem_periodos.txt"

        eliminarepetidos()
        Dim valores(k_fin) As String
        Dim colores(k_fin) As String
        Dim sr As New IO.StreamReader(nombrefichero)
        Dim srcolor As New IO.StreamReader(nombrecolores)
        Dim linea As String
        Dim lineacolor As String

        'Calculo del número de registros iniciales
        For x = 1 To k_fin
            linea = sr.ReadLine()
            lineacolor = srcolor.ReadLine()
            If Not linea Is Nothing Then
                valores(x) = linea
                colores(x) = lineacolor
            End If
        Next x
        sr.Close() : srcolor.Close()
        sr = Nothing : srcolor = Nothing


        'Ejecución de las consultas
        Dim dibujo As Document
        dibujo = Application.DocumentManager.MdiActiveDocument

        Dim amap As MapApplication =
Autodesk.Gis.Map.HostMapApplicationServices.Application
        Dim prj As ProjectModel = amap.ActiveProject
        Dim qry As Query.QueryModel

        qry = prj.CurQuery
        Dim sqlcon As New Query.SqlCondition
        Dim mainqrybr As New
Query.QueryBranch(Constants.JoinOperator.OperatorAnd)
        Dim alteracion As PropertyAlterationDefinition
         For x = 1 To k_fin
            qry.Clear()
            Clave = valores(x)
            Color = colores(x)
```

```
            'Definir la sentencia where y la plantilla de vinculo
            planti = ""
            valorwere()
            sqlcon.WhereCondition = were
            sqlcon.LinkTemplate = planti
            mainqrybr.Clear()
            mainqrybr.AppendOperand(sqlcon)

            'asignar color
            alteracion = qry.PropertyAlteration
            alteracion.Clear()
            alteracion.AddAlteration(Constants.AlterationType.AlterationColor)
            alteracion.Item(0).Expression = colores(x)
            qry.EnablePropertyAlteration(True)

            'Ejecutar query
            qry.Define(mainqrybr)
            qry.Run()
        Next
        Me.Refresh()
        Me.Close()
    End Sub

    Private Sub cmd_leyen_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cmd_leyen.Click
        Dim templateFileName As String

        '*******************************************************
        Dim camGEN, camSUP As String
        camGEN =
Autodesk.AutoCAD.ApplicationServices.Application.Preferences.Files.AutoSavePath
.ToString
        camSUP =
Autodesk.AutoCAD.ApplicationServices.Application.Preferences.Files.TemplateDwgP
ath.ToString
        nombrefichero = camSUP + "\consul.txt"
        '*******************************************************

        Dim archisoporte As String

        Dim dwgprimero, dwgleyenda As
Autodesk.AutoCAD.ApplicationServices.Document
        archisoporte = camSUP
        templateFileName = archisoporte + "\01_leyenda.dwt"

        'Creación de un nuevo dibujo con la plantilla 01_leyenda.dwt
        dwgprimero = Application.DocumentManager.MdiActiveDocument
        dwgleyenda = Application.DocumentManager.Add(templateFileName)
        Application.DocumentManager.MdiActiveDocument = dwgleyenda

        Dim iPoint(0 To 2) As Double
        Dim mytext As New DBText
        Dim mytexto As New DBText()

        Dim dbconnection As OleDbConnection
        Dim dbdataset As DataSet
        Dim dbdataadapter As OleDbDataAdapter
        Dim cadenaconexion As String
        Dim archivodatos As String
        Dim nombretabla As String
        Dim sentencia As String

        'Consulta de los términos de la consulta
```

```
        archivodatos = "SOPORTE.MDB"
        sentencia = "SELECT * FROM conIndice"
        nombretabla = "Consulta"
        cadenaconexion = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" &
camSUP & "\" & archivodatos & ";"
        dbconnection = New OleDbConnection(cadenaconexion)
        dbconnection.Open()

        'Consulta para conocer los valores
        dbdataset = New DataSet
        dbdataadapter = New OleDbDataAdapter(sentencia, dbconnection)
        dbdataadapter.Fill(dbdataset, "Consulta")
        dbconnection.Close()
        Dim tabla As System.Data.DataTable
        tabla = dbdataset.Tables("Consulta")
        Dim fila As DataRow
        Dim texto As String
        texto = ""

        For Each fila In tabla.Rows
            If fila.Item("indice") = Combotipo.Text Then
                texto = fila.Item("cfichtxttematico")
                archisoporte = camSUP + "\" + texto
                Exit For
            End If
        Next
        iPoint(0) = 0.75 : iPoint(1) = 105.3 : iPoint(2) = 0
        mytexto = New DBText
        mytexto.TextString = "Consulta: " & Combotipo.Text
        mytexto.Position = New Point3d(iPoint(0), iPoint(1), iPoint(2))
        mytexto.Height = 0.15
        AModelSpace(HostApplicationServices.WorkingDatabase, mytexto)

        'Escritura de valores en la plantilla leyenda.dwt
        Dim sr As New IO.StreamReader(archisoporte)
        Dim linea As String
        Dim contenido, kont As String
        Dim caracter1, caracter2, caracter3 As String
        Dim fin As Integer
        fin = 0
        contenido = ""
        For y = 1 To 40
            For x = 1 To 3
                kont = 3 * (y - 1) + x
                iPoint(0) = 1.2 * x : iPoint(1) = 105 - (0.4 * y) : iPoint(2) =
0
                linea = sr.ReadLine()
                If Not linea Is Nothing Then
                    texto = LTrim(RTrim(linea))
                    If Combotipo.Text = "UE" Or Combotipo.Text = "Actividad" Or
Combotipo.Text = "Grupo de actividad" Then texto = LTrim(RTrim(linea))
                    If Combotipo.Text = "Piedras" Then texto =
LTrim(RTrim(linea))
                    If Combotipo.Text = "Fase" Then texto =
LTrim(RTrim(Str(Val(Mid(linea, 1, 3)))))
                    If Combotipo.Text = "Periodo" Then
                        linea = LTrim(RTrim(linea))
                        caracter1 = Mid(linea, 1, 1) : caracter2 = Mid(linea,
2, 1) : caracter3 = Mid(linea, 3, 1)
                        If caracter2 = "." Then caracter2 = ""
                        If caracter3 = "." Then caracter3 = ""
                        texto = LTrim(RTrim(caracter1 + caracter2 + caracter3))
                    End If
                    mytexto = New DBText
```

```
                    mytexto.TextString = texto
                    mytexto.Height = 0.09
                    mytexto.Position = New Point3d(iPoint(0), iPoint(1),
iPoint(2))

                    AModelSpace(HostApplicationServices.WorkingDatabase,
mytexto)
                Else
                    fin = 1 : Exit For
                End If
            Next x
            If fin = 1 Then Exit For
        Next y
        Me.Close()
    End Sub
End Class
```

7.14 Código fuente comentado del programa ObjInf

frm_objinf


```
Imports Autodesk.Gis.Map
Imports Autodesk.Gis.Map.DisplayManagement
Imports Autodesk.Gis.Map.Project
Imports Autodesk.AutoCAD.Publishing
Imports Autodesk.AutoCAD.Runtime
Imports Autodesk.AutoCAD
Imports Autodesk.AutoCAD.DatabaseServices
Imports Autodesk.AutoCAD.EditorInput
Imports Autodesk.AutoCAD.Geometry
Imports Autodesk.AutoCAD.ApplicationServices
Imports Autodesk.AutoCAD.ApplicationServices.Application
Imports Autodesk.AutoCAD.LayerManager
Imports Autodesk.AutoCAD.Windows
Imports Autodesk.AutoCAD.Colors
Imports System.Data.OleDb
Imports System.Console
Imports System.Collections
Imports System.Windows
Imports System


Public Class frm_objinf
    'Definición de aquellas variables que se utilizaran en los
    ' diferentes módulos del programa
    Public etilin As String
    Public etivin As String
    Public objid As ObjectId
    Public objcode As Autodesk.AutoCAD.DatabaseServices.ObjectId
    Public planvin As String
    Public id As ObjectId
    Public etiqueta As String
    Public numeti As Integer
    Public formulario, criterio, proyectodatos As String

    Sub abreformu()
        'Esta subrutina devuelve el valor de la sentencia a emplear
        ' el formulario a abrir yla base de datos correspondiente a
        ' la plantilla de vinculo solicitada
```

```
        'Declaración de variables
        Dim eti(0 To 4) As String
        Dim valstr(0 To 4) As String
        Dim valint(0 To 4) As String

        'Asignación de valores
        eti(1) = eti1.Text : valstr(1) = val1.Text
        eti(2) = eti2.Text : valstr(2) = val2.Text
        eti(3) = eti3.Text : valstr(3) = val3.Text
        eti(4) = eti4.Text : valstr(4) = val4.Text
        etiqueta = eti(numeti)

        'Variables para la conexión con BD
        Dim dbconnection As OleDbConnection
        Dim dbdataset As DataSet
        Dim dbdataadapter As OleDbDataAdapter
        Dim cadenaconexion As String
        Dim archivodatos As String
        Dim nombretabla As String
        Dim sentencia As String

