Cuadernos Didácticos

Guía de Diseño y Construcción de Repositorios con Multibase Cosmos

Ingeniería Informática

Cuaderno Nº 18

B. Cristina Pelayo García-Bustelo
Universidad de Oviedo

Juan Manuel Cueva Lovelle
Universidad de Oviedo

Oviedo, Enero 2000

Cuadernos Didácticos
Ingeniería Informática

Cuaderno Nº 18

Guía de Diseño y Construcción de Repositorios con Multibase COSMOS

Autores:

B. Cristina Pelayo García-Bustelo
Universidad de Oviedo – España

J.M. Cueva Lovelle
Universidad de Oviedo - España

Editorial:

SERVITEC

ISBN: 84-699-2081-2

Oviedo, Enero 2000

1ª Edición

Consultor Editorial
Juan Manuel Cueva Lovelle
cueva@lsi.uniovi.es

MULTIBASE

Guía de Diseño y Construcción de Repositorios

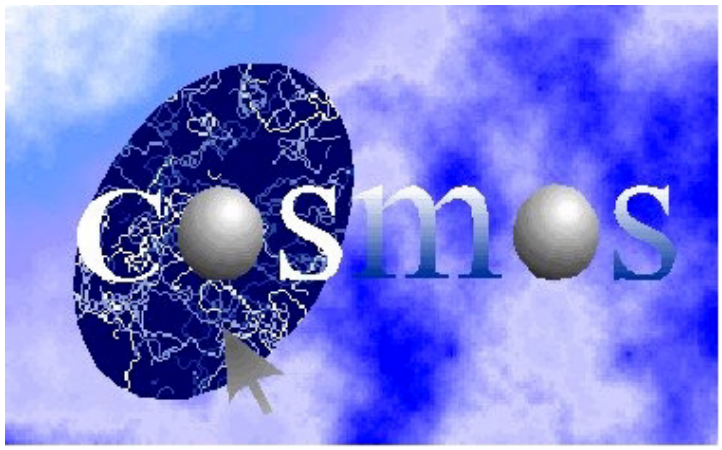
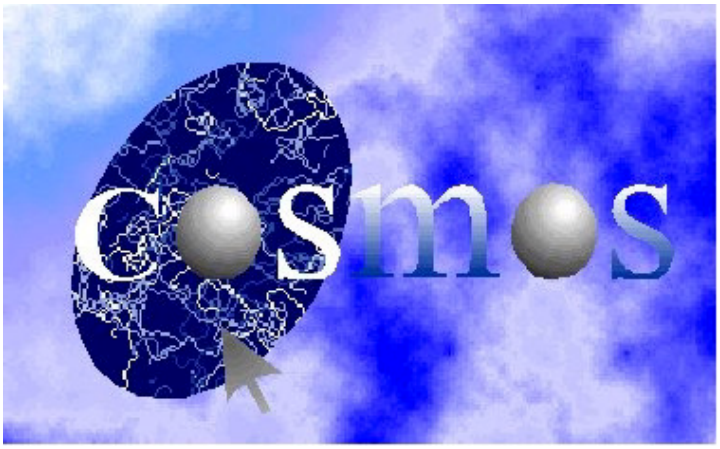


Tabla de Contenidos

Capítulo 1

Primer proyecto con Bases de Datos

1. Introducción.
2. Crear un proyecto.
3. El Editor de Repositorio.
4. Crear una tabla.
5. Añadir columnas a una tabla.
6. Crear una clave primaria.
7. Crear un índice.
8. Crear una base de datos.
9. Añadir un repositorio a un proyecto.
10. Crear un módulo mediante asistentes (Wizards).
11. Ejecución del módulo.
12. Código generado para el módulo.
13. Archivos generados en este capítulo.



1. Introducción.

En este capítulo se van a realizar los pasos necesarios para desarrollar el mantenimiento de los datos de una tabla, desde la creación de la tabla hasta la generación mediante asistentes (Wizards) de un programa con altas, bajas, modificaciones y búsquedas sobre la tabla.

2. Mi primer proyecto con base de datos.

Vamos a comenzar realizando un primer proyecto que contendrá una única tabla que con los datos de una persona. Para ello arrancamos el Editor Visual de Cosmos y seleccionamos del menú desplegable la opción ***File*** y dentro de esta ***New***:

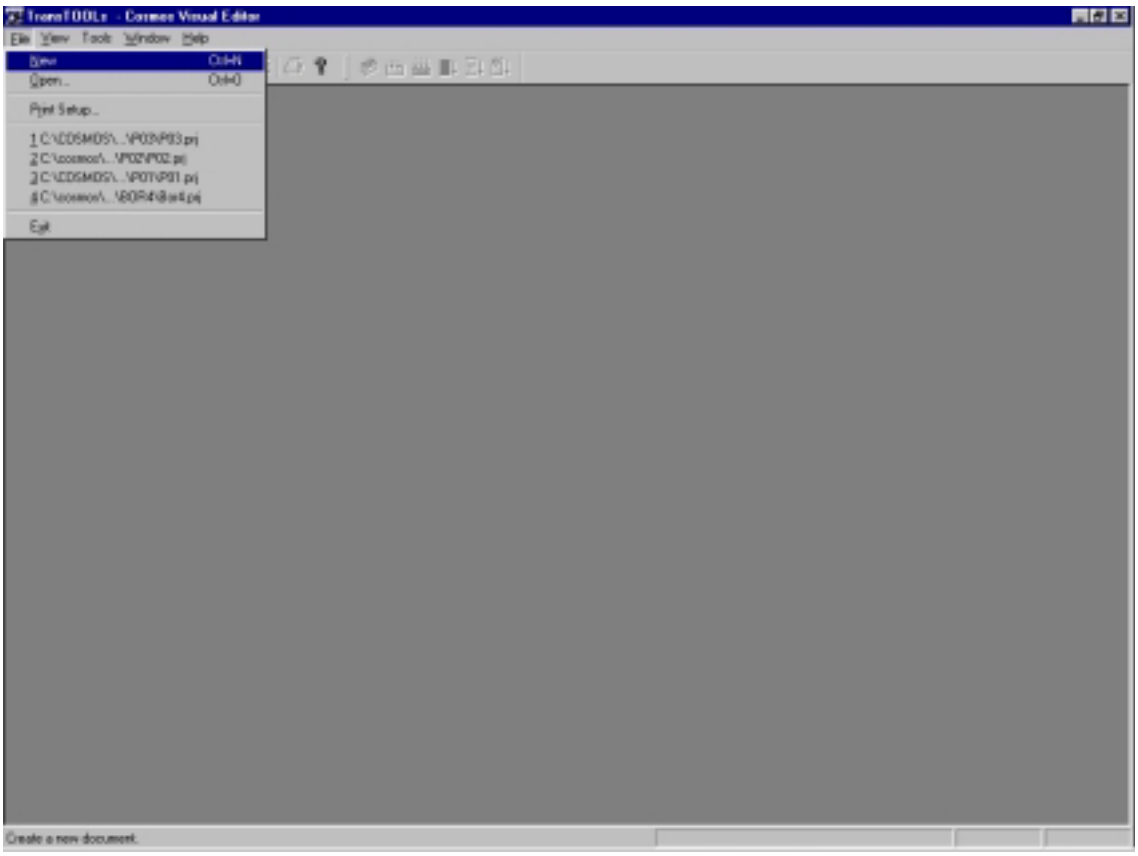


Figura 1.1. Creación de una nueva aplicación.

Aparece un cuadro de diálogo donde tenemos que seleccionar el tipo de documento a crear. Hay tres tipos:

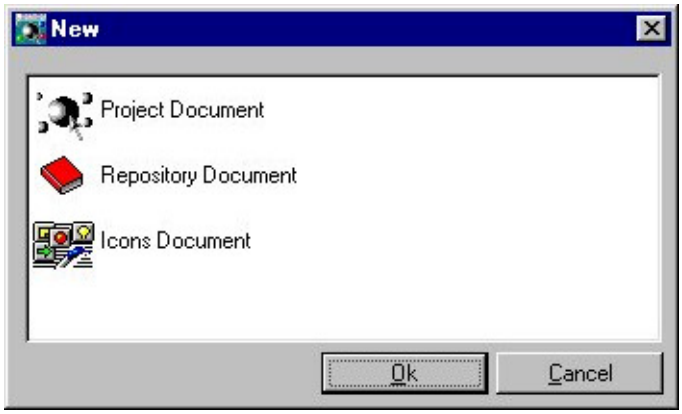


Figura 1.2. Selección del tipo de Documento a crear.

Seleccionamos la primera opción, ***Project Document*** porque vamos a hacer un proyecto que contenga una tabla de una base de datos. (Figura 1.2.)

Ahora se pide el nombre del nuevo proyecto así como el lugar donde se almacenaran todos los ficheros que se van generando (Figura 1.3.). Con el nombre del proyecto se crea una carpeta en la ubicación especificada.

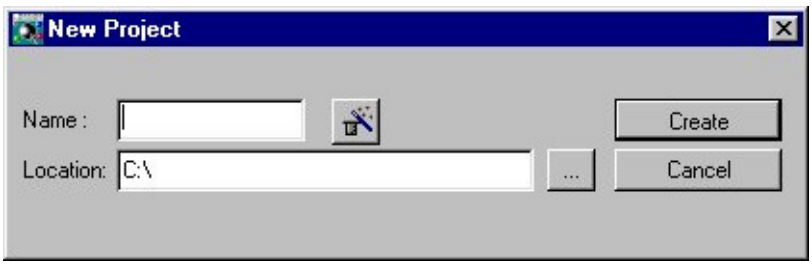


Figura 1.3. Cuadro de dialogo para especificar el nombre y ubicación del proyecto.

Hemos puesto como nombre del nuevo proyecto “MiProyecto” y al pulsar el botón ***Create*** del cuadro de diálogo de la figura 1.3. se genera la paleta de proyecto dentro del editor visual.

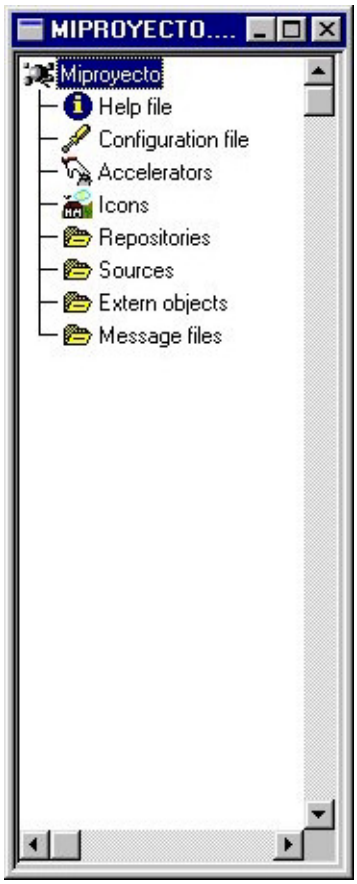


Figura 1.4.Paleta del nuevo proyecto.

En este momento ya tenemos preparado el proyecto para poder asociarle un Repositorio y los módulos de programación que deseemos.

3. El Editor de Repositorio

El repositorio esta compuesto por el conjunto de datos necesarios para definir el esquema de una base de datos. También contiene la información válida para el desarrollo de programas y para su generación automática como veremos posteriormente.

Lo primero que haremos será invocar a este editor para comenzar a utilizar las tablas en Cosmos. Para ello seleccionamos en el Editor Visual, dentro del menú ***Tools*** la opción ***Repository Manager***. (Figura 1.5.)

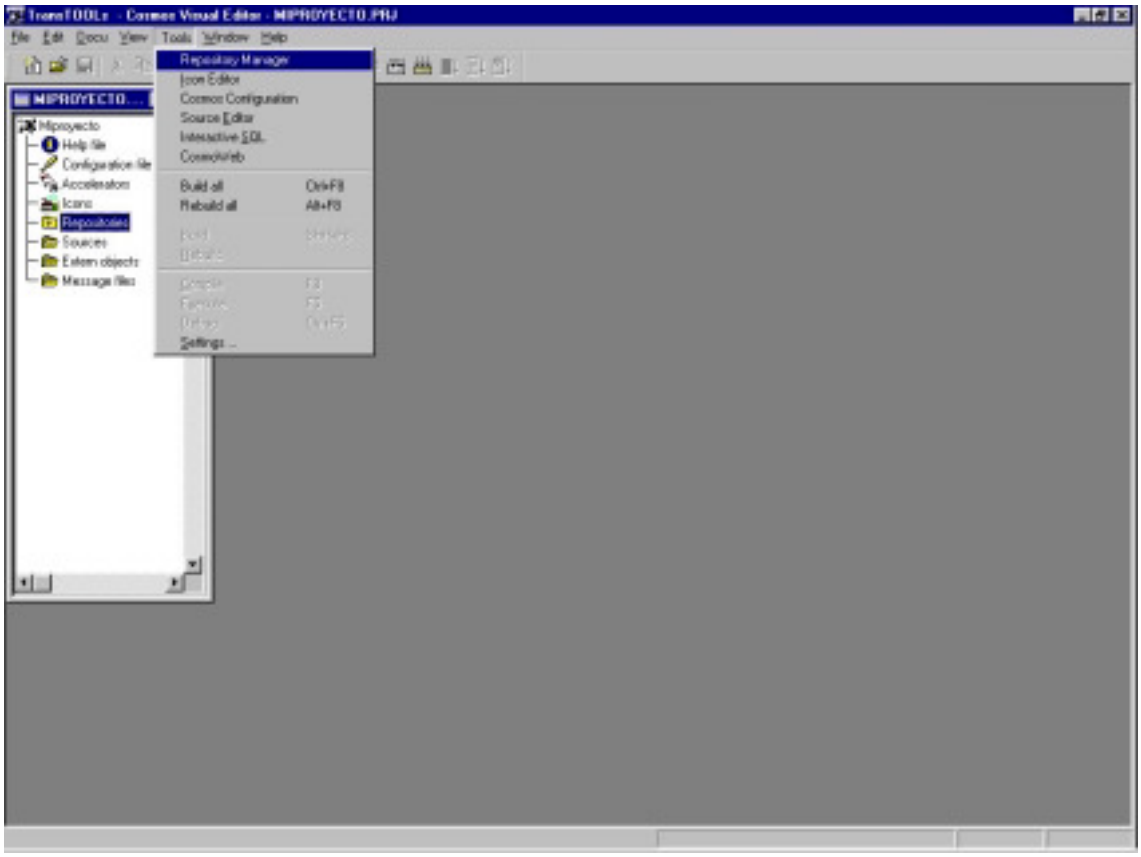


Figura 1.5. Abrir el Editor de Repositorio.

Cuando abrimos en Editor de Repositorio el aspecto que tiene es el de la figura 1.6. Esto nos indica que no tenemos ningún repositorio en ejecución en el momento actual. Ahora vamos a dar todos los pasos necesarios para crear un Repositorio que contenga una tabla.

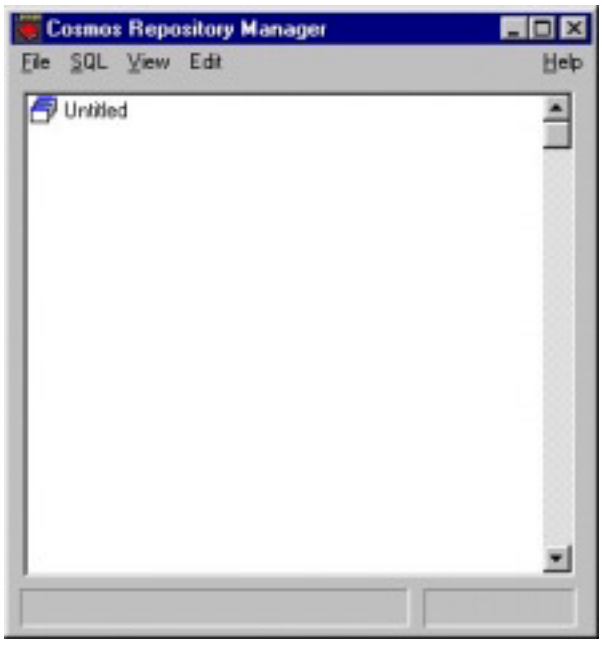


Figura 1.6. Editor de Repositorio.

4. Crear una tabla.

En primer lugar se hace click con el botón secundario del ratón sobre el editor del Repositorio y se selecciona la opción ***New*** para poder crear la primera tabla de datos en nuestro Repositorio.

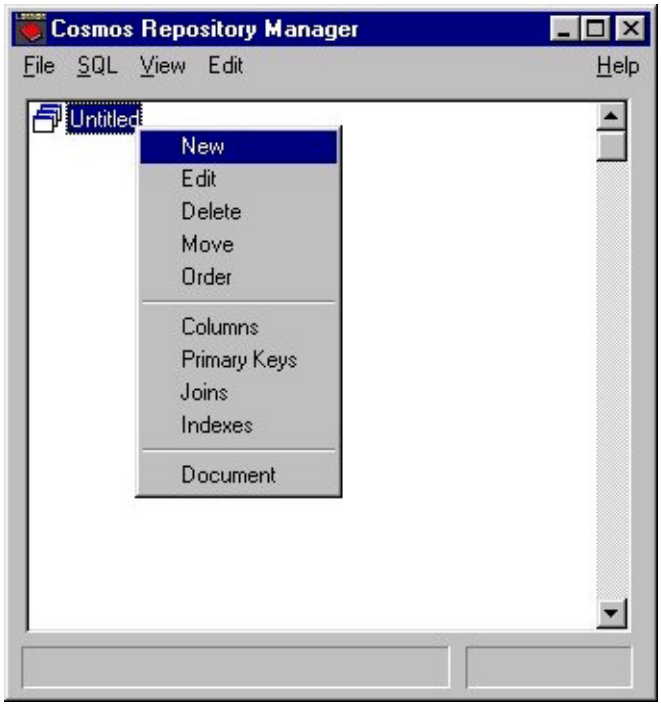


Figura 1.7. Cuadro de diálogo de Edición de una Tabla.