        'Asignación de los paths a utilizar
        Dim camGEN, camSUP As String
        camGEN =
Autodesk.AutoCAD.ApplicationServices.Application.Preferences.Files.AutoSavePath
.ToString
        camSUP =
Autodesk.AutoCAD.ApplicationServices.Application.Preferences.Files.TemplateDwgP
ath.ToString

        'Creación del origen de datos utilizado y consulta de la tabla
enlacesBD
        archivodatos = "\SOPORTE.MDB"
        sentencia = "SELECT * FROM enlacesBD"
        nombretabla = "Consulta"
        cadenaconexion = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" &
camSUP & archivodatos & ";"
        dbconnection = New OleDbConnection(cadenaconexion)
        dbconnection.Open()
        dbdataset = New DataSet
        dbdataadapter = New OleDbDataAdapter(sentencia, dbconnection)
        dbdataadapter.Fill(dbdataset, "Consulta")
        dbconnection.Close()

        'Consulta de los datos necesarios para abrir el formulario
correspondiente
        'a los valores de las casillas de texto
        Dim tabla As System.Data.DataTable
        tabla = dbdataset.Tables("Consulta")
        Dim fila As DataRow
        Dim cetiqueta, campo, baseda As String
        campo = "" : baseda = ""
        For Each fila In tabla.Rows
            cetiqueta = fila.Item("cplantillavinculo")
            If cetiqueta = etiqueta Then
                formulario = fila.Item("cformulariomostrado")
                campo = fila.Item("ccampoclave")
                baseda = fila.Item("cbasededatos")
                Exit For
            End If
        Next
        dbconnection.Dispose()
```

```
        'Consulta de los datos necesarios en la tabla conIndice
        'de la base de datos soporte.mdb
        dbconnection = New OleDbConnection(cadenaconexion)
        sentencia = "SELECT * FROM conIndice"
        dbconnection.Open()
        dbdataset = New DataSet
        dbdataadapter = New OleDbDataAdapter(sentencia, dbconnection)
        dbdataadapter.Fill(dbdataset, "Consulta")
        dbconnection.Close()
        tabla = dbdataset.Tables("Consulta")
        Dim tn As String = ""
        For Each fila In tabla.Rows
            If eti(numeti) = UCase(fila.Item("indice")) Then
                tn = fila.Item("textonum")
                Exit For
            End If
        Next
        Dim idficha As String
        If tn = "N" Then
            idficha = CInt(valstr(numeti))
        Else
            idficha = Chr(34) & valstr(numeti) & Chr(34)
        End If

        'Asignacion de los valores necesarios para abrir
        'el formulario correcto
        criterio = campo & " = " & idficha
        proyectodatos = camGEN & "\datos\" & baseda


        dbconnection.Dispose()
    End Sub
    Sub editlink()
        'Esta subrutina devuelve el valor enlazado a una entidad
        ' correspondiente a una plantilla de vinculo establecida

        Dim indice As Integer
        On Error Resume Next

        'Decalración de variables
        Dim dbc As CAO.DbConnect
        Dim linkt As CAO.LinkTemplate
        Dim objlink As CAO.Link
        Dim objlinks As CAO.Links
        Dim idarray(0 To 0) As Long

        'Conexion del el CAO
        dbc = AcadApplication.GetInterfaceObject("CAO.DbConnect.16")
        linkt = dbc.GetLinkTemplates(0).Item(planvin)
        If Err.Number <> 0 Then
            MsgBox(planvin & "no encontrada")
            GoTo done
        End If

        'Consulta de todos los links asociados a la entidad
        Dim ObjIds(0 To 0) As Int32
        ObjIds(0) = objcode.OldId
        objlinks = dbc.GetLinks(linkt, ObjIds, 2, 0)
        If Err.Number <> 0 Then
            GoTo done
        End If

        'Extracción del valor correspondiente a la plantilla definida
        objlink = objlinks.Item(0)
```

```
        If objlinks.Count = 0 Then
            etilin = 0 : etivin = "kk"
        Else
            etilin = objlink.KeyValues.Item(0).Value
            etivin = objlink.LinkTemplate.Name
        End If

done:
        dbc = Nothing
    End Sub
    Private Sub cmd_selec_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cmd_selec.Click
        eti1.Text = "" : eti2.Text = "" : eti3.Text = "" : eti4.Text = ""
        val1.Text = "" : val2.Text = "" : val3.Text = "" : val4.Text = ""

        'Declaracion de variables
        Dim myDWG As ApplicationServices.Document
        Dim myDB As DatabaseServices.Database
        Dim myEd As EditorInput.Editor
        Dim mytransman As DatabaseServices.TransactionManager
        Dim mytrans As DatabaseServices.Transaction
        Dim myBT As DatabaseServices.BlockTable
        Dim myBTR As DatabaseServices.BlockTableRecord
        Dim retIDColl As New ObjectIdCollection
        myDWG =
ApplicationServices.Application.DocumentManager.MdiActiveDocument
        myDWG.LockDocument(DocumentLockMode.AutoWrite, Nothing, Nothing, True)
        myDB = myDWG.Database
        myEd = myDWG.Editor
        mytransman = myDWG.TransactionManager
        mytrans = mytransman.StartTransaction
        myBT =
myDWG.Database.BlockTableId.GetObject(DatabaseServices.OpenMode.ForWrite)
        myBTR =
myBT(DatabaseServices.BlockTableRecord.ModelSpace).GetObject(DatabaseServices.O
penMode.ForWrite)

        'Selección de la entidad
        Dim Opts As New PromptSelectionOptions()
        Me.Hide()
        Opts.SingleOnly = True
        Opts.MessageForAdding = "Selecciona una única entidad...." & vbCrLf
        Dim res As PromptSelectionResult = myEd.GetSelection(Opts)
        If res.Status = PromptStatus.Error Then
            Return
        End If
        Dim SS As Autodesk.AutoCAD.EditorInput.SelectionSet = res.Value
        Dim idarray As ObjectId() = SS.GetObjectIds()
        Dim etity As Entity
        For Each id In idarray
            etity = mytransman.GetObject(id, OpenMode.ForRead, True)
            objcode = etity.ObjectId
            etity.Dispose()
        Next id

        'Declaración de las variables para la conexión con la BD
        Dim dbc As CAO.DbConnect
        Dim colplan As CAO.LinkTemplates
        Dim planvinculo As CAO.LinkTemplate
        Dim ite, recuadro As Integer

        'Establecer plantilla de vínculo
        dbc = AcadApplication.GetInterfaceObject("CAO.DbConnect.16")
        colplan = dbc.GetLinkTemplates(0)
```

```
        'Proceso que se repite para cada plantilla de vinculo
        ite = 1
        recuadro = 0
        Dim VinPlan(0 To 10) As String
        Dim VinVal(0 To 10) As String
        Dim VinNum As Int16
        VinNum = 0
        For x = 0 To colplan.Count - 1
            planvinculo = colplan(x)
            planvin = planvinculo.Name
            editlink()
            If etivin <> "kk" Then
                VinPlan(VinNum) = etivin
                VinVal(VinNum) = etilin
                VinNum += 1
                If VinNum = 1 Then val1.Text = etilin : eti1.Text = etivin
                If VinNum = 2 Then val2.Text = etilin : eti2.Text = etivin
                If VinNum = 3 Then val3.Text = etilin : eti3.Text = etivin
                If VinNum = 4 Then val4.Text = etilin : eti4.Text = etivin
            End If
        Next
        'resetear los valores utilizados
        Dim textovinculos As String = ""
        SS.Dispose()
        VinNum = 0
        mytrans.Commit()
        mytrans.Dispose()
        mytransman.Dispose()
        Me.Show()
    End Sub

    Private Sub abre1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles abre1.Click

        'Crear una instancia de Access con los valores adecuados
        Dim oaccess As Access.Application
        numeti = 1
        abreformu()
        oaccess = CreateObject("Access.Application")
        oaccess.OpenCurrentDatabase(proyectodatos, False)
        oaccess.DoCmd.OpenForm(formulario, Access.AcFormView.acNormal, ,
criterio)
        oaccess.Visible = True
        MsgBox("Aurrera..")

    End Sub

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

    End Sub

    Private Sub abre2_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles abre2.Click
        'Crear una instancia de Access con los valores adecuados
        Dim oaccess As Access.Application
        numeti = 2
        abreformu()
        oaccess = CreateObject("Access.Application")
        oaccess.OpenCurrentDatabase(proyectodatos, False)
        oaccess.DoCmd.OpenForm(formulario, Access.AcFormView.acNormal, ,
criterio)
        oaccess.Visible = True
```

```
        MsgBox("Aurrera..")
    End Sub

    Private Sub abre3_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles abre3.Click
        'Crear una instancia de Access con los valores adecuados
        Dim oaccess As Access.Application
        numeti = 3
        abreformu()
        oaccess = CreateObject("Access.Application")
        oaccess.OpenCurrentDatabase(proyectodatos, False)
        oaccess.DoCmd.OpenForm(formulario, Access.AcFormView.acNormal, ,
criterio)
        oaccess.Visible = True
        MsgBox("Aurrera..")
    End Sub

    Private Sub abre4_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles abre4.Click
        'Crear una instancia de Access con los valores adecuados
        Dim oaccess As Access.Application
        numeti = 4
        abreformu()
        oaccess = CreateObject("Access.Application")
        oaccess.DoCmd.OpenForm(formulario, Access.AcFormView.acNormal, ,
criterio)
        oaccess.Visible = True
        MsgBox("Aurrera..")
    End Sub
End Class
```

7.15 Código fuente comentado del programa ResDib


```
Imports Autodesk.Gis.Map
Imports Autodesk.Gis.Map.DisplayManagement
Imports Autodesk.Gis.Map.Project
Imports Autodesk.AutoCAD.Publishing
Imports Autodesk.AutoCAD.Runtime
Imports Autodesk.AutoCAD
Imports Autodesk.AutoCAD.DatabaseServices
Imports Autodesk.AutoCAD.EditorInput
Imports Autodesk.AutoCAD.Geometry
Imports Autodesk.AutoCAD.ApplicationServices
Imports Autodesk.AutoCAD.ApplicationServices.Application
Imports Autodesk.AutoCAD.LayerManager
Imports Autodesk.AutoCAD.Windows
Imports Autodesk.AutoCAD.Colors
Imports System.Data.OleDb
Imports System.Console
Imports System.Collections
Imports System.Windows
Imports System

Public Class frm_resdib
    Public etilin As String
    Public etivin As String
    Public objid As ObjectId
    Public objcode As Autodesk.AutoCAD.DatabaseServices.ObjectId
    Public planvin As String
    Public id As ObjectId
```

```
    Public etiqueta As String
    Public numeti As Integer
    Public formulario, criterio, proyectodatos As String

    Sub abreformu()
        'Esta subrutina define el criterio y la base de datos que le
corresponde
        etiqueta = planvin
        Dim dbconnection As OleDbConnection
        Dim dbdataset As DataSet
        Dim dbdataadapter As OleDbDataAdapter
        Dim cadenaconexion As String
        Dim archivodatos As String
        Dim nombretabla As String
        Dim sentencia As String

        Dim camGEN, camSUP As String
        camGEN =
Autodesk.AutoCAD.ApplicationServices.Application.Preferences.Files.AutoSavePath
.ToString
        camSUP =
Autodesk.AutoCAD.ApplicationServices.Application.Preferences.Files.TemplateDwgP
ath.ToString

        'Consulta de los campos a utilizar
        archivodatos = "\SOPORTE.MDB"
        sentencia = "SELECT * FROM enlacesBD"
        nombretabla = "Consulta"
        cadenaconexion = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" &
camSUP & archivodatos & ";"
        dbconnection = New OleDbConnection(cadenaconexion)
        dbconnection.Open()
        dbdataset = New DataSet
        dbdataadapter = New OleDbDataAdapter(sentencia, dbconnection)
        dbdataadapter.Fill(dbdataset, "Consulta")
        dbconnection.Close()
        Dim tabla As System.Data.DataTable
        tabla = dbdataset.Tables("Consulta")
        Dim fila As DataRow
        Dim cetiqueta, campo, baseda As String
        campo = "" : baseda = ""
        For Each fila In tabla.Rows
            cetiqueta = fila.Item("cplantillavinculo")
            If cetiqueta = etiqueta Then
                formulario = fila.Item("cformulariomostrado")
                campo = fila.Item("ccampoclave")
                baseda = fila.Item("cbasededatos")
                Exit For
            End If
        Next
        dbconnection.Dispose()
        'Consulta para extraer el valor del filtro
        dbconnection = New OleDbConnection(cadenaconexion)
        sentencia = "SELECT * FROM conIndice"
        dbconnection.Open()
        dbdataset = New DataSet
        dbdataadapter = New OleDbDataAdapter(sentencia, dbconnection)
        dbdataadapter.Fill(dbdataset, "Consulta")
        dbconnection.Close()
        tabla = dbdataset.Tables("Consulta")
        Dim tn As String = ""
        For Each fila In tabla.Rows
            If planvin = UCase(fila.Item("indice")) Then
                tn = fila.Item("textonum")
```

```
                Exit For
            End If
        Next
        'creación del filtro criterio
        Dim idficha As String
        Dim dgvvalor As String = dgv_clave.Item(0,
dgv_clave.CurrentRow.Index).Value

        If tn = "N" Then
            idficha = CInt(dgvvalor)
        Else
            idficha = Chr(34) & dgvvalor & Chr(34)
        End If

        criterio = campo & " = " & idficha
        proyectodatos = camGEN & "\datos\" & baseda
        dbconnection.Dispose()
        dbconnection.Close()
    End Sub

    Sub GrupoLink(ByVal objcode2() As Integer)
        'Definición de carpetas de trabajo
        Dim camGEN, camSUP As String
        camGEN =
Autodesk.AutoCAD.ApplicationServices.Application.Preferences.Files.AutoSavePath
.ToString
        camSUP =
Autodesk.AutoCAD.ApplicationServices.Application.Preferences.Files.TemplateDwgP
ath.ToString

        'Declaración de variables
        Dim myfsw As IO.StreamWriter
        Dim indice As Integer
        On Error Resume Next
        Dim dbc As CAO.DbConnect
        Dim linkt As CAO.LinkTemplate
        Dim objlink As CAO.Link
        Dim objlinks As CAO.Links
        Dim idarray(0 To 0) As Long
        dbc = AcadApplication.GetInterfaceObject("CAO.DbConnect.16")
        linkt = dbc.GetLinkTemplates(0).Item(planvin)

        If Err.Number <> 0 Then
            MsgBox(planvin & "no encontrada")
            GoTo done
        End If

        'Declaración de la matriz que contendrá los números
        'IDs de las entidades seleccionadas
        Dim ObjIds(objcode2.Count - 1) As Int32
        For x = 0 To objcode2.Count - 1
            ObjIds(x) = objcode2(x)
        Next x
        objlinks = dbc.GetLinks(linkt, ObjIds, 2, 0)

        If Err.Number <> 0 Then
            GoTo done
        End If
        Dim valores(objlinks.Count - 1) As String
        Dim valoresrepe(objlinks.Count - 1) As String
        Dim kdif, repe As Integer
        kdif = 0

        'Archivo generado con los valores de los enlaces
```

```
        Dim fich As String
        fich = camSUP & "\" & planvin & ".txt"
        myfsw = New IO.StreamWriter(fich)


        'Bucle para todos los elementos
        For x = 0 To objlinks.Count - 1
            'Se extrae el valor de la plantilla de vinculo actual
            objlink = objlinks.Item(x)
            If objlinks.Count = 0 Then
                etilin = 0 : etivin = "kk"
            Else
                etilin = objlink.KeyValues.Item(0).Value
                etivin = objlink.LinkTemplate.Name
            End If
            'Válido para objetos con links
            If etivin <> "kk" Then
                If x > 0 Then
                    'Calcular si el valor del enlace está repetido
                    ' y en caso contrario añadirlo a las lista de valores
                    repe = 0
                    For i = 0 To kdif - 1
                        If etilin = valores(i) Then
                            repe = 1
                            Exit For
                        End If
                    Next i
                    If repe = 0 Then
                        valores(kdif) = etilin
                        kdif = kdif + 1
                    End If
                Else
                    'El enlace del primer elemento no puede estar repetido
                    valores(kdif) = etilin
                    kdif = kdif + 1
                End If
            End If
        Next x