En el cuadro de diálogo que aparece pondremos el nombre de la tabla que debe ser único en el Repositorio actual y su etiqueta. La etiqueta dará información sobre una determinada tabla cuando esta tenga el foco de la aplicación.

También en este cuadro de diálogo podemos definir de que tipo va a ser la nueva tabla, de momento definimos una “Database Table”.

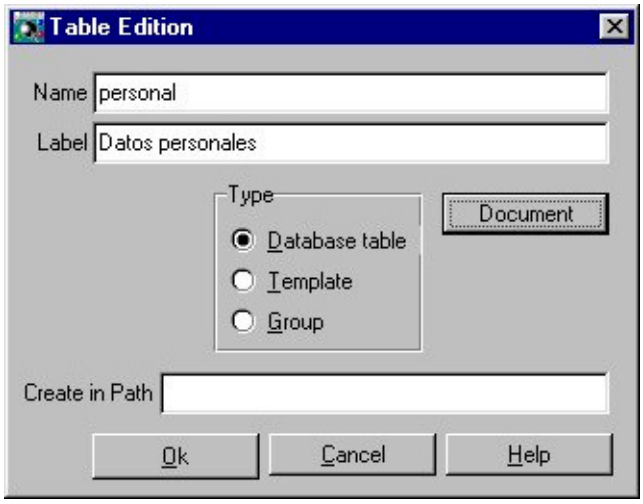


Figura 1.8. Definición de la tabla ”Personal”.

El botón marcado como ***Document*** nos permite definir un texto que luego aparecerá en un fichero con extensión .dfr y con el mismo nombre de nuestro repositorio. Mediante esta utilidad podemos ir documentando nuestro repositorio, ya que se pueden documentar tablas, plantillas, estructuras y columnas, como ya veremos posteriormente. En la tabla “personal” hemos introducido en la sección ***Document*** el texto que se muestra en la figura 1.9:

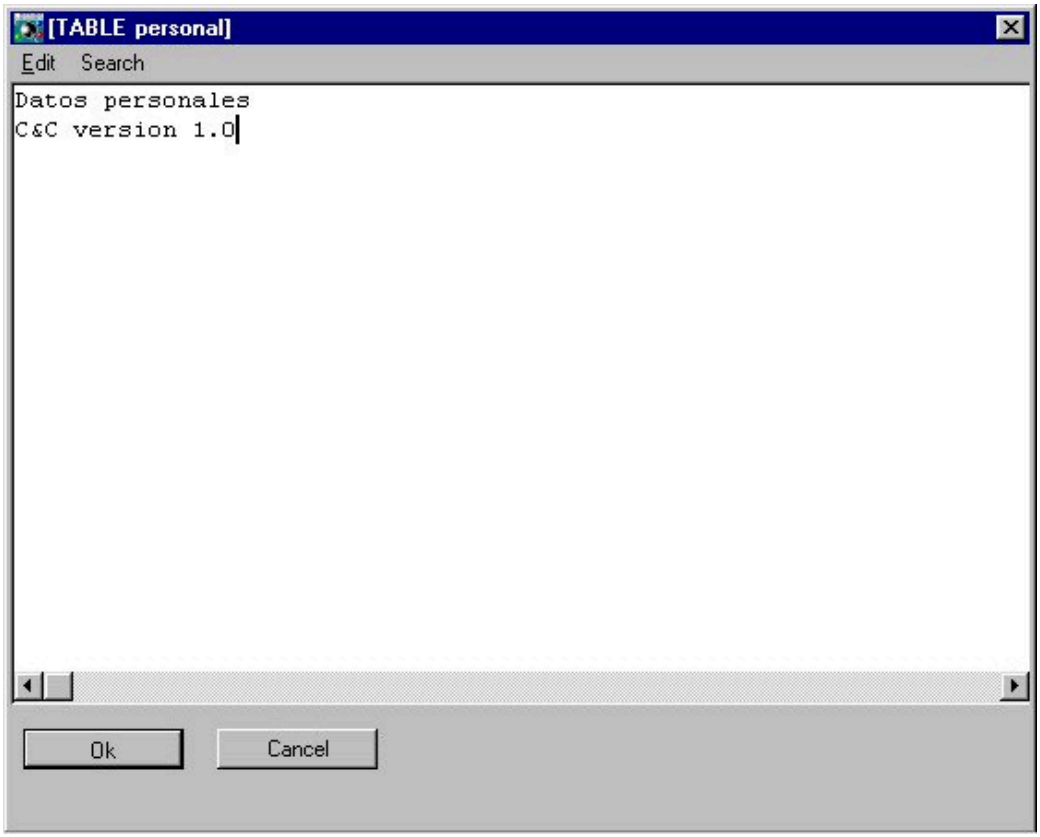


Figura 1.9. Documentación de la tabla “personal”.

Una vez creada esta tabla en la estructura en forma de árbol del Editor de Repositorio aparece el nombre de la misma. (Figura 1.10)

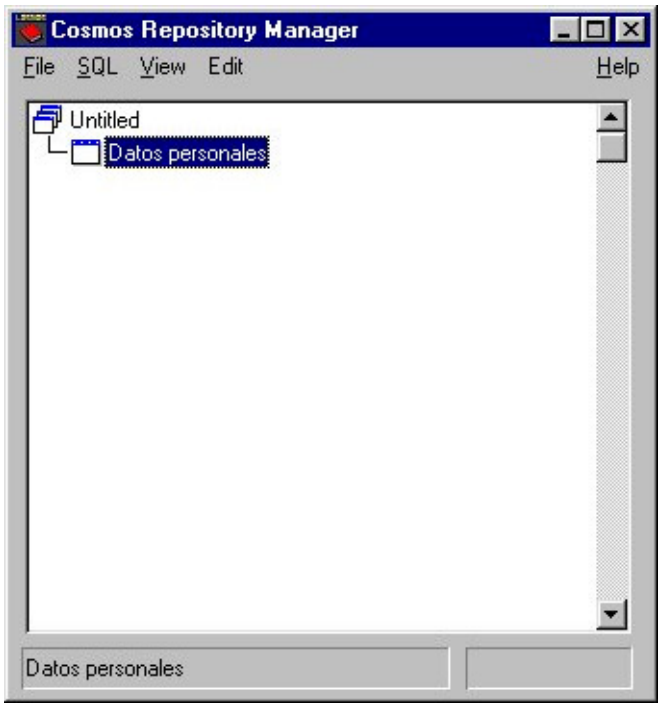


Figura 1.10. Editor de Repositorio con la nueva tabla.

En lugar de aparecer el nombre verdadero de la tabla (“personal”) aparece la etiqueta que le pusimos a esa tabla en su definición y al encontrarse seleccionada en este momento también aparece dicha etiqueta en la parte inferior izquierda del Editor de Repositorio.

En este momento ya estamos dispuestos para crear las diferentes columnas que contendrá la tabla “personal”.

5. Añadir Columnas a una Tabla

Para añadir columnas a la tabla que hemos creado, hacemos doble click sobre la etiqueta “Datos personales” y nos aparece un cuadro de diálogo donde podemos añadir las nuevas columnas.

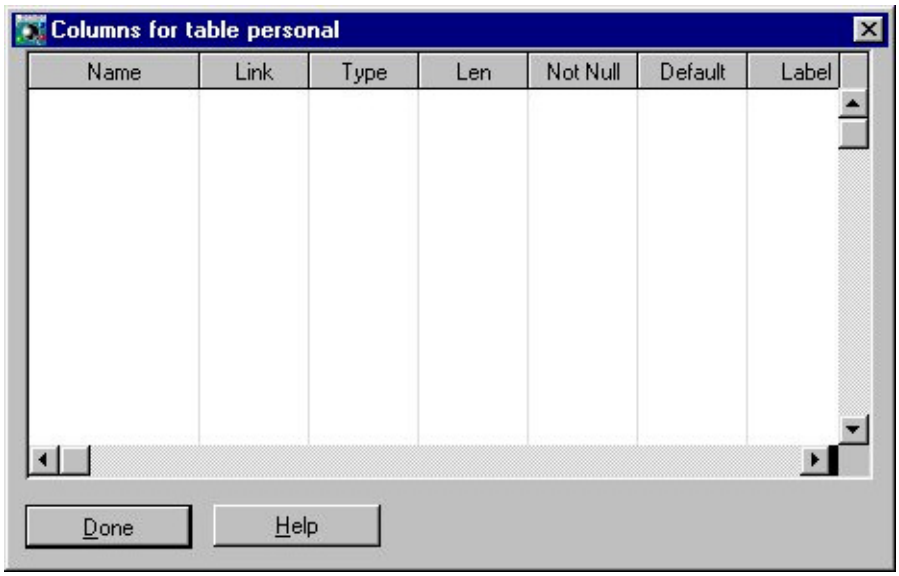


Figura 1.11. Cuadro de diálogo para añadir columnas a la tabla “personal”.

Pulsando sobre este cuadro el botón secundario del ratón aparece un cuadro de opciones del cual seleccionamos ***New***. (Figura 1.12)

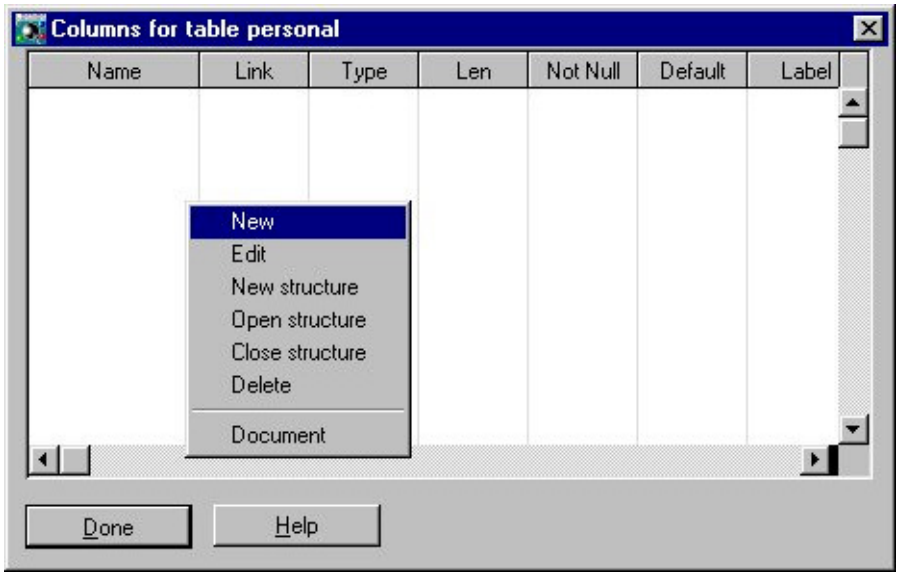


Figura 1.12. Añadir una nueva columna.

En el cuadro de dialogo que se muestra en la figura 1.13. se deben añadir todas las características que deseamos tenga la nueva columna que estamos añadiendo.

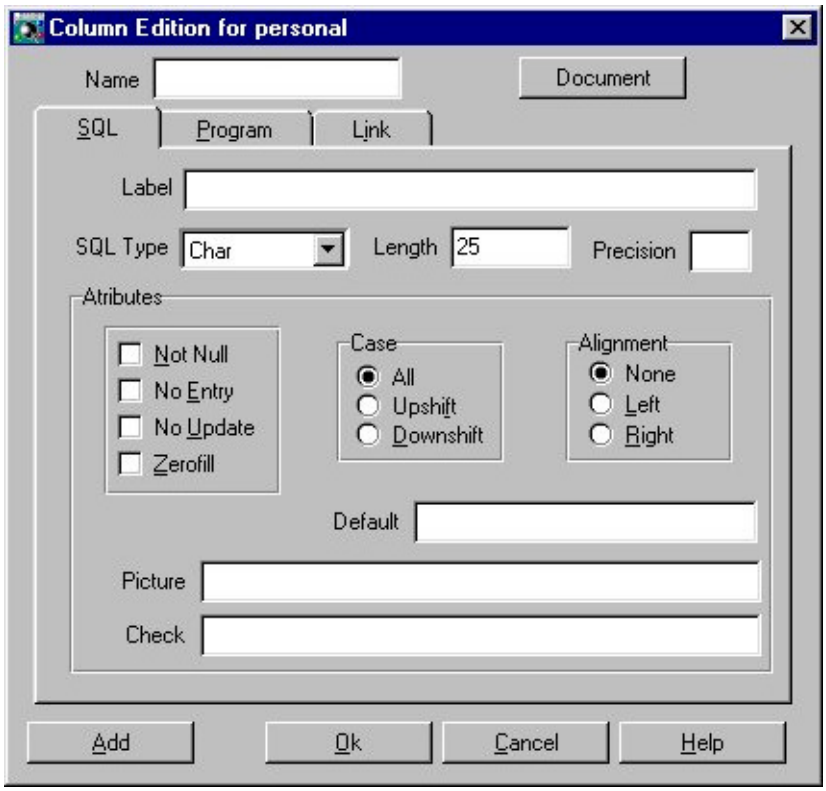


Figura 1.13. Añadir una columna a la tabla “personal”.

En nuestro ejemplo vamos a crear cuatro columnas cuyo nombre y características son las siguientes:

Columna	Tipo	Longitud	Atributos
dni	Char	9	not null
nombre	Char	25	
apellidos	Char	30	
fecha_nacimiento	Date	4	

El atributo ***not null*** de la columna “dni” indica que este campo no puede dejarse vacío en la introducción de los datos en la tabla, es decir la columna debe tener un valor distinto de null.

Cuando hemos introducido la definición de la primera columna, en este caso “dni” pulsamos el botón ***Add*** y nos añadirá esa columna y vuelve a mostrar el cuadro de diálogo de la figura 1.13. para seguir añadiendo columnas sin necesidad de repetir los pasos dados previamente. En el momento que hemos añadido todas las columnas podemos pulsar ***Cancel*** para salir de este cuadro de diálogo y se muestra las columnas que hemos definido para la tabla “personal”.

Figura 1.14. Columnas de la tabla “personal”.

6. Crear una clave primaria.

Una clave primaria es un conjunto de columnas, ninguna de las cuales ha de admitir valores nulos (NOT NULL), que va a identificar de forma unívoca cada una de las filas de la tabla. Una tabla admite únicamente una clave primaria.

Ahora vamos a definir la clave primaria para esta tabla, para ello en el cuadro de diálogo de la figura 1.14. pulsamos el botón ***Done*** y con el botón secundario del ratón pulsamos sobre la tabla “Datos personales” de la estructura del repositorio y seleccionamos la opción correspondiente.

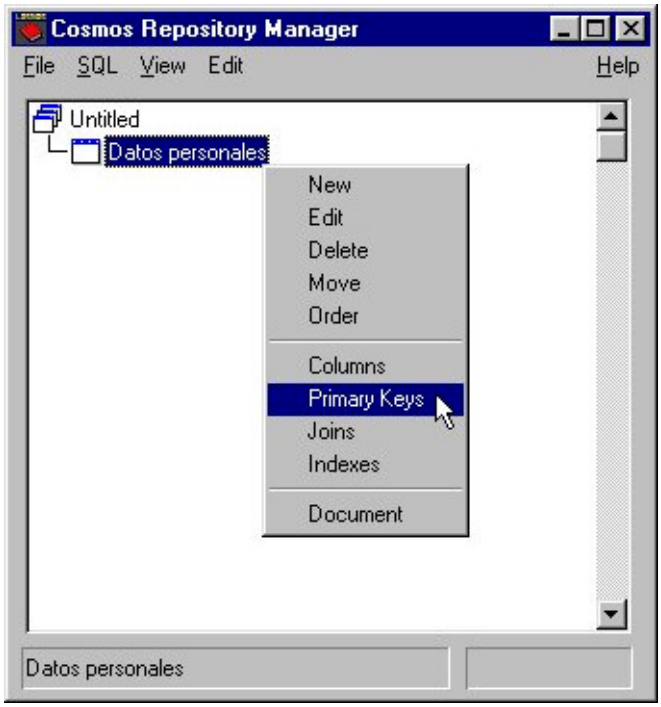


Figura 1.15. Crear una clave primaria.

Aparece un cuadro de diálogo que contiene las columnas de la tabla. Seleccionamos "dni" y pulsando la flecha  aparece el dni en el panel de la derecha, lo que indica que es la clave primaria. Para terminar pulsamos el botón "Ok".

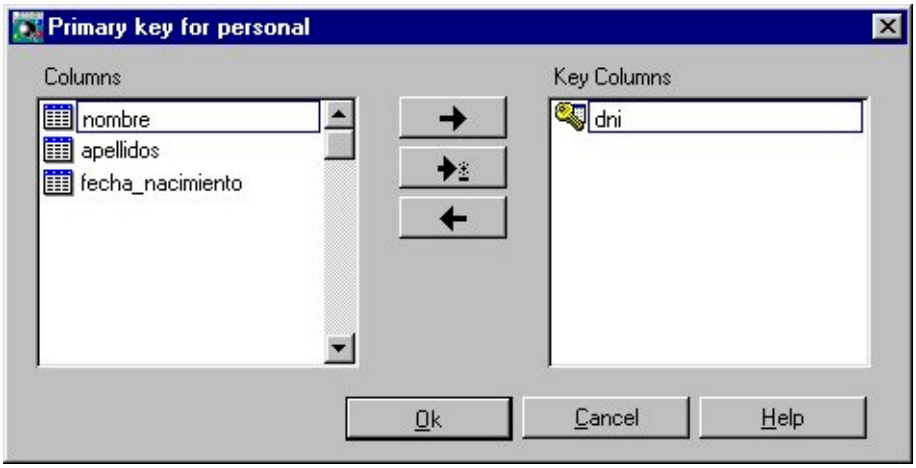


Figura 1.16. Definir dni como clave primaria.