        'Se escribe en el fichero los valores almacenados
        For x = 0 To kdif - 1
            If x = 0 Then Me.cbx_plantillas.Items.Add(etivin)
            myfsw.WriteLine(valores(x))
        Next x
        myfsw.Close()
        myfsw.Dispose()

done:
        dbc = Nothing
    End Sub

    Sub llenadatagrid()
        'Declaración de variables
        Dim camGEN, camSUP As String
        camGEN =
Autodesk.AutoCAD.ApplicationServices.Application.Preferences.Files.AutoSavePath
.ToString
        camSUP =
Autodesk.AutoCAD.ApplicationServices.Application.Preferences.Files.TemplateDwgP
ath.ToString

        dgv_clave.Rows.Clear()
        'Definición del archivo de lectura
        Dim myfsw As IO.StreamReader
```

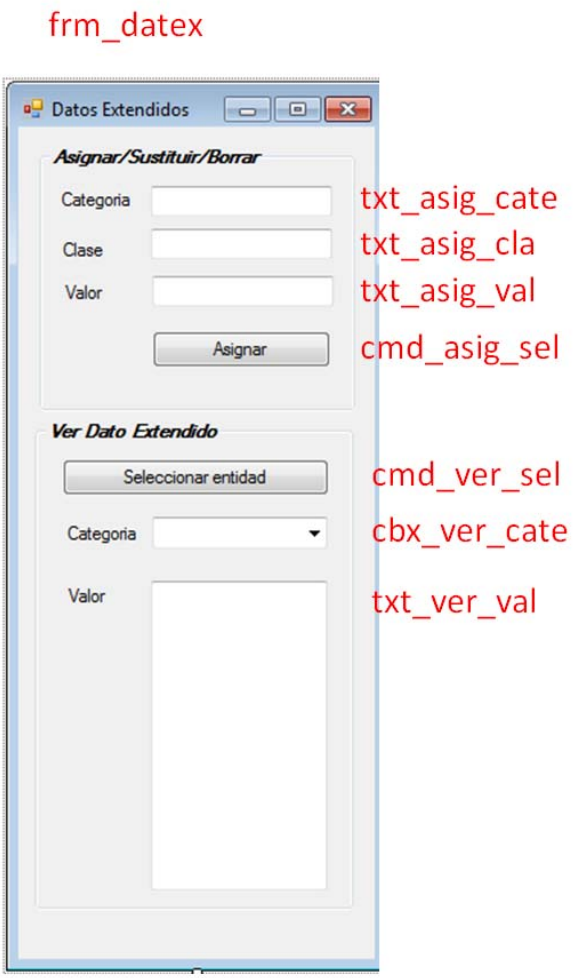
```
        Dim fich As String
        fich = camSUP & "\" & cbx_plantillas.Text & ".txt"
        myfsw = New IO.StreamReader(fich)
        Dim linea As String
        'Escritura del archivo
        Do
            linea = myfsw.ReadLine()
            If Not linea Is Nothing Then
                If linea <> "" Then
                    dgv_clave.Rows.Add(linea)
                End If
            End If
        Loop Until linea Is Nothing
    End Sub

    Private Sub cmd_selec_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cmd_selec.Click
        'Ocultar formulario y puesta a cero de los valores de la lista
        Me.Hide()
        Me.cbx_plantillas.Items.Clear()
        Me.dgv_clave.Rows.Clear()

        'Declarar y asignar valores a diferentes variables
        Dim mydwg As ApplicationServices.Document
        Dim mydb As DatabaseServices.Database
        Dim mytransman As DatabaseServices.TransactionManager
        Dim mytrans As DatabaseServices.Transaction

        mydwg =
ApplicationServices.Application.DocumentManager.MdiActiveDocument
        mydb = mydwg.Database
        mytransman = mydwg.TransactionManager
        mytrans = mytransman.StartTransaction

        Dim myEd As Editor = DocumentManager.MdiActiveDocument.Editor
        Dim myPSR As PromptSelectionResult = myEd.GetSelection
        Dim SS As Autodesk.AutoCAD.EditorInput.SelectionSet = myPSR.Value

        'Si la seleccion es correcta...
        If myPSR.Status = PromptStatus.OK Then
            If IsNothing(myPSR.Value) = False Then
                Dim dbc As CAO.DbConnect
                Dim colplan As CAO.LinkTemplates
                Dim planvinculo As CAO.LinkTemplate
                Dim ite, recuadro As Integer
                dbc = AcadApplication.GetInterfaceObject("CAO.DbConnect.16")
                colplan = dbc.GetLinkTemplates(0)

                'Se calculan cuantos objetos se ha seleccionado
                Dim contador As Integer
                contador = 0
                For Each myObjID As ObjectId In myPSR.Value.GetObjectIds
                    contador = contador + 1
                Next

                'Se asignan sus IDs a una matriz de valores
                Dim objcode2(contador - 1) As Integer
                contador = 0
                For Each myObjID As ObjectId In myPSR.Value.GetObjectIds
                    objcode2(contador) = myObjID.OldId
                    contador = contador + 1
                Next
                ite = 1
                recuadro = 0
```

```
                'Para cada plantilla de vinculo se realiza la subrutina
Grupolink
                For x = 0 To colplan.Count - 1
                    planvinculo = colplan(x)
                    planvin = planvinculo.Name
                    GrupoLink(objcode2)
                Next x
                Dim textovinculos As String = ""
            End If
        End If

        'Se llena el datagrid con la platilla seleccinada
        '(en caso de no estar seleccionada se elige UC
        If cbx_plantillas.Text = "" Then
            cbx_plantillas.Text = "UC"
        Else
            llenadatagrid()
        End If

        mytrans.Commit()
        mytrans.Dispose()
        mytransman.Dispose()
        Me.Show()
    End Sub

    Private Sub cbx_plantillas_SelectedIndexChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
cbx_plantillas.SelectedIndexChanged
        'Limpiar el datagrid
        dgv_clave.Rows.Clear()
        Dim myfsw As IO.StreamReader
        Dim fich As String

        'Asignar caminos de carpetas
        Dim camGEN, camSUP As String
        camGEN =
Autodesk.AutoCAD.ApplicationServices.Application.Preferences.Files.AutoSavePath
.ToString
        camSUP =
Autodesk.AutoCAD.ApplicationServices.Application.Preferences.Files.TemplateDwgP
ath.ToString

        'Llenar el datagrid con los registros del archivo correspondiente
        fich = camSUP & "\" & cbx_plantillas.SelectedItem & ".txt"
        myfsw = New IO.StreamReader(fich)
        Dim linea As String
        Do
            linea = myfsw.ReadLine()
            If Not linea Is Nothing Then
                If linea <> "" Then
                    dgv_clave.Rows.Add(linea)
                End If
            End If
        Loop Until linea Is Nothing
    End Sub

    Private Sub dgv_clave_CellDoubleClick(ByVal sender As Object, ByVal e As
System.Windows.Forms.DataGridViewCellEventArgs) Handles
dgv_clave.CellDoubleClick
        'Se abre una instancia de access tomando en cuenta los valores del
        'desplegable
        Dim oaccess As Access.Application
        planvin = cbx_plantillas.SelectedItem
        abreformu()
```

```
        oaccess = CreateObject("Access.Application")
        oaccess.OpenCurrentDatabase(proyectodatos, False)
        oaccess.DoCmd.OpenForm(formulario, Access.AcFormView.acNormal, ,
criterio)
        oaccess.Visible = True
        oaccess.RefreshDatabaseWindow()
    End Sub

    Private Sub cmd_consulta_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cmd_consulta.Click
        'Ocultar formulario y puesta a cero de los valores de la lista
        Me.Hide()
        Me.cbx_plantillas.Items.Clear()
        Me.dgv_clave.Rows.Clear()

        'Declarar y asignar valores a diferentes variables
        Dim mydwg As ApplicationServices.Document
        Dim mydb As DatabaseServices.Database
        Dim mytransman As DatabaseServices.TransactionManager
        Dim mytrans As DatabaseServices.Transaction

        mydwg =
ApplicationServices.Application.DocumentManager.MdiActiveDocument
        mydb = mydwg.Database
        mytransman = mydwg.TransactionManager
        mytrans = mytransman.StartTransaction

        'Selección de todos los elemntos del dibujo actual
        Dim myEd As Editor = DocumentManager.MdiActiveDocument.Editor
        Dim mySS As SelectionSet = mydwg.Editor.SelectAll.Value

        'Si la seleccion es correcta...
        If IsNothing(mySS) = False Then
            Dim dbc As CAO.DbConnect
            Dim colplan As CAO.LinkTemplates
            Dim planvinculo As CAO.LinkTemplate
            Dim ite, recuadro As Integer
            dbc = AcadApplication.GetInterfaceObject("CAO.DbConnect.16")
            colplan = dbc.GetLinkTemplates(0)

            'Se calculan cuantos objetos se ha seleccionado
            Dim contador As Integer
            contador = 0
            For Each myObjID As ObjectId In mySS.GetObjectIds
                contador = contador + 1
            Next

            'Se asignan sus IDs a una matriz de valores
            Dim objcode2(contador - 1) As Integer
            contador = 0
            For Each myObjID As ObjectId In mySS.GetObjectIds
                objcode2(contador) = myObjID.OldId
                contador = contador + 1
            Next
            ite = 1
            recuadro = 0

            'Para cada plantilla de vinculo se realiza la subrutina Grupolink
            For x = 0 To colplan.Count - 1
                planvinculo = colplan(x)
                planvin = planvinculo.Name
                GrupoLink(objcode2)
            Next x
            Dim textovinculos As String = ""
```

```
        End If

        'Se llena el datagrid con la platilla seleccinada
        '(en caso de no estar seleccionada se elige UC
        If cbx_plantillas.Text = "" Then
            cbx_plantillas.Text = "UC"
        Else
            llenadatagrid()
        End If
        mytrans.Commit()
        mytrans.Dispose()
        mytransman.Dispose()
        Me.Show()
    End Sub
End Class
```

7.16 Código fuente comentado del programa Datex




```
Imports Autodesk.Gis.Map
Imports Autodesk.Gis.Map.DisplayManagement
Imports Autodesk.Gis.Map.Project
Imports Autodesk.AutoCAD.Publishing
Imports Autodesk.AutoCAD.Runtime
Imports Autodesk.AutoCAD
Imports Autodesk.AutoCAD.DatabaseServices
Imports Autodesk.AutoCAD.EditorInput
Imports Autodesk.AutoCAD.Geometry
Imports Autodesk.AutoCAD.ApplicationServices
Imports Autodesk.AutoCAD.ApplicationServices.Application
Imports Autodesk.AutoCAD.LayerManager
Imports Autodesk.AutoCAD.Windows
Imports Autodesk.AutoCAD.Colors
Imports System.Data.OleDb
Imports System.Console
Imports System.Collections
Imports System.Windows
Imports System
Imports System.Windows.Forms


Public Class frm_Datex
    Public pubID As ObjectId
```

```
    Function GetXData(ByVal entID As ObjectId, ByVal categoria As String) As
ResultBuffer
        Using tr As Transaction =
entID.Database.TransactionManager.StartTransaction
            Dim ent As Entity = entID.GetObject(OpenMode.ForRead)
            Dim rb As ResultBuffer = ent.GetXDataForApplication(categoria)
            Return rb
        End Using
    End Function

    Private Sub cmd_asig_sel_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cmd_asig_sel.Click

        'Actualizar valores de consulta de datex
        txt_ver_val.Text = ""
        txt_ver_val.BackColor = Drawing.Color.White

        Me.Hide()

        Dim mydwg As ApplicationServices.Document
        Dim mydb As DatabaseServices.Database
        Dim mytransman As DatabaseServices.TransactionManager
        Dim mytrans As DatabaseServices.Transaction

        mydwg =
ApplicationServices.Application.DocumentManager.MdiActiveDocument
        mydwg.LockDocument(DocumentLockMode.AutoWrite, Nothing, Nothing, True)
        mydb = mydwg.Database
        mytransman = mydwg.TransactionManager
        mytrans = mytransman.StartTransaction

        Dim myEd As Editor = DocumentManager.MdiActiveDocument.Editor
        Dim myPSR As PromptSelectionResult = myEd.GetSelection
        Dim SS As Autodesk.AutoCAD.EditorInput.SelectionSet = myPSR.Value

        Dim categoria, clase, valor As String
        clase = txt_asig_cla.Text
        categoria = txt_asig_cate.Text
        valor = txt_asig_val.Text

        ' ************* Añade o Sustituye datos extendidos
**************************
        If categoria <> "" And clase <> "" And valor <> "" Then
            If myPSR.Status = PromptStatus.OK Then
                If IsNothing(myPSR.Value) = False Then
                    'Para cada objeto de la selccion
                    For Each myObjID As ObjectId In myPSR.Value.GetObjectIds
                        Dim ent As Entity = mytrans.GetObject(myObjID,
OpenMode.ForWrite)
                        Dim rb As ResultBuffer =
ent.GetXDataForApplication("SIM")
                        If rb Is Nothing Then
                            'Registra la categoria SIM si no la tiene definida
el objeto
                            'crea añade la clase y el valor indicados
                            Dim regAppTbl As RegAppTable =
mytrans.GetObject(mydb.RegAppTableId, OpenMode.ForWrite)
                            If regAppTbl.Has(categoria) = False Then
                                Dim regAppRec As New RegAppTableRecord
                                regAppTbl.UpgradeOpen()
                                regAppRec.Name = categoria
                                regAppTbl.Add(regAppRec)
```

```
                                mytransman.AddNewlyCreatedDBObject(regAppRec,
True)
                            End If
                            ent.XData = New ResultBuffer(New TypedValue(1001,
categoria), New TypedValue(1000, clase), New TypedValue(1000, valor))
                        Else

                            Dim listv As New List(Of TypedValue)(rb.AsArray())
                            Dim ndatex As New List(Of TypedValue)
                            Dim actualizado As Boolean
                            actualizado = False
                            ndatex.Add(New TypedValue(1001, categoria))
                            'Recorre todos los valores de clase asignados a la
categoria SIM
                            'Los valores siempre se establecen por parejas
                            'Crea un nuevo Dato extendido con el que ya existia
                            'añadiendo o sustituyendo los nuevos valores.

                            For x = 0 To ((rb.AsArray.Count - 1) / 2) - 1
                                If rb.AsArray(2 * x + 1).Value.ToString = clase
Then
                                    ndatex.Add(New TypedValue(1000, clase))
                                    ndatex.Add(New TypedValue(1000, valor))
                                    actualizado = True
                                Else
                                    ndatex.Add(New TypedValue(1000,
rb.AsArray(2 * x + 1).Value.ToString))
                                    ndatex.Add(New TypedValue(1000,
rb.AsArray(2 * x + 2).Value.ToString))
                                End If
                            Next

                            If actualizado = False Then
                                ndatex.Add(New TypedValue(1000, clase))
                                ndatex.Add(New TypedValue(1000, valor))
                                actualizado = True
                            End If
                            ent.XData = New ResultBuffer(ndatex.ToArray())
                        End If
                    Next
                End If
            End If
        End If


        ' *************  Borra datos extendidos
*****************************************

        If categoria <> "" And clase <> "" And valor = "" Then
            Dim message As String = "¿Seguro que quieres borrar datos
extendidos?"
            Dim caption As String = "¡Atención!"
            Dim result = MessageBox.Show(message, caption,
MessageBoxButtons.YesNo, MessageBoxIcon.Question)

            If (result = DialogResult.Yes) Then
                If myPSR.Status = PromptStatus.OK Then
                    If IsNothing(myPSR.Value) = False Then

                        'Para cada objeto de la selccion
                        For Each myObjID As ObjectId In
myPSR.Value.GetObjectIds
                            Dim ent As Entity = mytrans.GetObject(myObjID,
OpenMode.ForWrite)
```

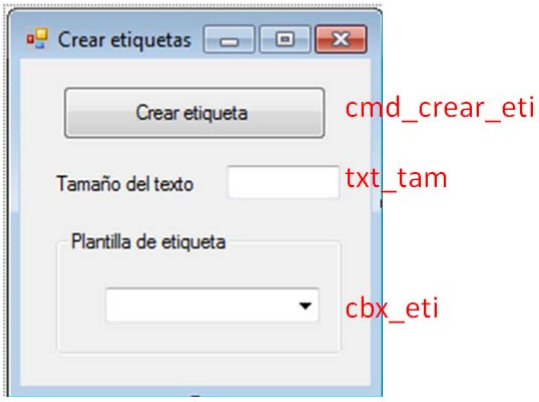
```
                            Dim rb As ResultBuffer =
ent.GetXDataForApplication("SIM")
                            If Not rb Is Nothing Then
                                Dim listv As New List(Of
TypedValue)(rb.AsArray())
                                Dim ndatex As New List(Of TypedValue)
                                Dim actualizado As Boolean
                                actualizado = False
                                ndatex.Add(New TypedValue(1001, categoria))
                                'Se va leyendo el campo que define la clase
                                'Si coincide con el indicado se pasa al
siguiente
                                'Si no coincide se añaden los valores al nuevo
dato extendido
                                'El nuevo dato extendido es el mismo pero sin
la pareja de datos
                                'que se quería borrar.
                                For x = 0 To ((rb.AsArray.Count - 1) / 2) - 1
                                    If rb.AsArray(2 * x + 1).Value.ToString =
clase Then
                                        actualizado = True
                                    Else
                                        ndatex.Add(New TypedValue(1000,
rb.AsArray(2 * x + 1).Value.ToString))
                                        ndatex.Add(New TypedValue(1000,
rb.AsArray(2 * x + 2).Value.ToString))
                                    End If
                                Next
                                ent.XData = New ResultBuffer(ndatex.ToArray())
                            End If
                        Next
                    End If
                End If
            End If
        End If
        mytrans.Commit()
        mytrans.Dispose()
        mytransman.Dispose()

        Me.Show()

    End Sub

    Private Sub cmd_ver_sel_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cmd_ver_sel.Click
        Me.Hide()
        txt_ver_val.BackColor = Drawing.Color.White

        'declaración de variables
        Dim mydwg As ApplicationServices.Document
        Dim mydb As DatabaseServices.Database
        Dim mytransman As DatabaseServices.TransactionManager
        Dim mytrans As DatabaseServices.Transaction

        mydwg =
ApplicationServices.Application.DocumentManager.MdiActiveDocument
        mydwg.LockDocument(DocumentLockMode.AutoWrite, Nothing, Nothing, True)
        mydb = mydwg.Database
        mytransman = mydwg.TransactionManager
        mytrans = mytransman.StartTransaction

        'Selección de entidades
        Dim myEd As Editor = DocumentManager.MdiActiveDocument.Editor
        Dim myPSR As PromptSelectionResult = myEd.GetSelection
```

```
        Dim SS As Autodesk.AutoCAD.EditorInput.SelectionSet = myPSR.Value

        Dim valor As String
        valor = ""
        Dim categoria As String
        categoria = cbx_ver_cate.Text