En este caso la clave primaria esta compuesta por una única columna, pero en el caso de que en una tabla necesitemos una clave primaria compuesta por mas de una columna podemos añadirlas mediante la flecha  y mediante  podemos quitar una columna de una clave primaria.

7. Crear un índice.

Un índice es una utilidad manejada por el SQL para el acceso a los datos de las tablas. Los motivos de definición de un índice son fundamentalmente los siguientes:

- Aumentar la velocidad de acceso a la tabla reduciendo el tiempo de respuesta.
- Asegurar la identificación unívoca de filas de una tabla.
- Facilitar los enlaces entre tablas, la ejecución de instrucciones SELECT del SQL y la realización de entradas de datos multitabla mediante el objeto Form del lenguaje de cuarta generación COOL de Cosmos.

Una clave primaria es un índice único. Esto significa que no se puede crear un índice compuesto con la/s misma/s columna/s que la clave primaria.

Ahora vamos a añadir un índice para poder buscar adecuadamente la información dentro de nuestra tabla. Para ello vamos a seleccionar ***Indexes*** dentro de las opciones que se nos presentan cuando pulsamos el botón secundario del ratón sobre el nombre de la tabla. (Figura 1.17)

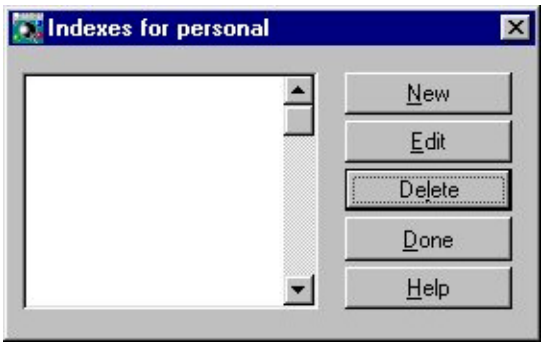


Figura 1.17. Cuadro de diálogo para añadir un índice.

El cuadro de diálogo que aparece nos permite definir un nuevo índice (***New***), modificar uno existente (***Edit***), borrar (***Delete***), o realizar los cambios efectivamente (***Done***).

Pulsamos el botón ***New*** y aparece el cuadro de diálogo que nos permitirá definir los índices deseados. (Figura 1.18)

Figura 1.18.Cuadro de diálogo para crear un índice.

Se debe poner un nombre al índice que estamos creando y si las columnas componentes de un índice pueden tener varios valores iguales en distintas filas de la tabla, entonces a este se le denomina índice duplicado y se marca el botón de radio ***Duplicated***. Por el contrario, si una determinada combinación de valores sólo puede aparecer una vez en la tabla, entonces se llamará índice único y se marcará el botón de radio ***Unique***. En nuestro caso como el apellido de una persona puede coincidir con otro marcamos nuestro índice como duplicado.

Se deben seguir los mismos pasos que para añadir la clave primaria, es decir, seleccionamos la columna “apellidos” y pulsamos  con lo que ya tenemos definido el índice deseado. Los botones  y  funcionan igual que en el caso de la clave primaria.

Para indicar el tipo de ordenación que van a tener la recuperación de los datos se pulsa el botón  y aparece en el lado derecho de la columna que compone el índice el indicador el símbolo ⬆ para ordenación ascendente que es el marcado por defecto o ⬇ para la ordenación descendente. Después de introducir los datos pulsamos la tecla “Ok”.

En este momento ya tenemos definido el repositorio y tenemos que guardarlo. Para ello en el menú desplegable seleccionamos ***File*** y la opción ***Save*** poniendo el nombre que deseamos y el lugar donde queremos almacenarlo.

8. Crear una Base de Datos.

Con la definición del repositorio damos a crear la base de datos asociada. Dentro del Editor de Repositorio seleccionamos la opción ***Create Database*** dentro de la opción ***SQL*** del menú desplegable. (Figura 1.19)

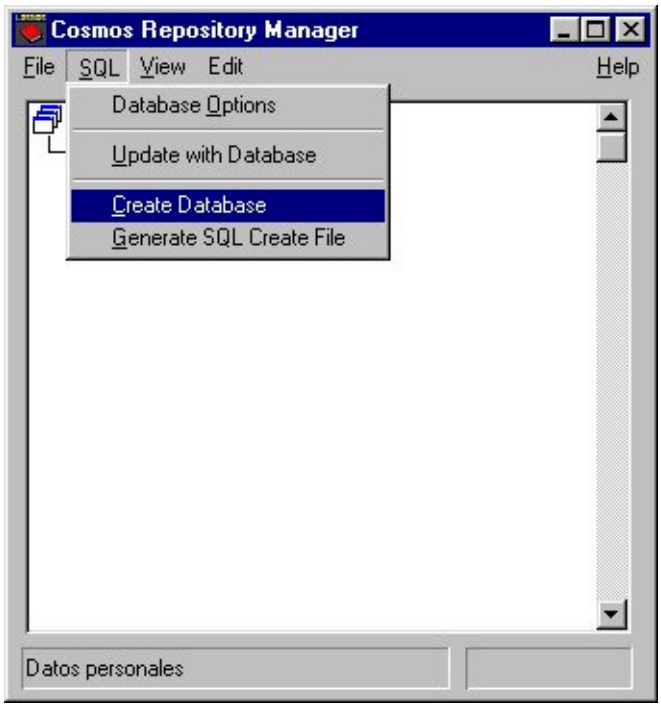


Figura 1.19. Crear la base de datos.

Se nos muestra un cuadro de diálogo donde tendremos que definir o seleccionar la conexión que deseamos utilizar para la base de datos. Esta conexión debe estar definida con el Editor de Configuración de Cosmos, para mas detalles ver la Guía de Referencia de Multibase Cosmos.

En el caso de no tener definida ninguna conexión podemos crearla mediante la pulsación del botón ***Add***. Hemos seleccionado una conexión local llamada “Conexión” y se crea la base de datos llamada “Datos” en el directorio que esta especificado en la variable de entorno DBPATH de la conexión seleccionada. Se creará una carpeta con el nombre de la base de datos donde se almacenarán las tablas del repositorio, los índices y en definitiva todo lo necesario para poder utilizar adecuadamente la base de datos.

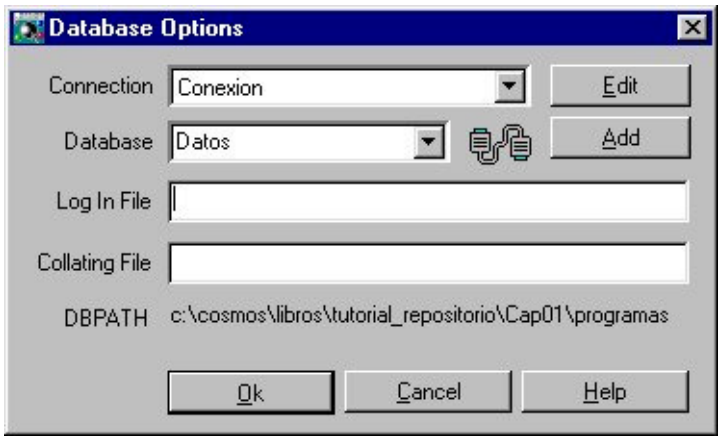


Figura 1.20. Cuadro de diálogo de las opciones de la base de datos.

Si se añade una ruta y un nombre de fichero en el campo edit marcado con el nombre “Log In File” se almacenará un histórico que se utilizará para el tratamiento de transacciones. En la definición del “Collating File” se indica el nombre del fichero que almacena la secuencia de ordenación que seguirán los datos según una tabla ASCII. Si se deja en blanco indicamos que deseamos utilizar la ordenación por defecto.

Pulsamos el botón “Ok” y el Editor Repositorio creará la base de datos y cuando termine mostrará el cuadro:

Figura 1.21. Se ha creado la base de datos.

Con esto hemos creado la base de datos y el repositorio y ya esta todo preparado para realizar la primera aplicación sencilla de base de datos.

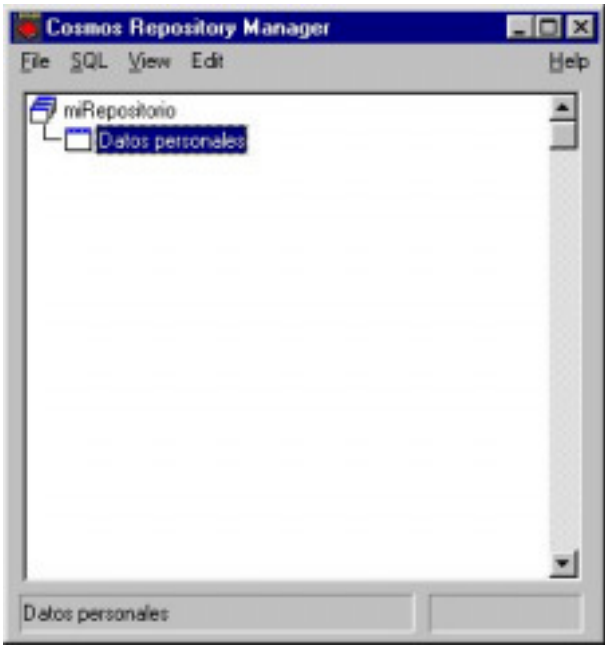


Figura 1.22. El Repositorio creado y su contenido.

9. Añadir un Repositorio a un proyecto.

Ahora añadimos el repositorio al proyecto que se creo en el apartado dos de este capítulo, para ello seleccionamos la sección ***Repositories*** dentro de la paleta de proyecto y pulsando el botón secundario del ratón seleccionamos ***Import***. (Figura 1.23)

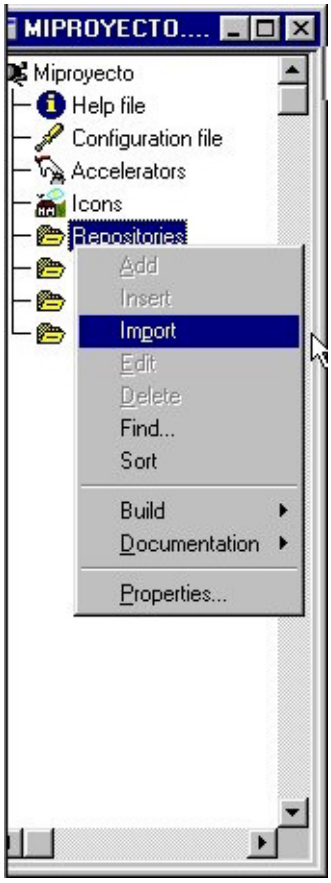


Figura 1.23. Importar el Repositorio.

Rellenamos los campos de Label y comment, para poder seleccionar el repositorio que deseamos importar pulsamos ... y podemos seleccionar el fichero que contiene el repositorio.

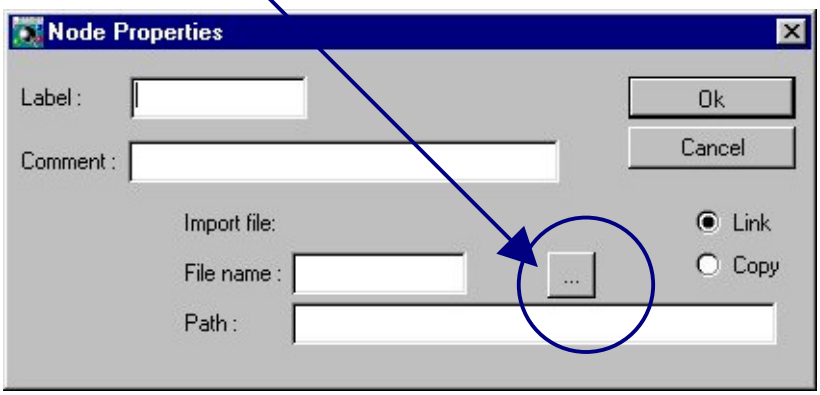


Figura 1.24. Cuadro de diálogo para importar un repositorio.

Terminamos la selección del repositorio pulsando el botón “Ok”.

10. Crear un módulo mediante asistentes (Wizards).

Ahora vamos a hacer uso de una de las facilidades de programación que nos da Cosmos, los asistentes (Wizards). Hay varios tipos de asistentes (Wizards):

- Wizard de proyecto: A partir de un repositorio crea un proyecto nuevo que contiene: Módulos para el mantenimiento de las tablas del repositorio (de todas o de las que haya seleccionado previamente) y módulos de listados de las mismas. Genera un menú como módulo de entrada al proyecto con una persiana por cada tipo de programa y una opción para cada módulo generado.

- Wizard de Módulos: A partir de una o más tablas del repositorio creará un módulo que implementa: El mantenimiento de las tablas elegidas y/o un listado de las mismas.

- Wizard de Forms: A partir de una o más tablas del repositorio creará una clase Form que implemente el mantenimiento de las tablas elegidas.

- Wizard de Pages: A partir de una o más tablas del repositorio creará una clase Page que implemente el listado de las tablas elegidas.

Vamos a utilizar el Wizard de módulos, para ello seleccionamos dentro de la paleta del proyecto en el Editor Visual, la sección Sources y pulsamos el botón secundario del ratón mostrando entonces una serie de opciones de las cuales escogemos ***Add***. (Figura 1.25)

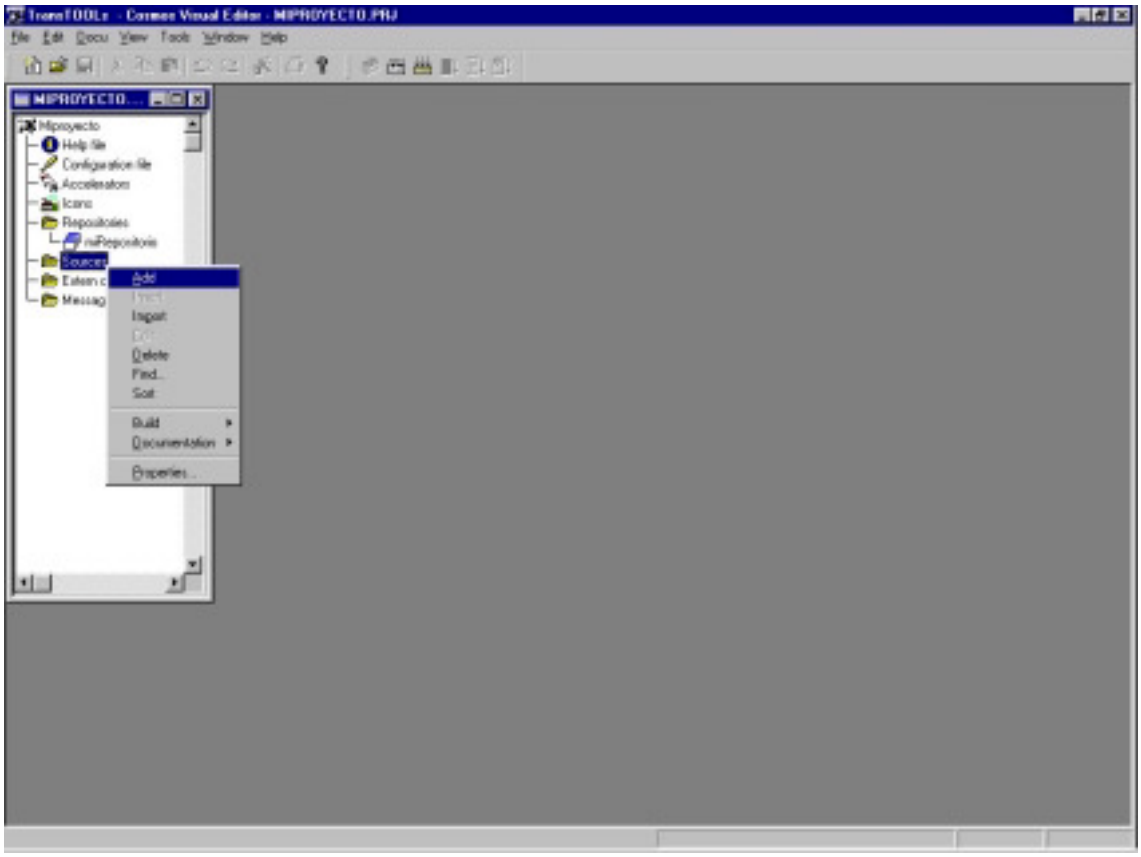


Figura 1.25. Añadir un módulo al proyecto.

En el cuadro de diálogo que se muestra en la figura 1.26. debemos introducir el nombre del módulo que deseamos añadir, y el nombre del fichero que lo almacenará. El módulo actual será el que nos permitirá mantener la tabla, realizando **a**ltas, **b**ajas y **m**odificaciones sobre la misma, por eso el nombre del módulo será “ABM”. También podremos realizar búsquedas sobre cualquiera de los campos de la tabla.

Después pulsamos el botón  lo que nos permite utilizar el Wizard de módulos y mediante este Wizard podremos realizar toda la funcionalidad especificada anteriormente.

Figura 1.26. Cuadro de diálogo de las propiedades de un módulo.

Lo primero que se nos pide es el repositorio que queremos utilizar para el módulo y el nombre de la clase que se generará. Seleccionamos el único repositorio que tenemos y dejamos el nombre la clase por defecto que es “CABM” que significa “Clase ABM”

Ahora vamos a seguir los pasos marcados por el Wizard donde tendremos que seleccionar el tipo de Form que deseamos utilizar para el mantenimiento de la tabla. Seleccionamos un ***Standard Form*** y pulsamos el botón  para continuar. (Figura 1.27)

Figura 1. 27. Seleccionar el tipo de módulo.

En este momento tenemos que seleccionar la tabla que queremos asociar al módulo, como en nuestro repositorio únicamente tenemos una tabla “personal”.

En cualquier momento podemos volver atrás sobre nuestros pasos pulsando la tecla << Back si queremos modificar alguna de las selecciones realizadas hasta el momento. También mediante la tecla Cancel podemos anular la creación del módulo.