        'Lectura de los valores asociados a la categoria SIM
        ' de la entidad seleccionada
        If myPSR.Status = PromptStatus.OK Then
            If IsNothing(myPSR.Value) = False Then
                For Each myObjID As ObjectId In myPSR.Value.GetObjectIds
                    pubID = myObjID
                    txt_ver_val.Text = ""
                    Dim rb As ResultBuffer = GetXData(myObjID, categoria)
                    If Not rb Is Nothing Then
                        For Each typV As TypedValue In rb
                            If Not typV.Value = categoria Then
                                valor = valor & typV.Value & vbCrLf
                            End If
                        Next
                        txt_ver_val.Text = valor
                        myObjID = myPSR.Value.GetObjectIds(0)
                    Else
                        txt_ver_val.BackColor = Drawing.Color.Salmon
                        txt_ver_val.Text = "No existen datos extendidos para
esta clave"
                    End If
                Next
            End If
        End If

        mytrans.Commit()
        mytrans.Dispose()
        mytransman.Dispose()
        Me.Show()
    End Sub

    Private Sub cmd_asigDatex_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        'Actualizar variables y textos
        txt_ver_val.BackColor = Drawing.Color.White
        txt_ver_val.Text = ""
        txt_asig_cate.Text = "SIM"
        cbx_ver_cate.Items.Clear()
        cbx_ver_cate.Items.Add("_JV_")
        cbx_ver_cate.Items.Add("SIM")
        cbx_ver_cate.Refresh()
        cbx_ver_cate.Text = "SIM"
    End Sub

    Private Sub cbx_ver_cate_SelectedIndexChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs) Handles
cbx_ver_cate.SelectedIndexChanged
        If Not pubID = Nothing Then
            txt_ver_val.BackColor = Drawing.Color.White
            txt_ver_val.Text = ""

            'Declarar las variables necesarias
            Dim mydwg As ApplicationServices.Document
            Dim mydb As DatabaseServices.Database
            Dim mytransman As DatabaseServices.TransactionManager
            Dim mytrans As DatabaseServices.Transaction
```

```
            mydwg =
ApplicationServices.Application.DocumentManager.MdiActiveDocument
            mydwg.LockDocument(DocumentLockMode.AutoWrite, Nothing, Nothing,
True)
            mydb = mydwg.Database
            mytransman = mydwg.TransactionManager
            mytrans = mytransman.StartTransaction

            Dim myEd As Editor = DocumentManager.MdiActiveDocument.Editor
            Dim valor As String
            valor = ""
            Dim categoria As String
            categoria = cbx_ver_cate.Text
            txt_ver_val.Text = ""
            'Recuperar lo valores del dato extendido para otra categoria
            Dim rb As ResultBuffer = GetXData(pubID, categoria)
            If Not rb Is Nothing Then
                For Each typV As TypedValue In rb
                    If Not typV.Value = categoria Then
                        valor = valor & typV.Value & vbCrLf
                    End If
                Next
                txt_ver_val.Text = valor
            Else
                txt_ver_val.BackColor = Drawing.Color.Salmon
                txt_ver_val.Text = "No existen datos extendidos para esta
clave"
            End If

            mytrans.Commit()
            mytrans.Dispose()
            mytransman.Dispose()
        End If
    End Sub
End Class
```

7.17 Código fuente comentado del programa Eti

frm_etiquetas




```
Imports Autodesk.Gis.Map
Imports Autodesk.Gis.Map.DisplayManagement
Imports Autodesk.Gis.Map.Project
Imports Autodesk.AutoCAD.Publishing
Imports Autodesk.AutoCAD.Runtime
Imports Autodesk.AutoCAD
Imports Autodesk.AutoCAD.DatabaseServices
Imports Autodesk.AutoCAD.EditorInput
Imports Autodesk.AutoCAD.Geometry
Imports Autodesk.AutoCAD.ApplicationServices
Imports Autodesk.AutoCAD.ApplicationServices.Application
Imports Autodesk.AutoCAD.LayerManager
Imports Autodesk.AutoCAD.Windows
Imports Autodesk.AutoCAD.Colors
Imports System.Data.OleDb
Imports System.Console
Imports System.Collections
Imports System.Windows
Imports System
Imports System.Windows.Forms


Public Class frm_etiquetas
    Public etilin As String
    Public etivin As String

    Public objid As ObjectId
    Public objcode As Autodesk.AutoCAD.DatabaseServices.ObjectId
    Public planvin As String
    Public txtEtiqueta As String
    Public textonum As String
    Public enlacenulo, plantillanula As Boolean

    Public vueltas As Single

    Sub resetini()
        'Poner a 0 los valores iniciales
        etilin = Nothing : etivin = Nothing
        objid = Nothing : objcode = Nothing
```

```
        planvin = Nothing : txtEtiqueta = Nothing
        textonum = Nothing : enlacenulo = False : plantillanula = False

    End Sub
    Sub selekval(ByVal temati As String)
        Dim dbconnection As OleDbConnection
        Dim dbdataset As DataSet
        Dim dbdataadapter As OleDbDataAdapter
        Dim cadenaconexion As String
        Dim archivodatos As String
        Dim nombretabla As String
        Dim sentencia As String

        Dim camGEN, camSUP, camDAT As String
        camGEN =
Autodesk.AutoCAD.ApplicationServices.Application.Preferences.Files.AutoSavePath
.ToString
        camSUP =
Autodesk.AutoCAD.ApplicationServices.Application.Preferences.Files.TemplateDwgP
ath.ToString

        'Extraccion de valores para consultar el valor de la etiqueta
        archivodatos = "SOPORTE.MDB"
        sentencia = "SELECT * FROM conIndice"
        nombretabla = "Consulta"
        cadenaconexion = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" &
camSUP & "\" & archivodatos & ";"
        dbconnection = New OleDbConnection(cadenaconexion)
        dbconnection.Open()
        dbdataset = New DataSet
        dbdataadapter = New OleDbDataAdapter(sentencia, dbconnection)
        dbdataadapter.Fill(dbdataset, "Consulta")
        dbconnection.Close()
        Dim tabla As System.Data.DataTable
        tabla = dbdataset.Tables("Consulta")
        Dim fila As DataRow
        Dim campoclave As String
        campoclave = "" : textonum = ""
        For Each fila In tabla.Rows
            If fila.Item("indice") = temati Then
                planvin = fila.Item("cplantilla")
                campoclave = fila.Item("ccampoclave")
                textonum = fila.Item("textonum")
            End If
        Next
        dbdataset.Dispose()
        dbdataadapter.Dispose()

        'Valor enlazado con la plantilla de vinculo.
        editlink(objid, planvin)

        'Asignación de los valores para realizar la consulta SQL
        ' que devuelva el valor de la etiqueta
        Dim dbdataset1 As DataSet
        Dim dbdataadapter1 As OleDbDataAdapter
        Dim nomBD, nomTabla, nomCamPlantilla As String
        nomBD = "" : nomTabla = "" : nomCamPlantilla = ""
        sentencia = "SELECT * FROM enlacesBD WHERE cPlantillaVinculo='" &
planvin & "'"
        dbconnection.Open()
        dbdataset1 = New DataSet
        dbdataadapter1 = New OleDbDataAdapter(sentencia, dbconnection)
        dbdataadapter1.Fill(dbdataset1, "Consulta1")
        dbconnection.Close()
```

```
        Dim tabla1 As System.Data.DataTable
        tabla1 = dbdataset1.Tables("Consulta1")
        Dim fila1 As DataRow
        For Each fila1 In tabla1.Rows
            If fila1.Item("cPlantillaVinculo") = planvin Then
                nomBD = fila1.Item("cBaseDeDatos")
                nomTabla = fila1.Item("cTablaBase")
                nomCamPlantilla = fila1.Item("cCampoClave")
            End If
        Next
        dbdataset1.Dispose()
        dbdataadapter1.Dispose()

        'Consulta del valor de la etiqueta
        Dim dbdataset2 As DataSet
        Dim dbdataadapter2 As OleDbDataAdapter

        archivodatos = nomBD
        camDAT = camGEN & "\DATOS"
        nombretabla = "Consulta2"
        cadenaconexion = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" &
camDAT & "\" & archivodatos & ";"
        dbconnection = New OleDbConnection(cadenaconexion)
        dbconnection.Open()
        dbdataset2 = New DataSet

        'Asignación de la consulta en función de si la clave es numérica o
alfanumerica
        If enlacenulo = False Then
            If textonum = "T" Then
                sentencia = "SELECT * FROM [" & nomTabla & "] WHERE " &
nomCamPlantilla & " ='" & etilin & "'"
            Else
                sentencia = "SELECT * FROM [" & nomTabla & "] WHERE " &
nomCamPlantilla & " =" & etilin
            End If

            dbdataadapter2 = New OleDbDataAdapter(sentencia, dbconnection)
            dbdataadapter2.Fill(dbdataset2, "Consulta2")
            dbconnection.Close()