Seleccionamos en este momento el tipo de Form que queremos utilizar en el módulo, podemos seleccionar seis tipos diferentes, seleccionamos Form mediano con barra de estado. (Figura 1.28)

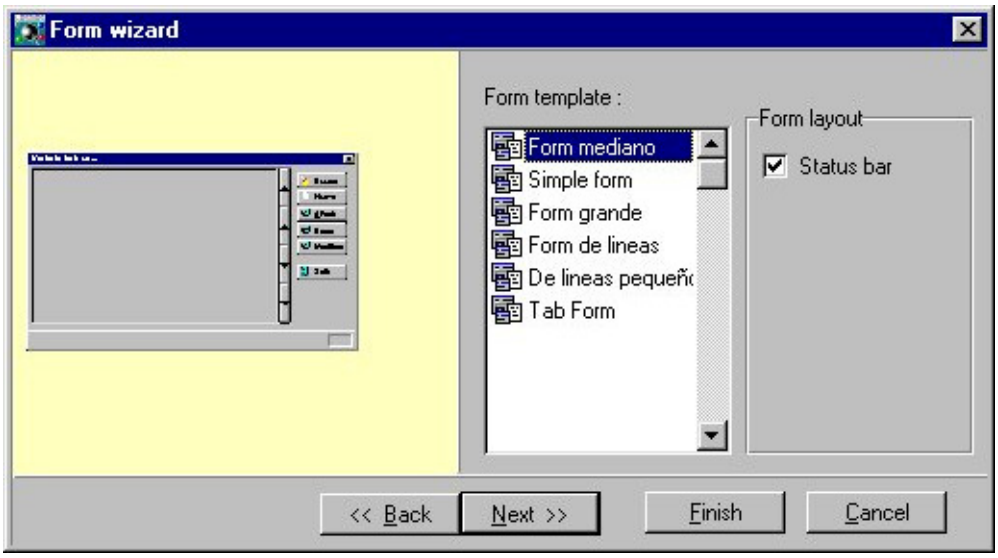


Figura 1.28. Seleccionar tipo de Form.

Después de pulsar el botón Next >> se nos muestra un cuadro de diálogo donde tenemos que seleccionar y añadir todas las columnas de la tabla en las columnas de la Form, mediante . Pulsamos nuevamente la tecla Next >>.

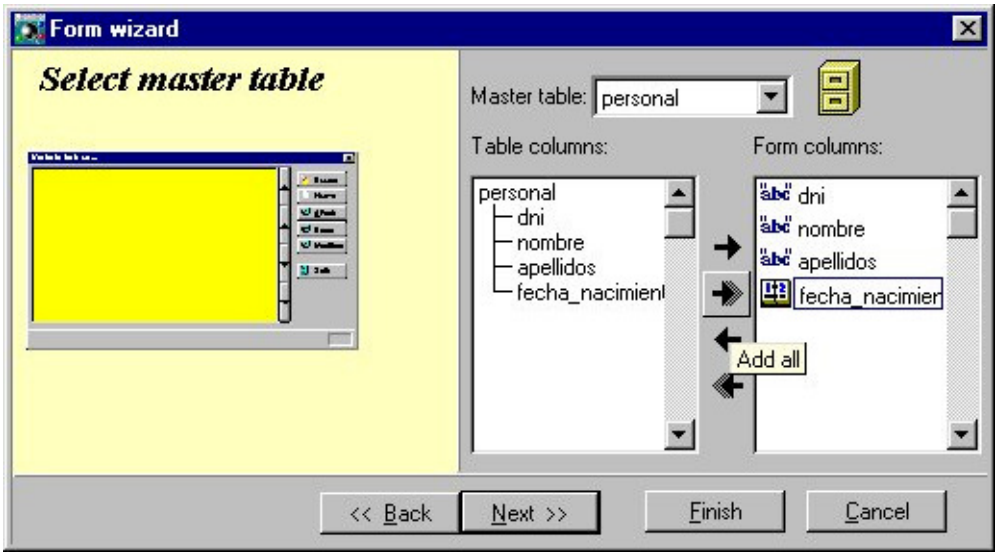


Figura 1.28. Seleccionamos las columnas.

Si deseamos que el Wizard realice la conexión con la base de datos para el módulo que estamos creando, seleccionamos la casilla ***Establish SQL connection on this module*** y aparece un cuadro de diálogo (Figura 1.29) donde seleccionamos la conexión que tenemos definida mediante el Editor de Configuración y la base de datos que hemos creado en los pasos anteriores.

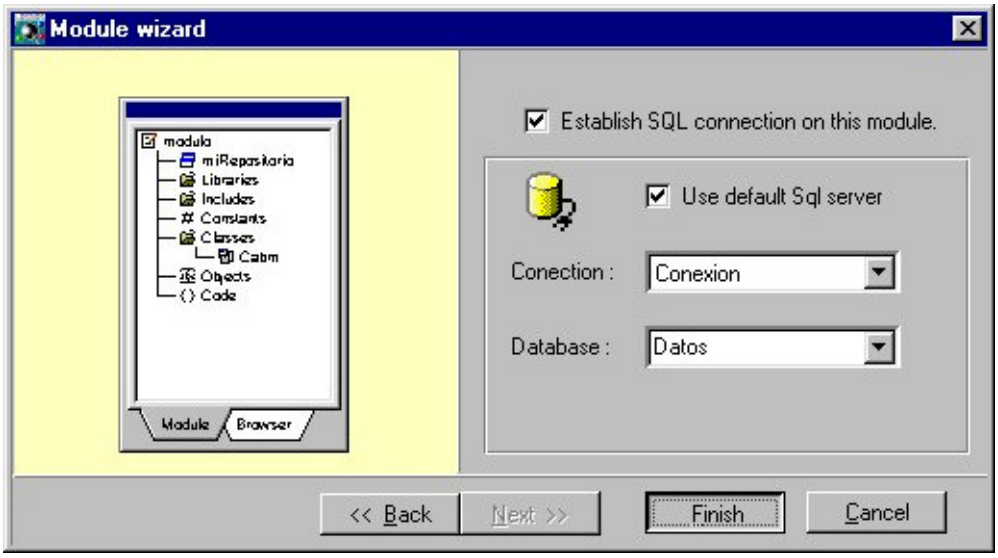


Figura 1.29. Establecer una conexión.

En este momento ya hemos terminado de introducir los datos necesarios al Wizard para que se cree el módulo, ahora pulsamos el botón Finish para que se realice la creación del módulo efectivamente.

En la paleta del proyecto aparece el nuevo módulo y si lo abrimos se nos muestra todos los componentes del nuevo módulo. (Figura 1.30)

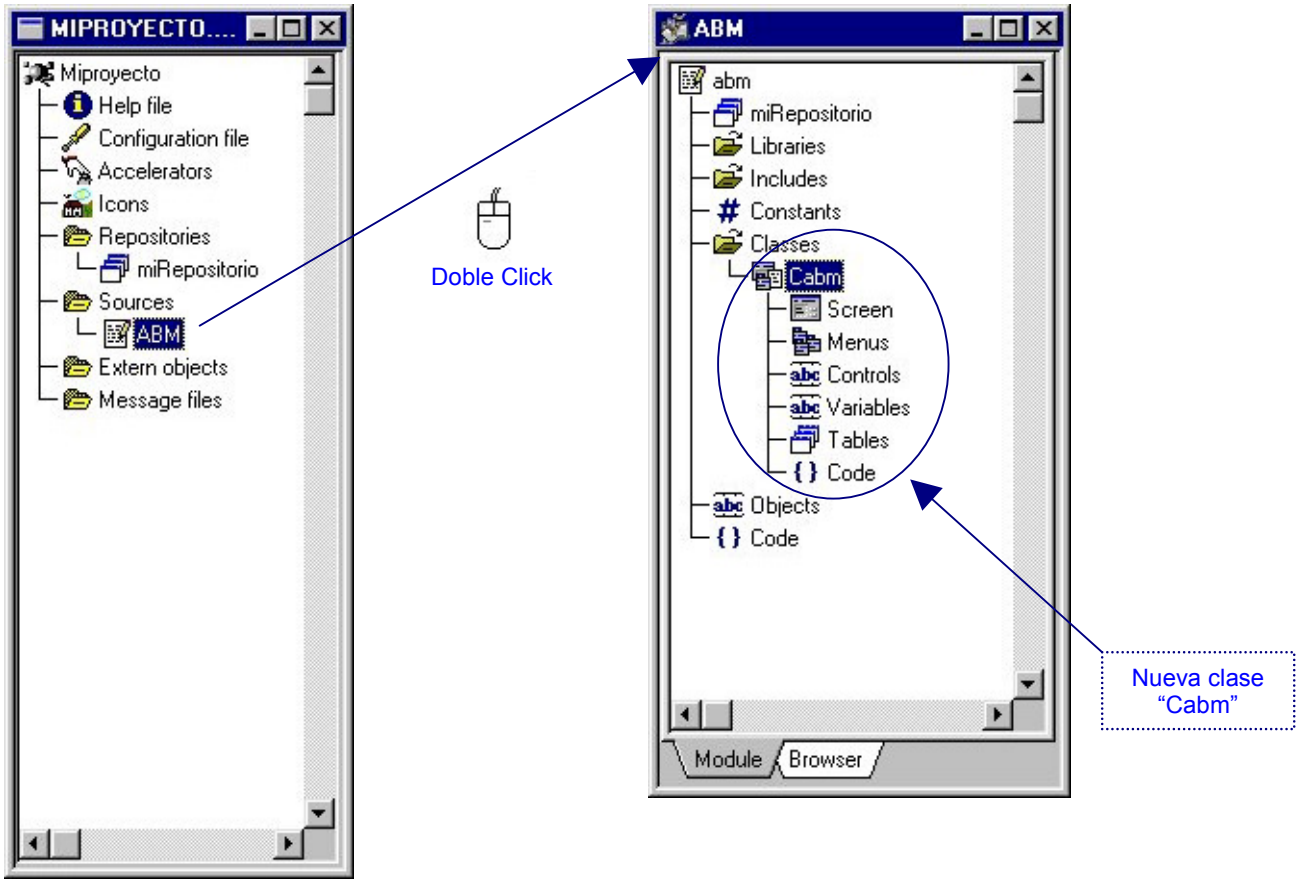


Figura 1.30. Nuevo modulo creado mediante el Wizard.

Si hacemos doble click sobre la seccion ***Screen*** de la nueva clase “Cabm” (Figura 1.30) nos aparece la pantalla que se mostrará en la ejecución de este módulo.

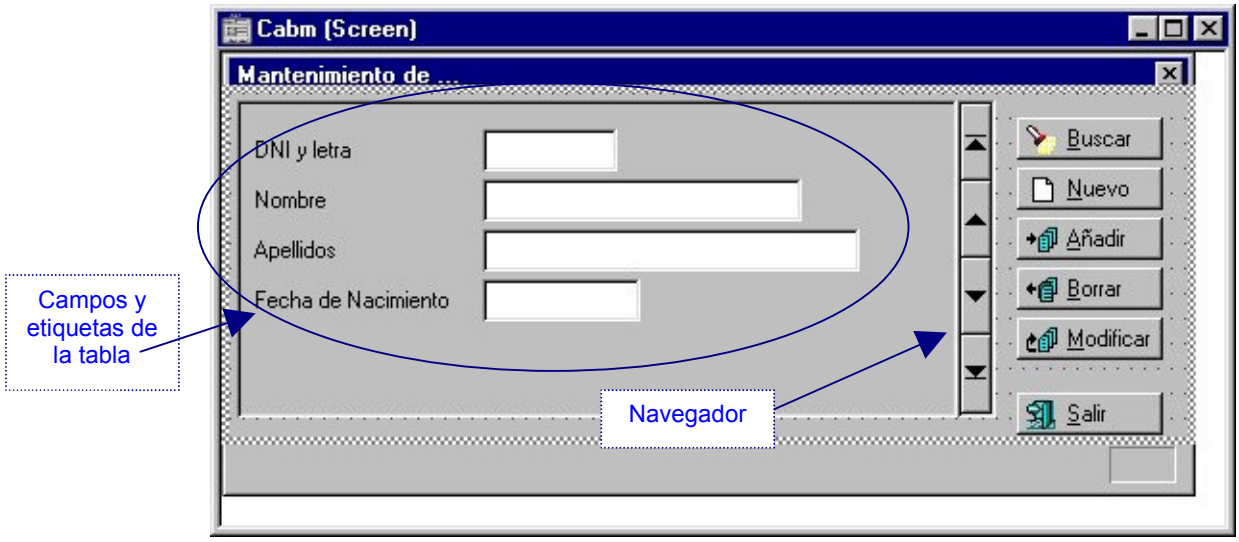


Figura 1.31. Sección Screen de la clase Cabm.

Mediante los botones situados a la derecha de la pantalla se pueden realizar altas, bajas, modificaciones y búsquedas sobre la tabla personal. Si pulsamos sobre el botón ***Nuevo*** aparece el cuadro de propiedades del botón. (Figura 1.32)

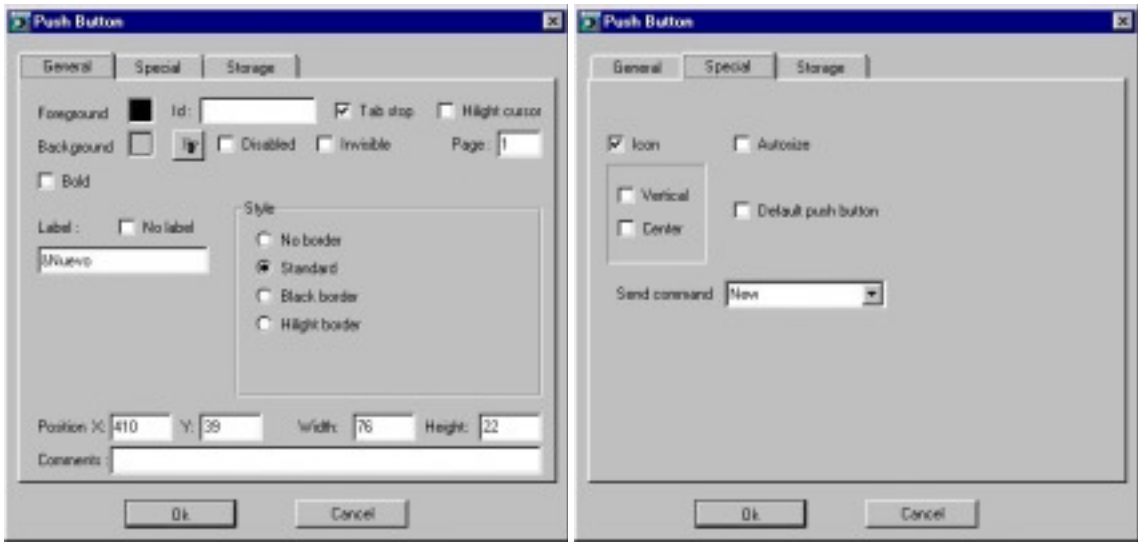


Figura 1.32. Cuadro de diálogo de las propiedades del botón “Añadir”.

En la pestaña ***General*** de este cuadro de diálogo el Wizard ha añadido la etiqueta que se muestra sobre el botón. En la pestaña ***Special*** se marca la opción ***Icon*** para indicar que el botón tiene un icono. Del menú desplegable ***Send Command*** indica el comando que se ejecutará sobre la tabla cuando se pulse este botón. Como se trata de preparar la pantalla para introducir nuevos datos, por tanto se selecciona la opción “New”. Dentro de la pestaña ***Storage*** el Wizard no ha realizado ninguna modificación.

Para el resto de los botones el Wizard ha realizado las mismas operaciones, asociando a cada uno el comando adecuado, según indica la tabla siguiente:

Nombre Botón	Comando	Significado
Buscar	QueryByForm	Muestra un cuadro de diálogo donde se pueden seleccionar los parámetros de la búsqueda
Nuevo	New	Prepara la pantalla para aceptar nuevos datos
Añadir	Add	Añade los datos en pantalla en la tabla
Borrar	Delete	Borrar los datos en pantalla de la tabla
Modificar	Update	Modifica los datos de la tabla con los mostrados en pantalla
Salir	Close	Cierra la pantalla y termina la ejecución del módulo

El Wizard de módulos también ha añadido código en la sección de código del módulo "ABM", haciendo doble click sobre la sección "Code" podemos visualizarlo. (Figura 1.33)

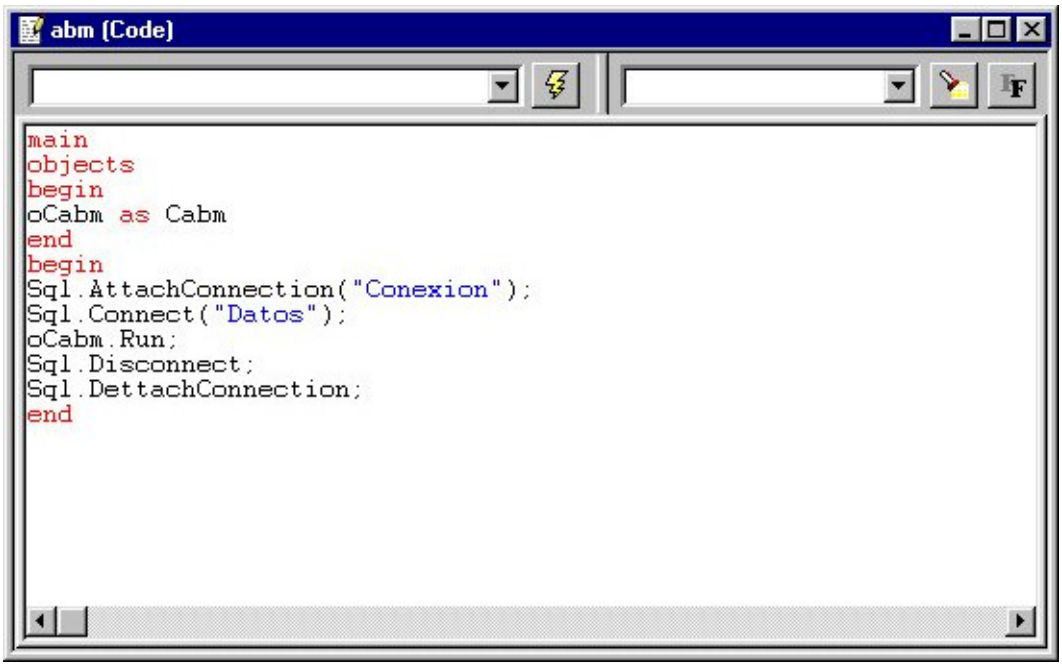


Figura 1.33. Código del módulo ABM.

En la ejecución del Wizard seleccionamos que se realizase la conexión con la base de datos y en la figura 1.33 se muestra código necesario para realizar esta conexión y también lanza la ejecución la pantalla de la clase "Cabm", creando para ello un objeto de esta clase, "oCabm" e invocando el método Run de la Form de la que deriva "Cabm".

Como hemos comprobado, mediante el Wizard de módulo hemos creado sin esfuerzo un módulo que nos permite hacer el mantenimiento completo de la tabla que hemos creado mediante el Editor de Repositorio.

11. Ejecución del módulo de mantenimiento de tabla.

La ejecución del módulo creado en el apartado 9 nos permitirá añadir datos a la tabla "personal" para poder trabajar posteriormente.

Para poder ejecutar este módulo desde el Editor Visual, debe encontrarse seleccionada la sección "Code" del módulo y luego realizar una de las siguientes opciones:

- Pulsar el icono que se encuentra en la barra de herramientas del Editor Visual. (Figura 1.35)
- Pulsar la tecla F5.
- Seleccionar del menú la opción ***Tools*** y dentro de esta la opción ***Execute abm.*** (Figura 1.34)

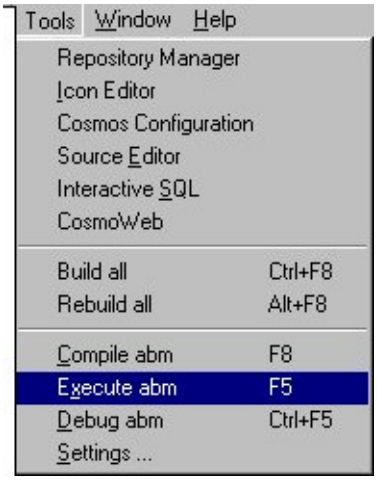


Figura 1.34. Menú Tools.

Por tanto los pasos para ejecutar el módulo creado y su resultado son los que se muestran en la Figura 1.35.

Al encontrarse la tabla "personal" vacía, no se encuentra activo ni el navegador, ni los botones de borrar y modificar.

En cualquier momento podemos abandonar la ejecución de la aplicación pulsando el botón Salir que tiene asociado el método ***close*** de la clase Form.

Ahora vamos a realizar algunas de las operaciones básicas sobre la tabla.

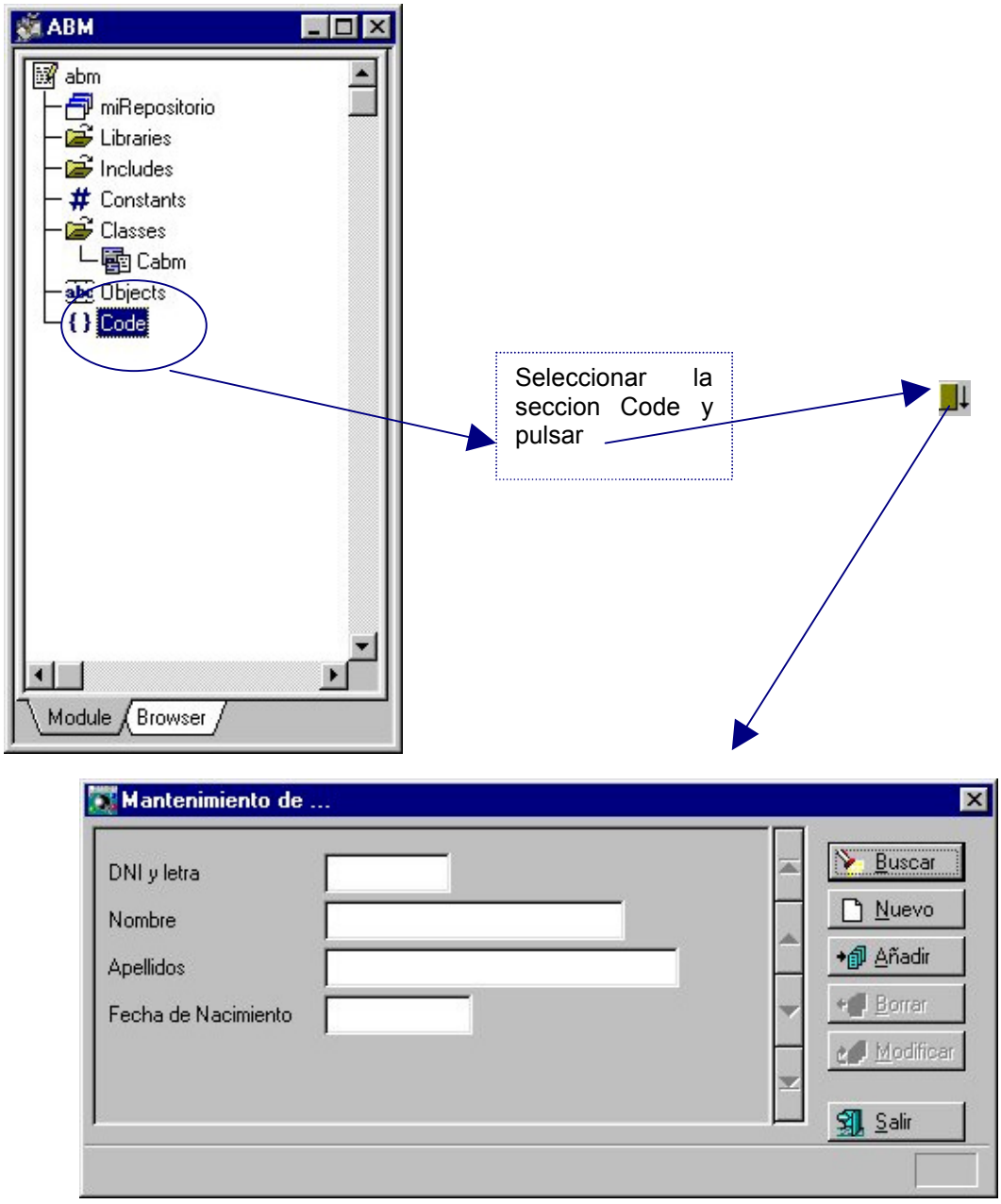


Figura 1.35. Ejecución del módulo “abm”.

Añadir una persona

1. Pulsar el botón 

para preparar los campos para la introducción de datos.
2. Rellenar los campos DNI, nombre, apellidos y Fecha de nacimiento con los nuevos datos. Hay que recordar que la columna “dni” de la tabla “personal” se declaro “not null” y además es clave primaria, por tanto este campo nunca se puede dejar vacio.

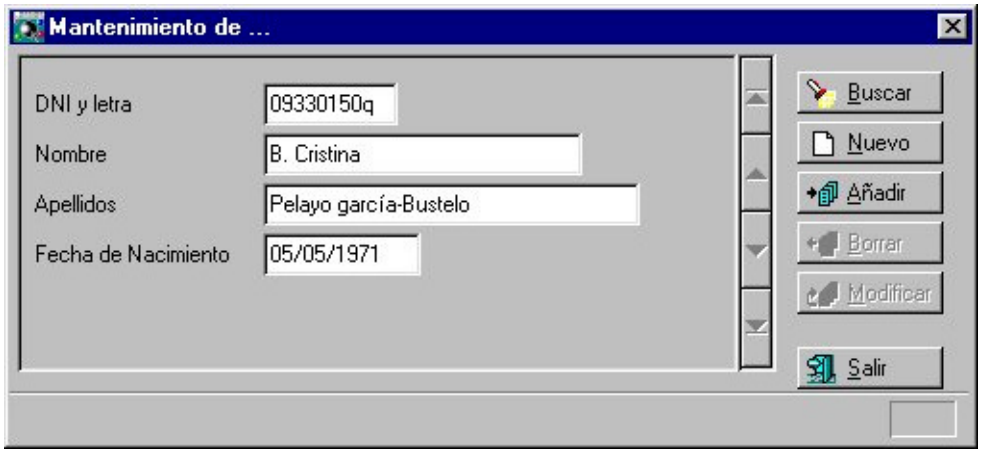


Figura 1.36. Añadir los datos de una persona.

3. Si los datos escritos son correctos, pulsar 

para añadir efectivamente los mismos a la base de datos. Después de realizar este paso se nos muestran los datos de la persona igual que en la figura 1.36. pero sin poder modificar el campo "dni" por tratarse de la clave primaria. También se encuentran activos los botones de Borrar y Modificar, así como la barra de navegación.

Buscar los datos de una persona

1. Pulsando el botón 

se nos muestra el cuadro de diálogo de la figura 1.37. donde podemos seleccionar las características que debe cumplir la persona buscada.

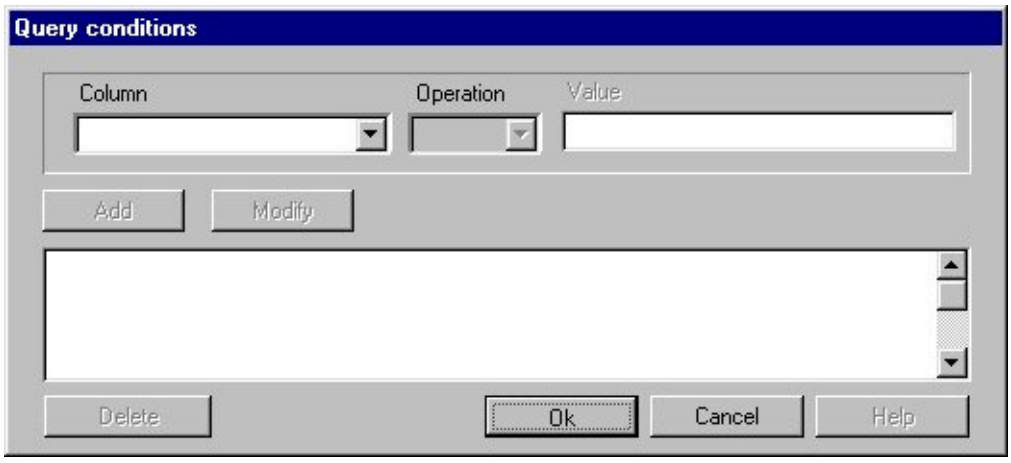


Figura 1.37. Cuadro de diálogo de las condiciones de búsqueda.

2. En el menú desplegable ***Column*** se muestran todas las columnas que tiene la tabla sobre la que estamos trabajando. Vamos a seleccionar "Nombre". (Figura 1.38)

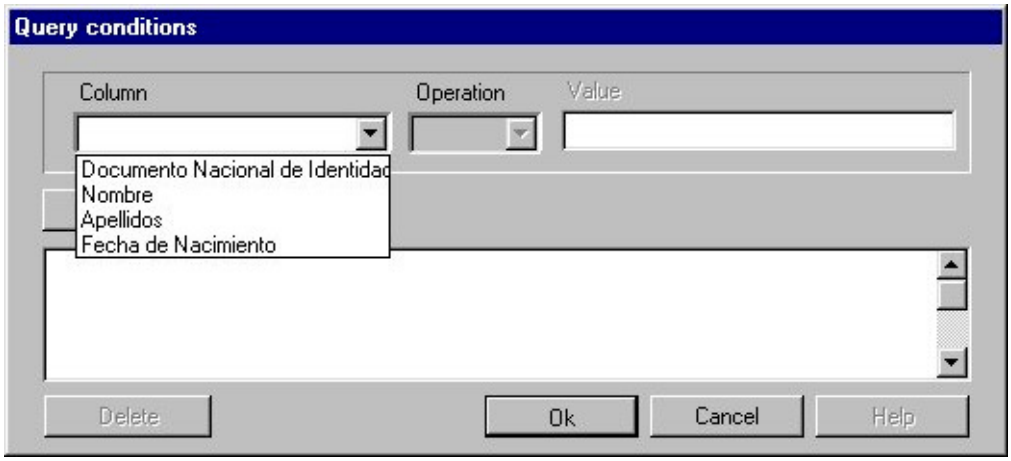


Figura 1.38. Columnas de la tabla “personal”

3. En ***Operation*** se muestran todas las operaciones que se pueden añadir a la condición de búsqueda de la tabla, son las siguientes:

Signo	Operación
=	Igual a
<>	Distinto a
>	Mayor que
<	Menor que
>=	Mayor o igual que
<=	Menor o igual que
in	Se encuentra en la lista de valores
not in	No se encuentra en la lista de valores
Is null	Esta vacío
Is not null	No esta vacío
Matches	Condición de comparación de expresiones de tipo alfanumérico. Se pueden utilizar los metacaracteres *,?,[caracteres],[^caracteres],\.
like	Condición de comparación de expresiones de tipo alfanumérico. Se pueden utilizar los metacaracteres % y _ (subrayado bajo)

En ***Operation*** vamos a seleccionar el signo de igualdad como se muestra en la figura 1.39.

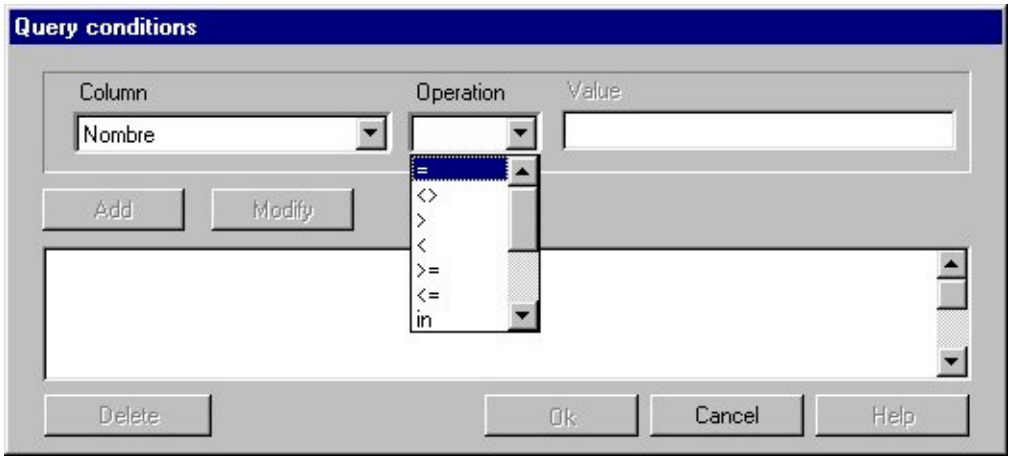


Figura 1.39. Selección de la operación.

4. Ahora tenemos que introducir el valor que queremos que cumpla la columna nombre dentro de la sección ***Value***. Dependiendo de la operación seleccionada anteriormente podemos introducir una expresión simple o una lista de expresiones para los casos de in y not in.

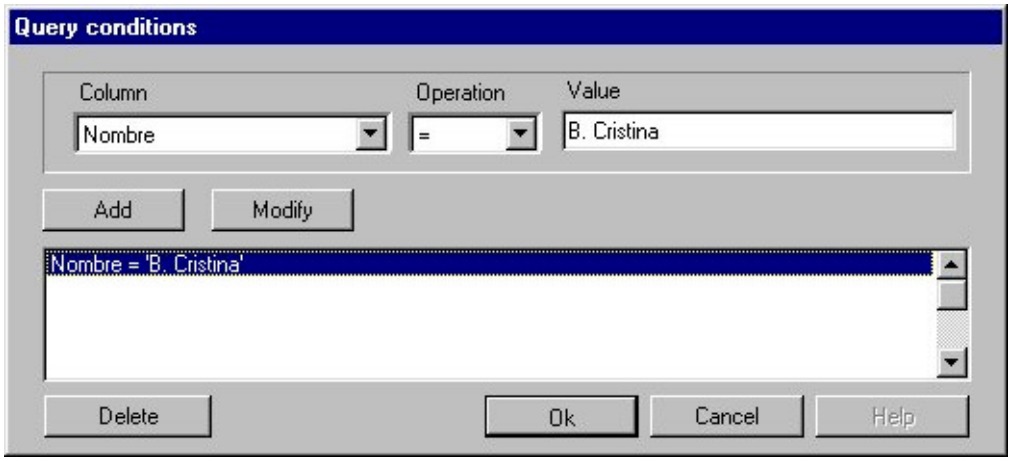
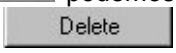


Figura 1.40. Aspecto final de la búsqueda.

Después de tener completos los tres campos de selección anteriores, se pulsa el botón 

para añadir esta búsqueda al campo de edición donde se mostrarán todas las búsquedas definidas. Mediante el botón 

podemos modificar las especificaciones de una búsqueda seleccionada. Con el botón 

podemos borrar la búsqueda seleccionada.

5. Pulsamos el botón “Ok” para indicar que deseamos realizar la búsqueda efectivamente. Tenemos que tener seleccionada previamente la búsqueda.

6. En la figura 1.41. se muestra el resultado de la búsqueda que hemos efectuado. En este caso las condiciones de búsqueda eran muy concretas y sólo hemos obtenido como resultado una única tupla, en el caso de un resultado con varias tuplas, mediante los botones del navegador nos podríamos mover por ellas.