            Dim tabla2 As System.Data.DataTable
            tabla2 = dbdataset2.Tables("Consulta2")
            Dim fila2 As DataRow
            For Each fila2 In tabla2.Rows
                If fila2.Item(nomCamPlantilla) = etilin Then
                    txtEtiqueta = fila2.Item(campoclave)
                End If
            Next
            dbdataset2.Dispose()
            dbdataadapter2.Dispose()
        End If

    End Sub

    Sub creaTexto(ByVal etilin As String)
        'Esta subrutina crea un texto con el valor previamente definido -
etilin-
        Dim mydwg As ApplicationServices.Document
        Dim mydb As DatabaseServices.Database
        mydwg =
ApplicationServices.Application.DocumentManager.MdiActiveDocument
        mydwg.LockDocument(DocumentLockMode.AutoWrite, Nothing, Nothing, True)
        mydb = mydwg.Database
```

```
        Dim myEd As Editor = DocumentManager.MdiActiveDocument.Editor
        'Selección del punto de inserción
        Dim myPoint As Point3d =
DocumentManager.MdiActiveDocument.Editor.GetPoint("Selecciona punto de
inserción...").Value
        Dim ucs As Geometry.Matrix3d = myEd.CurrentUserCoordinateSystem

        'Asignación de las caracteristicas del texto e inserción en el dibujo
        Using myTrans As Transaction = mydb.TransactionManager.StartTransaction
            Dim myBT As BlockTable =
mydb.BlockTableId.GetObject(OpenMode.ForRead)
            Dim myModelSpace As BlockTableRecord =
myBT(BlockTableRecord.ModelSpace).GetObject(OpenMode.ForWrite)
            Dim myText As New DBText()
            myText.TextString = etilin
            myText.Height = CSng(txt_tam.Text)
            myText.Position = myPoint
            myModelSpace.AppendEntity(myText)
            myText.TransformBy(ucs)
            myTrans.AddNewlyCreatedDBObject(myText, True)
            myTrans.Commit()
            myTrans.Dispose()
            myModelSpace.Dispose()
            myBT.Dispose()
        End Using

    End Sub

    Sub editlink(ByVal objcode As ObjectId, ByVal plantvin As String)
        'Esta subrutina devuelve el valor enlazado a una entidad
        ' correspondiente a una plantilla de vinculo establecida
        On Error Resume Next

        'Declaración de variables
        Dim dbk As CAO.DbConnect
        Dim linkt As CAO.LinkTemplate
        Dim objlink As CAO.Link
        Dim objlinks As CAO.Links
        Dim idarray(0 To 0) As Long
        plantillanula = False

        'Consulta de plantillas de vinculo del dibujo
        dbk =
Autodesk.AutoCAD.ApplicationServices.Application.AcadApplication.GetInterfaceOb
ject("CAO.DbConnect.16")
        linkt = dbk.GetLinkTemplates(0).Item(plantvin)
        If Err.Number <> 0 Then
            MsgBox(planvin & " no encontrada")
            etilin = 0
            plantillanula = True
            GoTo done
        End If

        'Consulta de todos los links asociados a la entidad
        Dim ObjIds(0 To 0) As Int32
        ObjIds(0) = objcode.OldId
        objlinks = dbk.GetLinks(linkt, ObjIds, 2, 0)
        If Err.Number <> 0 Then
            GoTo done
        End If
        enlacenulo = False

        'Extracción del valor correspondiente a la plantilla definida
        objlink = objlinks.Item(0)
```

```
        If objlinks.Count = 0 Then
            enlacenulo = True
        Else
            etilin = objlink.KeyValues.Item(0).Value
            etivin = objlink.LinkTemplate.Name
        End If
done:
        dbk = Nothing

    End Sub

    Private Sub cmd_crear_eti_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cmd_crear_eti.Click
        Me.Hide()

        'Declaración de variables
        Dim mydwg As ApplicationServices.Document
        Dim mydb As DatabaseServices.Database
        Dim mytransman As DatabaseServices.TransactionManager
        Dim mytrans As DatabaseServices.Transaction

        mydwg =
ApplicationServices.Application.DocumentManager.MdiActiveDocument
        mydwg.LockDocument(DocumentLockMode.AutoWrite, Nothing, Nothing, True)
        mydb = mydwg.Database
        mytransman = mydwg.TransactionManager
        mytrans = mytransman.StartTransaction

        'Seleccion de la entidad
        Dim myEd As Editor = DocumentManager.MdiActiveDocument.Editor
        Dim myPSR As PromptSelectionResult = myEd.GetSelection
        Dim SS As Autodesk.AutoCAD.EditorInput.SelectionSet = myPSR.Value
        Dim myBT As BlockTable = mydb.BlockTableId.GetObject(OpenMode.ForRead)
        Dim myModelSpace As BlockTableRecord =
myBT(BlockTableRecord.ModelSpace).GetObject(OpenMode.ForWrite)

        'Asignación del tamaño de letra
        Dim tamano, temati As String
        tamano = txt_tam.Text
        temati = cbx_eti.Text

        ' ************* Añadir etiquetas*****************************
        If tamano <> "" And temati <> "" Then
            If myPSR.Status = PromptStatus.OK Then
                If IsNothing(myPSR.Value) = False Then
                    objid = myPSR.Value.GetObjectIds(0)
                    'Obtención del valor de la etiqueta
                    selekval(temati)
                    If enlacenulo = False Then
                        'inserción del texto con el valor de la etiqueta
                        creaTexto(txtEtiqueta)
                    Else
                        If plantillanula = True Then
                            MsgBox("No existe la plantilla necesaria")
                        Else
                            MsgBox("No existe el enlace solicitado")
                        End If
                    End If
                End If
            End If
        End If
        mytrans.Commit()
        mytrans.Dispose()
        mytransman.Dispose()
```

```
        resetini()
        Me.Show()
    End Sub

    Private Sub frm_etiquetas_Load(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
        'Asignación de valores al desplegable con los campos consultables
        Dim dbconnection As OleDbConnection
        Dim dbdataset As DataSet
        Dim dbdataadapter As OleDbDataAdapter
        Dim cadenaconexion As String
        Dim archivodatos As String
        Dim nombretabla As String
        Dim sentencia As String
        Dim camGEN, camSUP As String
        resetini()
        camGEN =
Autodesk.AutoCAD.ApplicationServices.Application.Preferences.Files.AutoSavePath
.ToString
        camSUP =
Autodesk.AutoCAD.ApplicationServices.Application.Preferences.Files.TemplateDwgP
ath.ToString

        'Consulta SQL sobre a tabla soporte
        archivodatos = "SOPORTE.MDB"
        sentencia = "SELECT * FROM conIndice ORDER BY indice"
        nombretabla = "Consulta"
        cadenaconexion = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" &
camSUP & "\" & archivodatos & ";"
        dbconnection = New OleDbConnection(cadenaconexion)
        dbconnection.Open()

        dbdataset = New DataSet
        dbdataadapter = New OleDbDataAdapter(sentencia, dbconnection)
        dbdataadapter.Fill(dbdataset, "Consulta")
        dbconnection.Close()
        Dim tabla As System.Data.DataTable
        tabla = dbdataset.Tables("Consulta")
        Dim fila As DataRow
        'Asignacion de valores
        For Each fila In tabla.Rows
            Me.cbx_eti.Items.Add(fila.Item("indice"))
        Next

    End Sub


End Class
```

7.18 Código fuente comentado del comando vermapa


```
Public Class frm_Mapa
    Private Sub cmd_asoexc_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cmd_asoexc.Click
        Dim form3 As New frm_asoexc
        form3.Show()
    End Sub

    Private Sub cmd_asoint_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cmd_asoint.Click
        Dim form1 As New frm_asoint
        form1.Show()
    End Sub

    Private Sub cmd_asoext_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cmd_asoext.Click
        Dim form2 As New frm_asoext
        form2.Show()
    End Sub

    Private Sub cmd_consim_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cmd_consim.Click
        Dim form4 As New frm_consim
        form4.Show()
    End Sub

    Private Sub cmd_conmul_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cmd_conmul.Click
        Dim form5 As New frm_conmul
        form5.Show()
    End Sub

    Private Sub cmd_contem_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cmd_contem.Click
```

```
        Dim form6 As New frm_contem
        form6.Show()
    End Sub

    Private Sub cmd_objinf_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cmd_objinf.Click
        Dim form7 As New frm_objinf
        form7.Show()
    End Sub

    Private Sub cmd_resdib_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles cmd_resdib.Click
        Dim form8 As New frm_resdib
        form8.Show()
    End Sub

End Class
```

7.19 Código fuente comentado del comando instalaSIM

```
Public Sub instalaSIM()
        'Definición de variables a utilizar
        Dim mydwg As ApplicationServices.Document
        Dim mydb As DatabaseServices.Database
        Dim mytransman As DatabaseServices.TransactionManager
        Dim mytrans As DatabaseServices.Transaction

        mydwg =
ApplicationServices.Application.DocumentManager.MdiActiveDocument
        mydb = mydwg.Database
        mytransman = mydwg.TransactionManager
        mytrans = mytransman.StartTransaction