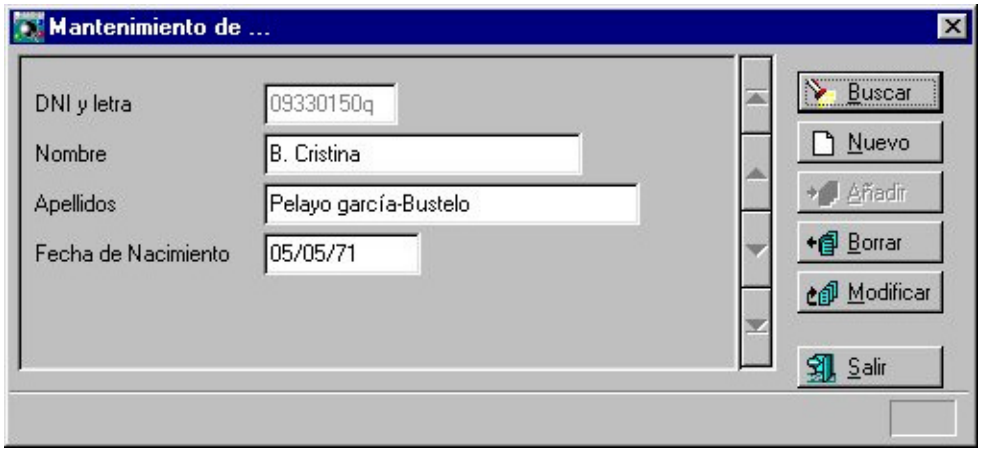


Figura 1.41. Resultado de la búsqueda.

Modificar los datos de una persona

1. Lo primero que tenemos que tener son los datos de la persona que queremos modificar visibles en los campos de edición correspondientes. La clave primaria de la tabla “personal”, que es dni, no se puede modificar luego aparece deshabilitado su campo de edición. Para esto podemos realizar una búsqueda como se ha explicado anteriormente.

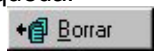
2. Comprobamos que en la columna “Apellidos” hemos cometido un error al introducir “Garcia” en minúsculas, lo vamos a modificar. Para ello nos situamos en el campo de edición correspondiente y realizamos la modificación. (Figura 1.42)

Figura 1.42. Modificación del apellido.

3. Pulsamos el botón 

para que se realice efectivamente la modificación.

Borrar una persona

1. Para poder borrar una persona de la tabla “personal” debemos tener visualizados sus datos mediante una búsqueda.
2. Pulsamos el botón 

para realizar el borrado.

12. Código generado para el módulo.

En este apartado vamos a analizar el código que el Wizard de módulos ha generado para el módulo “abm”.

Este código se encuentra almacenado en la ruta especificada en su definición (apartado 10 de este capítulo) en el fichero “abm.smd”.

La extensión *.smd es utilizada para almacenar el código fuente de un programa, librería o include. Son ficheros de tipo ASCII.

Fichero Abm.smd

```
REPOSITORY mirepositorio
CONSTANTS  begin
end
CLASSES BEGIN
    Cabm is Form begin
        objects begin
        end
        Table personal is personal
        begin
            dni as column required
            nombre as column
            apellidos as column
            fecha_nacimiento as column
        end
        INTERFACE
            POSITION 0 0 507 231
            LABEL "Mantenimiento de ..."
            SYSMENU
        BEGIN
            CONTROL AS BOX
                POSITION 0 0 499 179
                FOREGROUND RGB 0 0 0
                ATTACH ALL
                NOLABEL
                NOBORDER
            BEGIN
                CONTROL AS BUTTON
                    POSITION 410 13 76 22
                    ATTACH RIGHT
                    LABEL "&Buscar"
                    ICON 6 FILE "Small Buttons"
                    COMMAND QueryByForm
                CONTROL AS BUTTON
                    POSITION 410 65 76 22
                    ATTACH RIGHT
                    LABEL "&Añadir"
                    ICON 157 FILE "Small Buttons"
                    COMMAND Add
                CONTROL AS BUTTON
                    POSITION 410 91 76 22
                    ATTACH RIGHT
                    LABEL "&Borrar"
                    ICON 159 FILE "Small Buttons"
                    COMMAND Delete
                CONTROL AS BUTTON
                    POSITION 410 117 76 22
                    ATTACH RIGHT
                    LABEL "&Modificar"
                    ICON 158 FILE "Small Buttons"
                    COMMAND Update
                CONTROL AS BUTTON
                    POSITION 410 39 76 22
                    ATTACH RIGHT
                    LABEL "&Nuevo"
                    ICON 84 FILE "Small Buttons"
                    COMMAND New
                CONTROL AS BUTTON
                    POSITION 410 156 76 22
                    ATTACH RIGHT
```

```
    LABEL "&Salir"
    ICON 4 FILE "Small Buttons"
    COMMAND Close
CONTROL IDMASTER AS BOX
    POSITION 4 4 375 165
    ATTACH ALL
    NOLABEL
    BORDER DOUBLE DOWN
    TABLE personal
BEGIN
    CONTROL AS TEXT
        POSITION 7 13 119 22
        NOBORDER
        DATATYPE CHAR
        TAGS "DNI y letra"
    CONTROL AS EDIT
        POSITION 126 13 70 22
        LABEL "Documento Nacional de Identidad"
        BORDER DOUBLE DOWN
        VARIABLE personal.dni
        LIKEVAR
    CONTROL AS TEXT
        POSITION 7 39 37 22
        NOBORDER
        DATATYPE CHAR
        TAGS "Nombre"
    CONTROL AS EDIT
        POSITION 126 39 166 22
        LABEL "Nombre"
        BORDER DOUBLE DOWN
        VARIABLE personal.nombre
        LIKEVAR
    CONTROL AS TEXT
        POSITION 7 65 42 22
        NOBORDER
        DATATYPE CHAR
        TAGS "Apellidos"
    CONTROL AS EDIT
        POSITION 126 65 196 22
        LABEL "Apellidos"
        BORDER DOUBLE DOWN
        VARIABLE personal.apellidos
        LIKEVAR
    CONTROL AS TEXT
        POSITION 7 91 101 22
        NOBORDER
        DATATYPE CHAR
        TAGS "Fecha de Nacimiento"
    CONTROL AS EDIT
        POSITION 126 91 82 22
        LABEL "Fecha de Nacimiento"
        BORDER DOUBLE DOWN
        VARIABLE personal.fecha_nacimiento
        LIKEVAR
END
CONTROL AS SPIN
    POSITION 379 4 20 165
    ATTACH TOP RIGHT BOTTOM
    BORDER DOUBLE DOWN
    VERTICAL
    QUADRUPLE
```

```
                    COMMAND
                END
                CONTROL AS BAR
                    POSITION 0 179 499 27
                    FOREGROUND RGB 0 0 0
                    ATTACH LEFT RIGHT BOTTOM
                    BORDER ETCHED EXTRA 3
                    BARTYPE STATUS
                BEGIN
                    CONTROL AS PANEL
                        POSITION 3 0 449 19
                        FOREGROUND RGB 0 0 0
                        ATTACH ALL
                        NOBORDER
                        PANELTYPE COMMENT
                    CONTROL AS PANEL
                        POSITION 455 0 35 19
                        FOREGROUND RGB 0 0 0
                        ATTACH TOP RIGHT BOTTOM
                        BORDER DOWN
                        PANELTYPE EDITMODE
                END
            END
        end
END
CODE BEGIN
//{{CODEBEGIN
main
objects
begin
oCabm as Cabm
end
begin
Sql.AttachConnection("Conexion");
Sql.Connect("Datos");
oCabm.Run;
Sql.Disconnect;
Sql.DettachConnection;
end
//{{CODEEND
END
```

Análisis del código

En la primera parte del código se incluye el repositorio que forma parte del módulo, en la definición del módulo seleccionamos un repositorio para que formase parte del mismo y ha sido reflejado en el código mediante la instrucción:

```
REPOSITORY mirepositorio
```

Como no hemos definido ninguna constante dentro del módulo la sección de ***Constants*** del código se encuentra vacia:

```
CONSTANTS  begin
end
```

La siguiente sección que nos encontramos en el código es ***Classes***. Mediante el Wizard definimos una clase derivada de la clase Form:

```
Cabm is Form begin
```

El `begin` indica que comienza la definición de la clase. Dentro de la definición de la clase existen también una serie de secciones:

Asociación de la tabla personal a la clase y definición de cada una de sus columnas:

```
Table personal is personal
  begin
      dni as column required
      nombre as column
      apellidos as column
      fecha_nacimiento as column
  end
```

Definición de la sección ***Screen*** que compone la interfaz de la aplicación. Se especifican cada uno de los componentes. Para los botones se indica la posición que ocupan en la pantalla, justificación, etiqueta, icono, y comando que ejecutan en el caso de los botones.

```
CONTROL AS BUTTON
    POSITION 410 13 76 22
    ATTACH RIGHT
    LABEL "&Buscar"
    ICON 6 FILE "Small Buttons"
    COMMAND QueryByForm
```

Luego tenemos un control de tipo ***Box*** que tiene asociada la tabla “personal” y que contiene las etiquetas y los campos de edición para las columnas de la tabla:

```
CONTROL IDMASTER AS BOX
    POSITION 4 4 375 165
    ATTACH ALL
    NOLABEL
    BORDER DOUBLE DOWN
    TABLE personal
BEGIN
    CONTROL AS TEXT
        POSITION 7 13 119 22
        NOBORDER
        DATATYPE CHAR
        TAGS "DNI y letra"
    CONTROL AS EDIT
        POSITION 126 13 70 22
        LABEL "Documento Nacional de Identidad"
        BORDER DOUBLE DOWN
        VARIABLE personal.dni
        LIKEVAR
  (.....)
  END
```

Etiqueta →

Campo de Edición →

También definimos un navegador con cuatro botones que nos permiten movernos por los datos que existen en la tabla:

```
CONTROL AS SPIN
    POSITION 379 4 20 165
    ATTACH TOP RIGHT BOTTOM
    BORDER DOUBLE DOWN
    VERTICAL
    QUADRUPLE
    COMMAND
```

Después tenemos la sección de código del módulo, que es la que vimos reflejada en la figura 1.33 de este capítulo:

```
CODE BEGIN
//{{CODEBEGIN
main
objects
begin
oCabm as Cabm
end
begin
Sql.AttachConnection("Conexion");
Sql.Connect("Datos");
oCabm.Run;
Sql.Disconnect;
Sql.DettachConnection;
end
//{{CODEEND
END
```

13. Archivos generados en este capítulo.

Se han generado dos carpetas en la ruta especificada y un archivo llamado “historico”, que fue el especificado en la figura 1.20 en el campo de edición ***log in file***.

El contenido de la carpeta “miproyecto” es el siguiente:

Archivo	Contenido
Miproyecto.prj	Fichero binario donde se guarda la estructura y configuración del proyecto.
Miproyecto.pws	Guarda el estado de ejecución del proyecto abierto.
Mirepositorio.crf	Ficheros binarios donde se guarda la estructura del repositorio, ha sido generado por el Editor de Repositorios.
Mirepositorio.dfr	Fichero donde se guarda la documentación que se ha ido generando en la construcción del repositorio.
Abm.omd	Ficheros objeto generados por el compilador de Cosmos. Durante la ejecución de una aplicación estos ficheros tienen que encontrarse en el mismo directorio que los módulos fuente.
Abm.smd	Fichero ASCII que contiene el código fuente del módulo “ABM”.

El contenido de la carpeta “datos.dbs” representa todos los ficheros necesarios para el uso y definición de la base de datos. Hay 26 ficheros, de dos tipos diferentes, diferenciados por la extensión, que son:

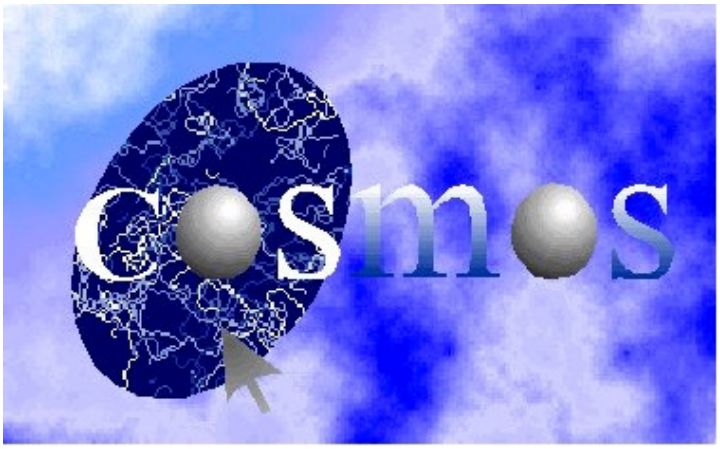
Extensión	Significado
*.dat	Fichero binario que contiene los datos de una base de datos. Debe estar en el mismo directorio que su respectivo “*.idx”
*.idx	Fichero binario que contenedrá la estructura de todos los índices relativos a una tabla de la base de datos.

En los siguientes capítulos vamos a profundizar en el uso de Wizards y de base de datos.

CAPÍTULO 2

Primer informe

1. Introducción.
2. Crear un informe mediante asistentes (Wizards).
3. Ejecución del módulo.
4. Clase Tinf_personal.
5. Código generado para el módulo.
6. Archivos generados en este capítulo.



1. Introducción.

En este capítulo vamos a realizar un informe con la tabla realizada en el capítulo 1 mediante el Wizard de módulos. Veremos el código que genera y como se pueden modificar las características del resultado.

Utilizaremos el mismo proyecto que desarrollamos a lo largo del capítulo 1.

2. Crear un informe mediante asistentes (Wizard).

Partimos del proyecto creado en el capítulo 1 llamado “Miproyecto” que tenía el módulo de mantenimiento de la tabla “personal”. (Figura 2.1)

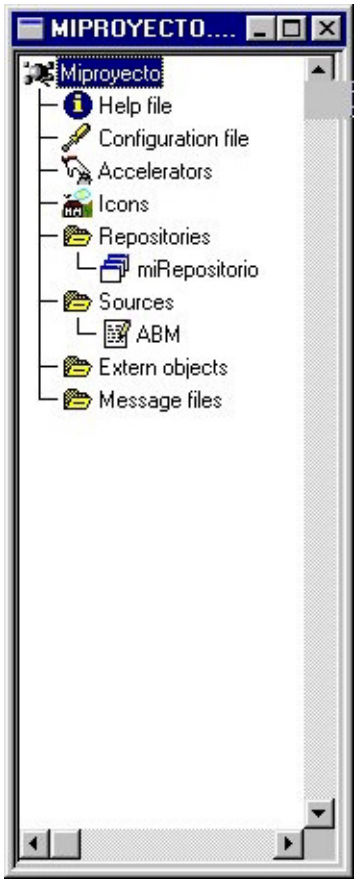


Figura 2.1. Aspecto del proyecto de partida.

Sobre este proyecto vamos a añadir un nuevo módulo. Para ello tenemos que situarnos sobre la sección ***Sources*** y pulsando el botón secundario del ratón seleccionamos la opción ***Add***. (Figura 2.2)

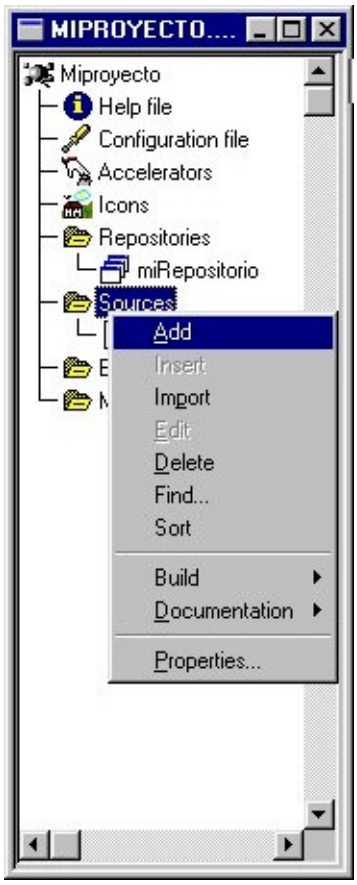


Figura 2.2. Seleccionar la opción Add.

En el cuadro de diálogo de las propiedades del módulo introducimos una etiqueta para el nuevo módulo, el comentario asociado, el nombre del fichero con extensión .smd que lo almacenará así como la ruta. Todas estas características se muestran en la figura 2.3. a continuación pulsamos el botón  para utilizar el Wizard de módulos.

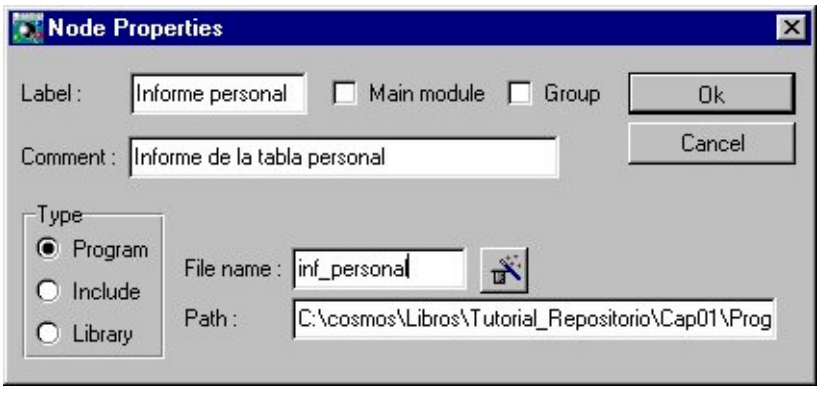


Figura 2.3. Propiedades del nuevo módulo

Dentro del Wizard de módulos debemos seleccionar el tipo de módulo que se generará, en este caso seleccionamos ***Simple Report*** porque queremos realizar un informe sobre la tabla "personal" que tenemos en nuestro repositorio. (Figura 2.4)

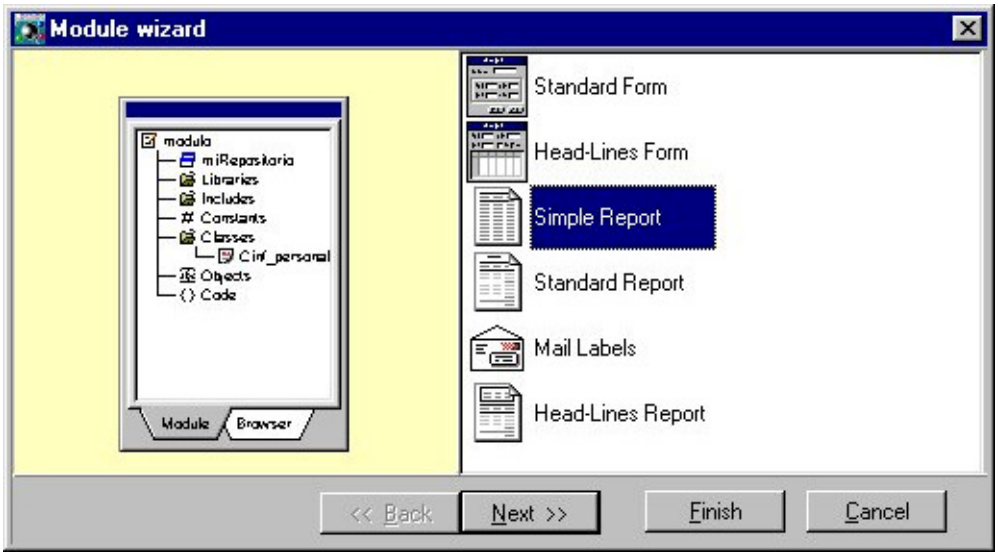


Figura 2.4. Selección del tipo de módulo.

Pulsamos el botón Next >> para continuar con la generación del módulo. En la figura 2.5. se muestra el siguiente cuadro de diálogo donde hay que seleccionar el repositorio y el nombre de la clase que se crea. El nombre que aparece por defecto es “Cinf_personal” debido a que el nombre con el que se va a almacenar el código es “inf_personal” que fue el que se especificó en el campo de edición correspondiente de la figura 2.3.

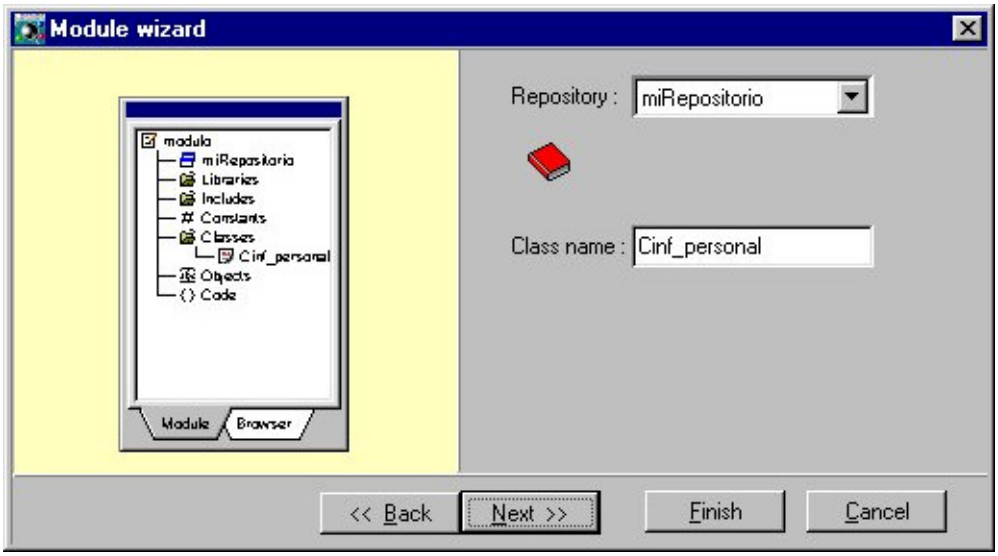


Figura 2.5. Seleccionar un repositorio y el nombre de la clase.

Como siempre el botón << Back permite regresar a la pantalla anterior y modificar las acciones ahí realizadas. Con Finish se termina la generación del módulo y Cancel cancela dicha acción.

En el siguiente cuadro de diálogo (Figura 2.6) seleccionamos el tipo de informe concreto que queremos realizar, se nos muestran tres alternativas: listado de líneas, informe en forma de ficha y generación de etiquetas. Se selecciona la primera opción.

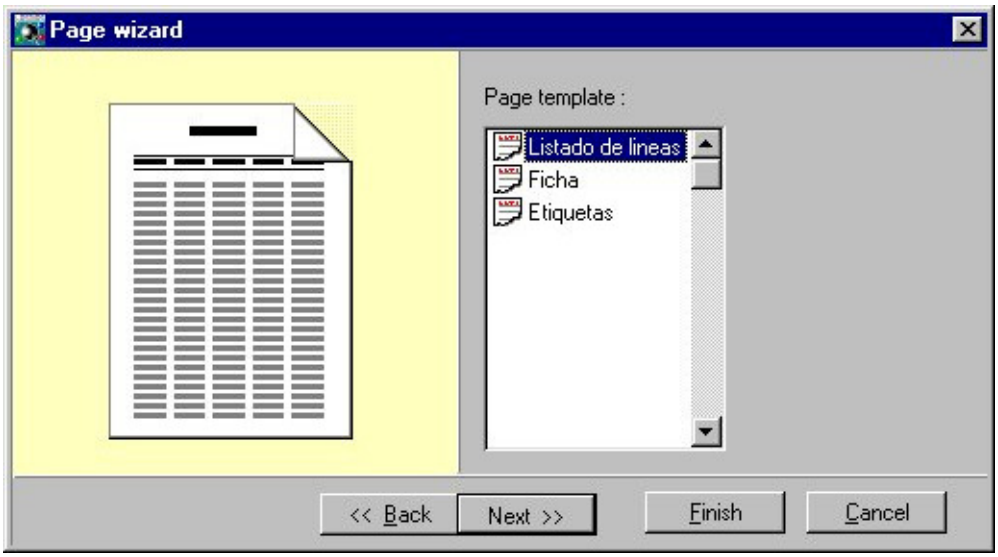


Figura 2.6. Selección del tipo de informe.

Ahora tenemos que seleccionar la tabla que pertenece al repositorio de la que vamos a realizar el informe. Solamente tenemos una tabla, "personal" que es la que seleccionamos. Se nos muestran entonces las columnas que componen dicha tabla para que realicemos la selección de aquellas que queremos mostrar en el informe. Como vamos a realizar un informe completo, es decir, con todos los datos de una persona, seleccionamos mediante  todas las columnas.

Si queremos seleccionar una única columna seleccionamos el nombre de la columna del cuadro de la izquierda (***Table columns***)y pulsamos  .

Si queremos suprimir una selección de columna previamente realizada, debemos seleccionar el nombre del cuadro de la derecha marcado como ***Page columns*** y pulsar el botón  . Si deseamos deshacer todas las columnas seleccionadas hasta el momento pulsamos  .

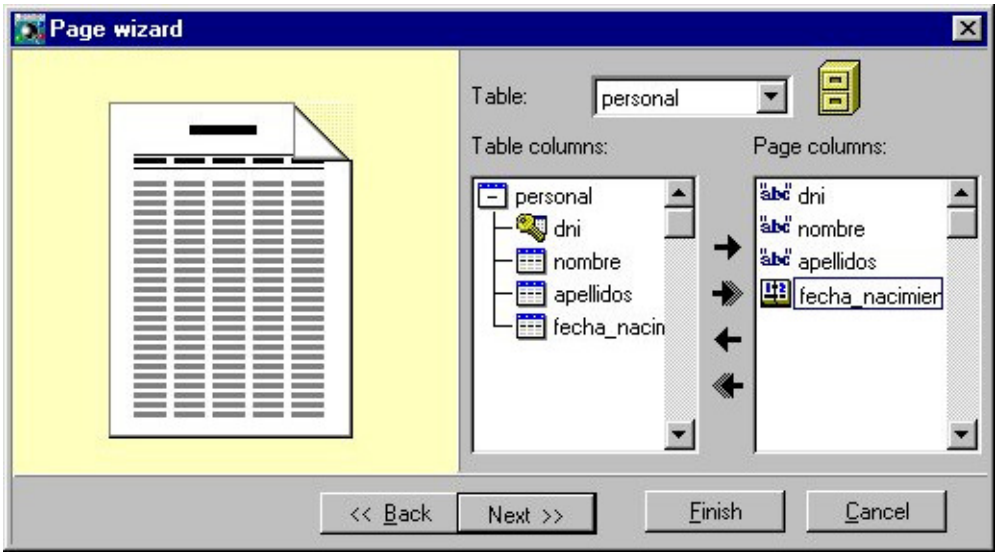
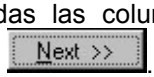


Figura 2.7. Selección de las columnas.

Cuando ya tenemos seleccionadas las columnas de la tabla que constituirán las columnas de la página, pulsamos el botón  .

El siguiente paso es introducir el nombre del título que se mostrará en el informe. Por defecto se nos muestra la cadena “Listado de” mas el nombre de la tabla que se ha seleccionado en la figura 2.7. Si deseamos modificar este nombre únicamente tenemos que situarnos sobre el campo de edición y escribir el título deseado. (Figura 2.8)

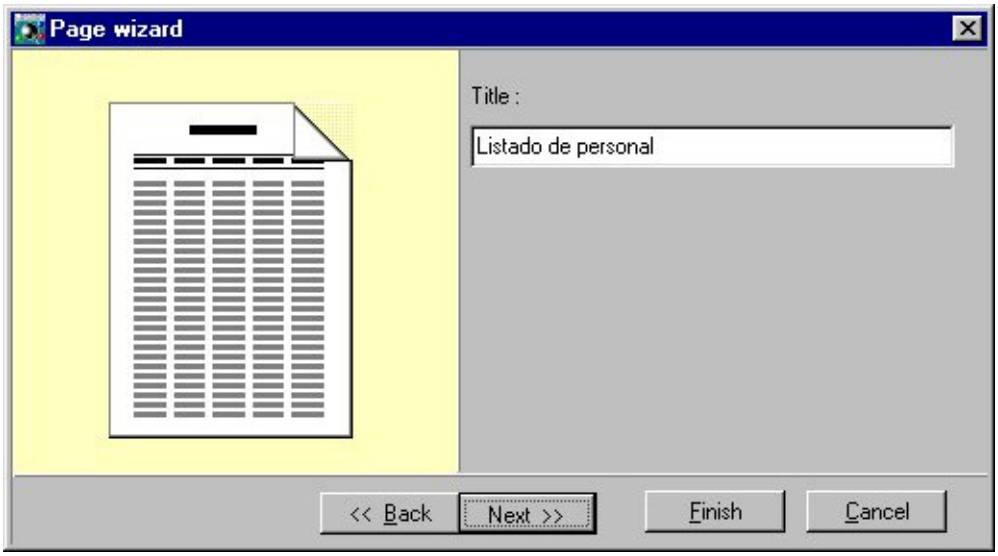


Figura 2.8. Título del informe.

En la figura 2.9. se nos muestra el último cuadro de diálogo de este asistente, donde se da la posibilidad de establecer la conexión SQL para el módulo. Si marcamos ese campo de selección se nos muestran en un menú desplegable tanto las conexiones definidas como las bases de datos de cada conexión. Seleccionamos las que hemos utilizado en este proyecto y pulsamos el botón 

para que se genere efectivamente el módulo del informe.

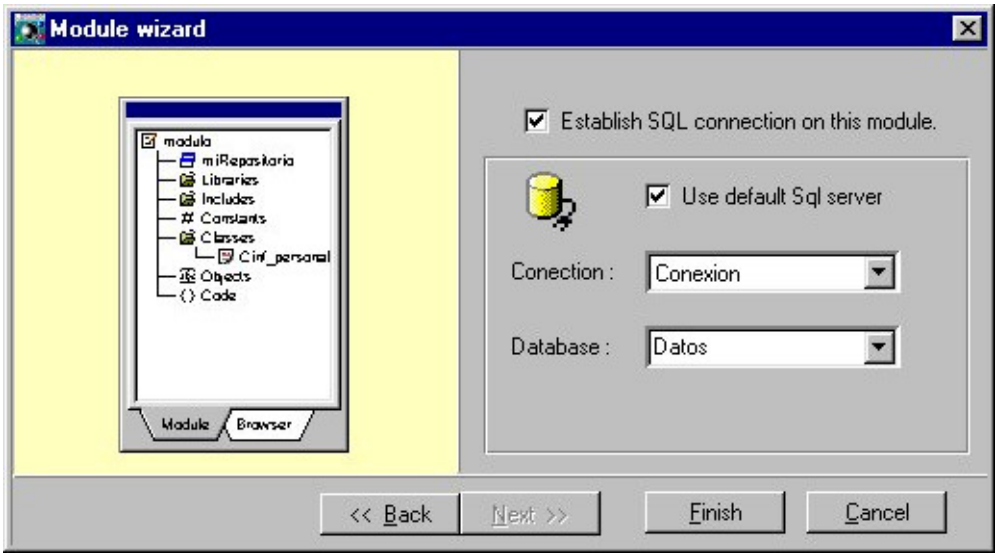


Figura 2.9. Establecer conexión con la base de datos.

La paleta del proyecto muestra en este momento el nuevo módulo añadido (Figura 2.10).

También se ha creado un grupo dentro de la sección ***Sources*** llamado ***Include*** que se compone por un nuevo módulo llamado “templinc2”, que no es un programa sino un ***Include***.

Un include es un módulo que exporta sus métodos públicos, clases públicas, objetos globales públicos y constantes. Así mismo un include exporta los métodos, clases, etc, públicos que incluye a su vez.

En “templinc2” es un módulo donde se encuentran definidas una serie de clases, objetos y métodos para realizar informes con Cosmos. En la aplicación actual únicamente haremos uso de su funcionalidad sin entrar en detalles de cómo se implementa esta. En la sección de anexos de este manual se analiza el código de este include.

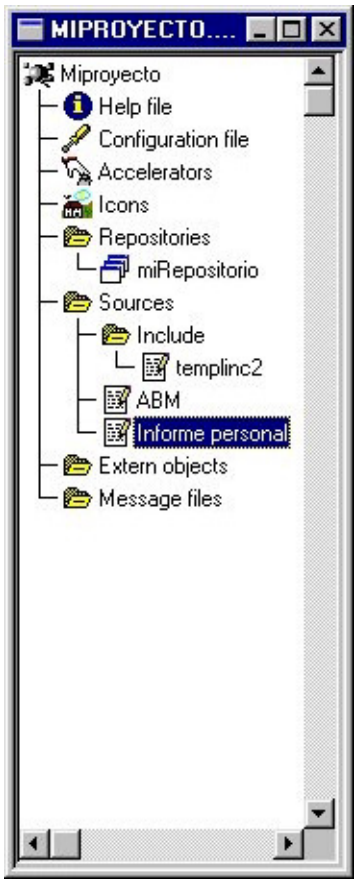


Figura 2.10. Paleta del proyecto con el nuevo módulo.

3. Ejecución del módulo.

Vamos a ver el aspecto que tiene este nuevo módulo, “Informe personal” cuando lo ejecutamos. Para ello lo primero que vamos a hacer es abrir la paleta del nuevo módulo. En el encabezamiento de la paleta aparece la etiqueta que se le asigno a este módulo y no el nombre con el que se guarda efectivamente que era “Inf_personal”.(Figura 2.11)

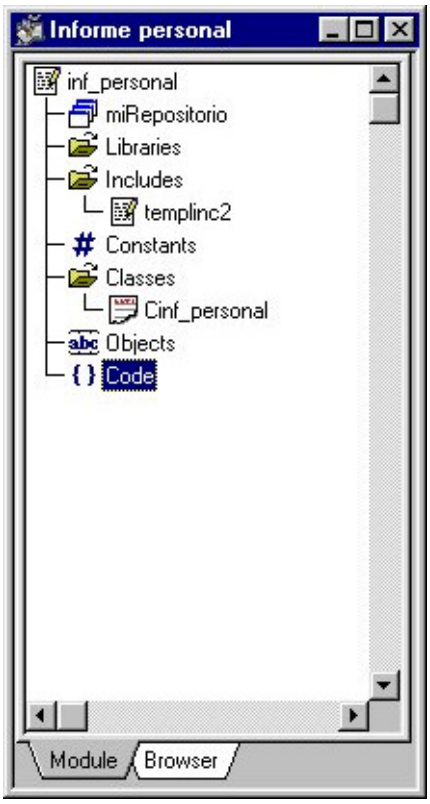


Figura 2.11. Paleta del módulo “Inf_personal”.

Se selecciona la sección ***Code*** de este módulo y se ejecuta pulsando F5 ó seleccionando la opción ***Execute inf_personal*** del menú ***Tools*** o pulsando el icono  .

Al tratarse de un informe lo primero que se nos muestra es la pantalla de configuración de impresora como se ve en la figura 2.12.

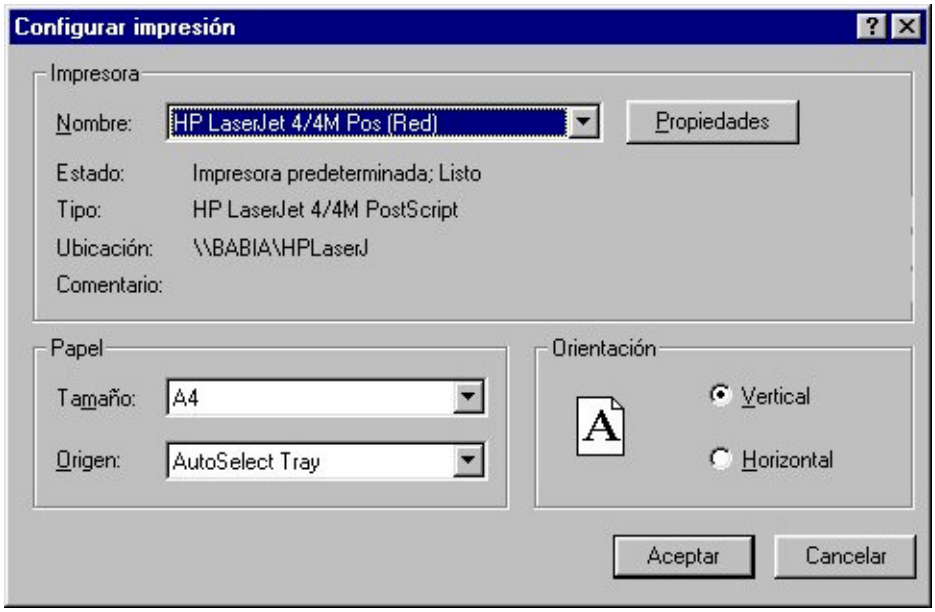


Figura 2.12. Configuración de impresora.