        Dim myEd As Editor = DocumentManager.MdiActiveDocument.Editor

        Dim file_instalacion, dir_instalacion, autoguardado, soporte_caminos,
soporte, origendatos As String
        dir_instalacion = "" : file_instalacion = ""

        'Indicamos la ruta del archivo
        Dim ofd As System.Windows.Forms.OpenFileDialog = New
System.Windows.Forms.OpenFileDialog()
        ofd.Filter = "Archivo de configuracion(config.path)|*.path"
        If ofd.ShowDialog() = DialogResult.OK Then
            file_instalacion = ofd.FileName
            dir_instalacion = System.IO.Path.GetDirectoryName(file_instalacion)
        End If


        'Guardar valores que se reasignaran al desinstalar el sistema
        Dim myfsw As New IO.StreamWriter(file_instalacion)
        autoguardado =
Autodesk.AutoCAD.ApplicationServices.Application.Preferences.Files.autosavepath
        myfsw.WriteLine("autoguardado") : myfsw.WriteLine(autoguardado)
        soporte_caminos =
Autodesk.AutoCAD.ApplicationServices.Application.Preferences.Files.supportpath
        myfsw.WriteLine("soporte_caminos") : myfsw.WriteLine(soporte_caminos)
        soporte =
Autodesk.AutoCAD.ApplicationServices.Application.Preferences.Files.TemplateDwgP
ath.ToString
        myfsw.WriteLine("soporte") : myfsw.WriteLine(soporte)
        origendatos =
Autodesk.AutoCAD.ApplicationServices.Application.Preferences.Files.workspacePat
h
        myfsw.WriteLine("origendatos") : myfsw.WriteLine(origendatos)


        myfsw.Close()
        myfsw.Dispose()

        'Asignar nuevas variables

Autodesk.AutoCAD.ApplicationServices.Application.Preferences.Files.autosavepath
= dir_instalacion

Autodesk.AutoCAD.ApplicationServices.Application.Preferences.Files.TemplateDwgP
ath = dir_instalacion & "\support"
```

```
Autodesk.AutoCAD.ApplicationServices.Application.Preferences.Files.supportpath
= dir_instalacion & "\support;" & soporte_caminos

Autodesk.AutoCAD.ApplicationServices.Application.Preferences.Files.workspacePat
h = dir_instalacion & "\support"

        'Actualizacion de variables
        autoguardado =
Autodesk.AutoCAD.ApplicationServices.Application.Preferences.Files.autosavepath
        soporte_caminos =
Autodesk.AutoCAD.ApplicationServices.Application.Preferences.Files.supportpath
        soporte =
Autodesk.AutoCAD.ApplicationServices.Application.Preferences.Files.TemplateDwgP
ath.ToString
        origendatos =
Autodesk.AutoCAD.ApplicationServices.Application.Preferences.Files.workspacePat
h


        'Asignar alias en el enlace de archivos
        Dim mapApi As MapApplication =
Autodesk.Gis.Map.HostMapApplicationServices.Application
        Dim aliases As Aliases = mapApi.Aliases
        Dim numalias, posint, posext, posexc As Integer
        Dim camINT, camEXT, camEXC As String
        posint = 0 : posext = 0 : posexc = 0
        numalias = aliases.AliasesCount
        For x = 0 To numalias - 1
            If aliases.Item(x).Name = "SIGINT" Then posint = 1
            If aliases.Item(x).Name = "SIGEXT" Then posext = 1
            If aliases.Item(x).Name = "SIGEXC" Then posexc = 1
        Next x

        camINT = autoguardado + "\ESPACIOS"
        camEXT = autoguardado + "\VOLUMENES"
        camEXC = autoguardado + "\EXCAVACIONES"
        If posint = 0 Then aliases.AddAlias("SIGINT", camINT)
        If posext = 0 Then aliases.AddAlias("SIGEXT", camEXT)
        If posexc = 0 Then aliases.AddAlias("SIGEXC", camEXC)


        'Cargar menu SIM.CUIX
        Dim cuiname As String = "vbnet_SIM"
        Dim barname As String = "SIM_VB"
        Dim cuifile As String = soporte & "\vbnet_SIM_parcial.cuix"
        Try

Autodesk.AutoCAD.ApplicationServices.Application.MenuGroups.Item(cuiname)
        Catch ex As System.Exception

Autodesk.AutoCAD.ApplicationServices.Application.MenuGroups.Load(cuifile)

Autodesk.AutoCAD.ApplicationServices.Application.MenuGroups.Item(cuiname)
        End Try

        'Configurar ficheros UDL para enlazar datos
        Dim nombd, nombdpolicro, nombdconstruc, nombdhistoria As String
        nombd = "\plantilla.udl"
        nombdpolicro = "policromias" : nombdconstruc = "construc" :
nombdhistoria = "historia"

        ' Hay que crear un fichero binario para que funciones. Con ASCCI no
funciona
```

```
        Dim sStr As String
        Dim bStr() As Byte
        'Creación del fichero UDL con la base de datos de POLICROMIAS
        FileOpen(1, origendatos & "\" & nombdpolicro & ".udl",
Microsoft.VisualBasic.OpenMode.Binary, OpenAccess.Write)
        FilePut(1, CByte(&HFFS), 1)
        FilePut(1, CByte(&HFES), 2)
        sStr = "[oledb]" & vbCrLf
        bStr = System.Text.UnicodeEncoding.Unicode.GetBytes(sStr)
        FilePut(1, bStr)
        sStr = "; Everything after this line is an OLE DB initstring" & vbCrLf
        bStr = System.Text.UnicodeEncoding.Unicode.GetBytes(sStr)
        FilePut(1, bStr)
        sStr = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" & _
               autoguardado & "\DATOS\" & nombdpolicro & ".mdb"
        bStr = System.Text.UnicodeEncoding.Unicode.GetBytes(sStr)
        FilePut(1, bStr)
        FileClose(1)

        'Creación del fichero UDL con la base de datos de HISTORIA
        FileOpen(1, origendatos & "\" & nombdhistoria & ".udl",
Microsoft.VisualBasic.OpenMode.Binary, OpenAccess.Write)
        FilePut(1, CByte(&HFFS), 1)
        FilePut(1, CByte(&HFES), 2)
        sStr = "[oledb]" & vbCrLf
        bStr = System.Text.UnicodeEncoding.Unicode.GetBytes(sStr)
        FilePut(1, bStr)
        sStr = "; Everything after this line is an OLE DB initstring" & vbCrLf
        bStr = System.Text.UnicodeEncoding.Unicode.GetBytes(sStr)
        FilePut(1, bStr)
        sStr = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" & _
               autoguardado & "\DATOS\" & nombdhistoria & ".mdb"
        bStr = System.Text.UnicodeEncoding.Unicode.GetBytes(sStr)
        FilePut(1, bStr)
        FileClose(1)

        'Creación del fichero UDL con la base de datos de CONSTRUC
        FileOpen(1, origendatos & "\" & nombdconstruc & ".udl",
Microsoft.VisualBasic.OpenMode.Binary, OpenAccess.Write)
        FilePut(1, CByte(&HFFS), 1)
        FilePut(1, CByte(&HFES), 2)
        sStr = "[oledb]" & vbCrLf
        bStr = System.Text.UnicodeEncoding.Unicode.GetBytes(sStr)
        FilePut(1, bStr)
        sStr = "; Everything after this line is an OLE DB initstring" & vbCrLf
        bStr = System.Text.UnicodeEncoding.Unicode.GetBytes(sStr)
        FilePut(1, bStr)
        sStr = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" & _
               autoguardado & "\DATOS\" & nombdconstruc & ".mdb"
        bStr = System.Text.UnicodeEncoding.Unicode.GetBytes(sStr)
        FilePut(1, bStr)
        FileClose(1)
        mytrans.Commit()
        'Mensaje de configuración realizada
        MsgBox("Configuración realizada con éxito")
    End Sub
```

7.20 Soporte digital

Junto a esta memoria se adjunta un CD que contiene el proyecto en formato digital.

El contenido del CD es el siguiente:

- **Memoria_pfm_ikoroso.pdf** : Memoria del Proyecto Fin de Máster en formato PDF.
- **Resumen_pfm_ikoroso.pdf**: Resumen del proyecto para su integración en el repositorio de Proyectos Fin de Máster que estará disponible públicamente en la web de la titulación
- Carpeta **SOLUCIONES**: Contiene el desarrollo en código de las herramientas desarrolladas:
    - Carpeta **SIM2D**: Solución que contiene el desarrollo de la aplicación Windows instalable denominada Kategrafia
    - Carpeta **UTIL**: Solución que contiene herramientas para la creación y edición de entidades gráficas, acceso a librerías y acceso a bases de datos mediante la tecnologías ADO.NET.
    - Carpeta **SIM**: Solución que contiene programas para la personalización del Sistema de Información Monumentas de la Catedral de Santa María

El CD se ha incluido en una funda colocada en el interior de la tapa trasera de la encuadernación.