Después de configurar la impresora pulsamos el botón 

y se muestra la visión preliminar del informe (Figura 2.13). En esta pantalla es donde podemos imprimir el informe o cancelar la impresión.

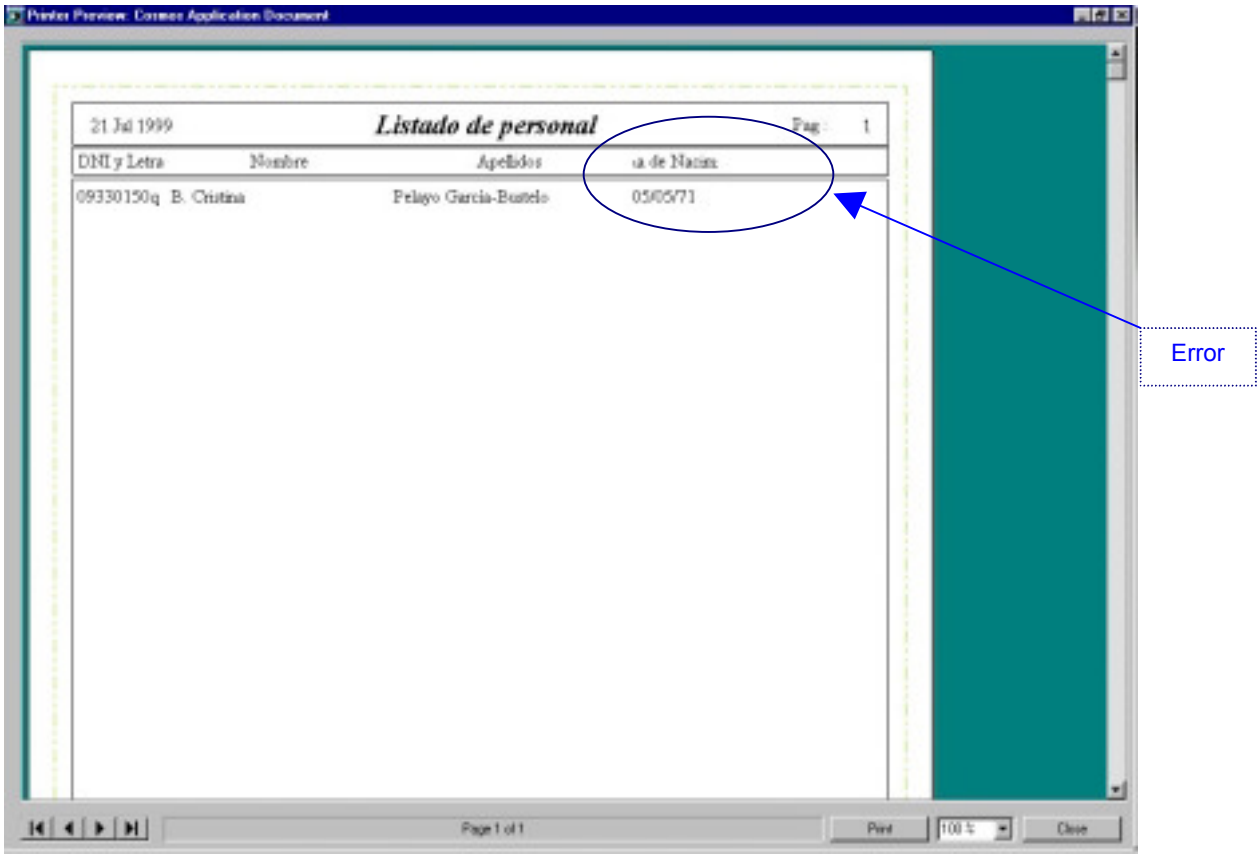
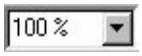


Figura 2.13. Preview del informe.

En la figura 2.13. hay varios componentes que es necesario explicar:

 Navegador por las diferentes páginas que puede tener un informe. En nuestro caso tenemos una persona y por tanto una única página, como se indica en la barra de estado.

Si se pulsa este botón se imprime el informe en la impresora especificada.



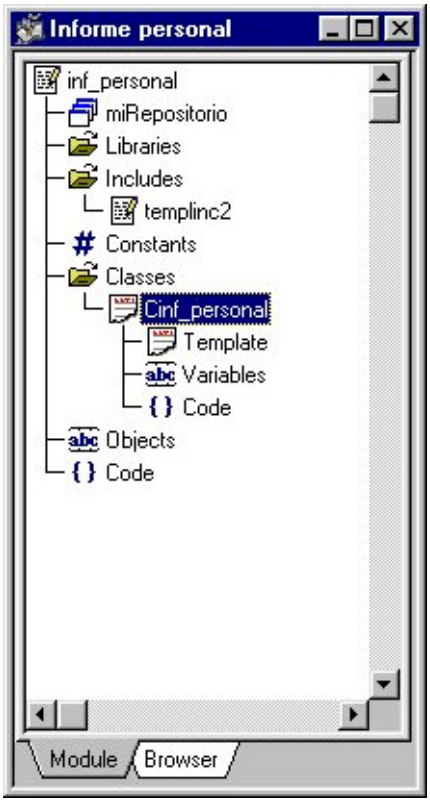
Se puede seleccionar el zoom con el que queremos visualizar el informe en este preview.

Cerrar la vista preliminar sin realizar la impresión del informe.

Como podemos ver existe un error marcado mediante un círculo, no se muestra adecuadamente la etiqueta de la columna “Fecha de nacimiento”. En el siguiente apartado solucionaremos este error y también modificaremos el aspecto del informe.

4. Clase Cinf_personal.

Seleccionamos en la paleta del módulo “Informe personal” (Figura 2.11) la clase Cinf_personal y se hace doble click sobre la misma. Se muestra el contenido de esta clase en la figura 2.14.



2.14. Contenido de la clase Cinf_personal.

Vamos a ir analizando el contenido de cada uno de los elementos que componen esta clase.

Sección Template

Si hacemos doble click de ratón sobre la sección ***Template*** se muestra la definición del informe como vemos en la figura 2.15.

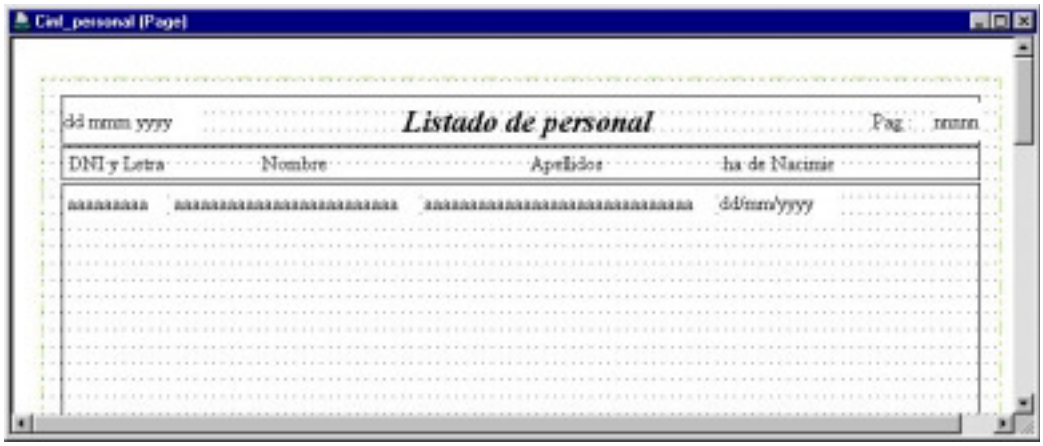


Figura 2.15. Definición de las características del informe.

En este momento es cuando vamos a modificar el aspecto del informe para corregir el error que detectamos en la figura 2.13. Seleccionamos la etiqueta de la variable que mostrará la fecha de nacimiento y estiramos su longitud hasta que se muestre la cabecera adecuadamente. (Figura 2.16)

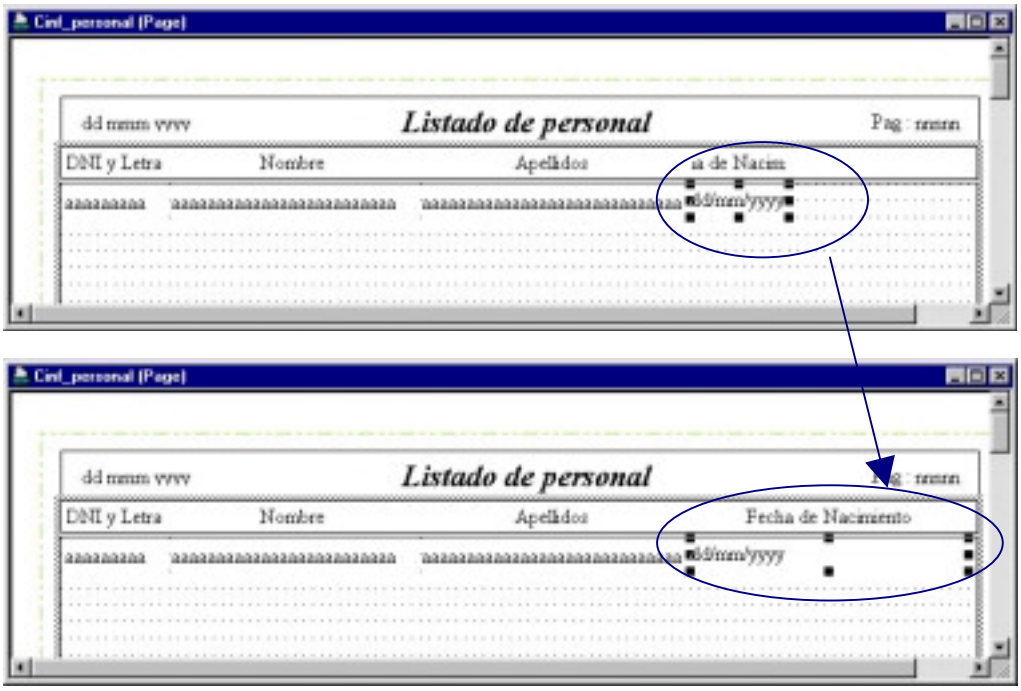


Figura 2.16. Modificar el tamaño de los campos.

Ahora vamos a añadir unas líneas que nos separen las diferentes columnas del informe. Para ello tenemos que seleccionar el control de tipo ***Group*** que es el que se marca en la figura 2.17.

Figura 2.17. Control de tipo group.

Haciendo doble click de ratón sobre él o pulsando el botón secundario del ratón y seleccionando la opción ***Properties***, se muestra el cuadro de diálogo con sus propiedades que tiene tres pestañas. En la figura 2.18. se muestran dos de estas tres pestañas, en la pestaña marcada como ***Storage*** no ha sido modificado ninguna parte de su contenido por el Wizard de módulos.

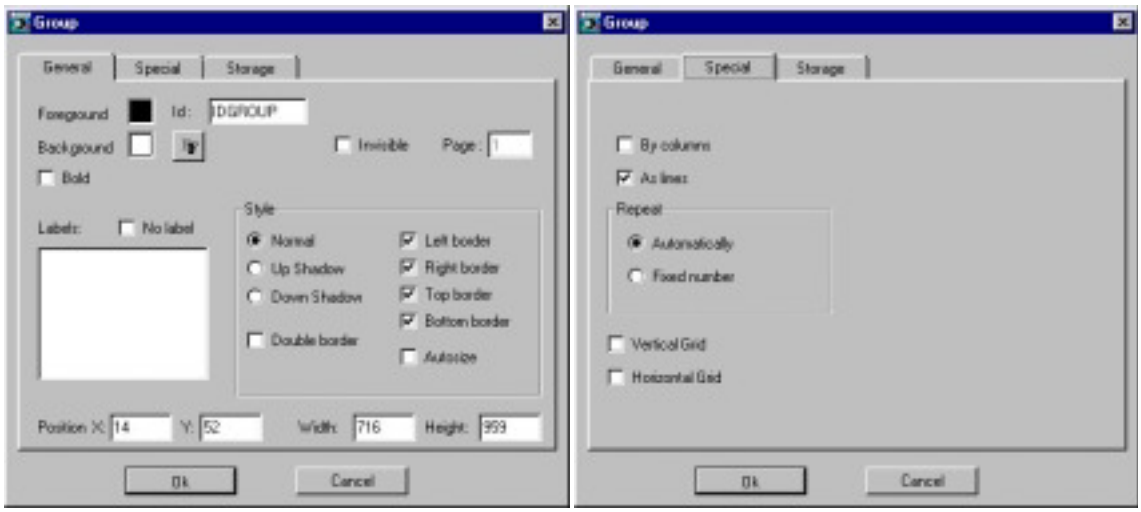


Figura 2.18. Propiedades del control de tipo Group.

Vamos a marcar en la pestaña ***Special*** la casilla de verificación ***Vertical Grid*** para que se muestren las líneas que separan a las columnas. El informe quedará como se muestra en la figura 2.19.

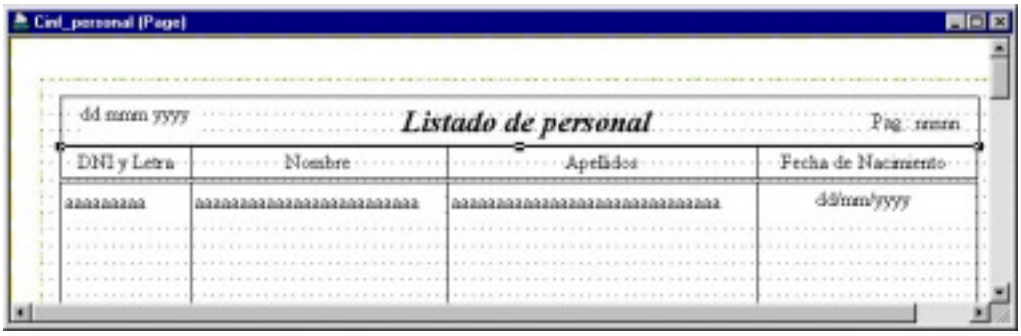


Figura 2.19. Aspecto del informe con las líneas verticales.

Sección Variables.

Hacemos doble click de ratón sobre esta sección en la paleta del módulo (Figura 2.14) se nos muestran las variables declaradas en la clase “Cinf_personal”. (Figura 2.20)

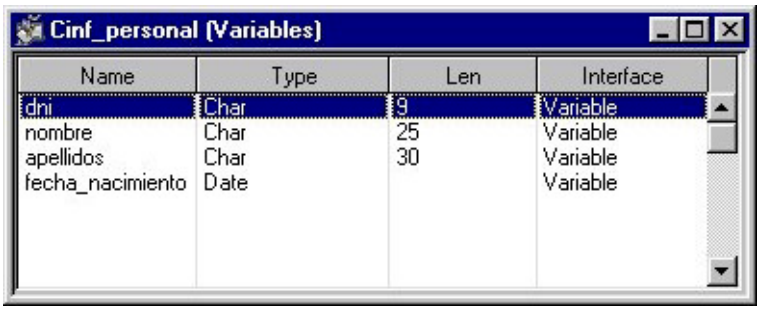


Figura 2.20. Contenido de la sección Variables.

Para ver las características de estas variables podemos hacer doble click de ratón sobre cada una de ellas. Por ejemplo para “dni” las características que se muestran son las de la figura 2.21.

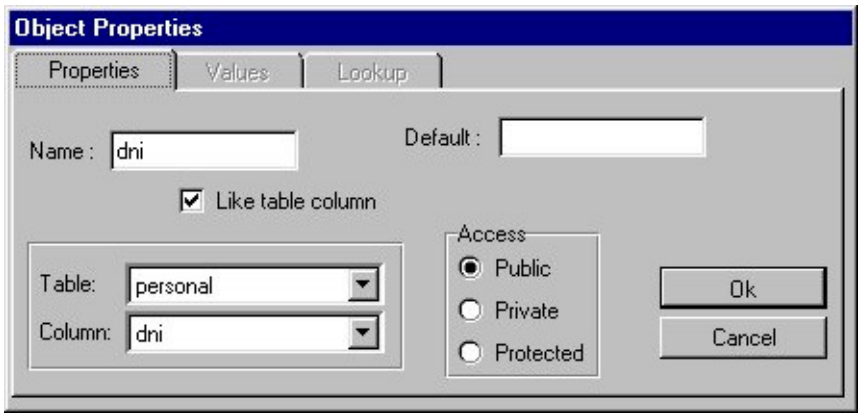


Figura 2.21. Propiedades de la variable “dni”.

En las propiedades de la variable “dni” el Wizard ha asociado la variable a una columna de la tabla personal, y se permite el acceso *public* a la misma. El resto de las variables declaradas en esta sección tienen la misma definición a cada una de ellas se le asigna una columna de la tabla “personal”.

Sección Code

En esta sección se encuentra escrito el código de la clase “Cinf_personal”. Si hacemos doble click sobre esta sección se muestra el editor de código conteniendo el mismo. (Figura 2.22). Esta sección contiene tres métodos que están definidos dentro del include “templic2.smd”, por tanto aquí se utiliza su funcionalidad. En el quinto de este capítulo se explica el código del módulo completo.

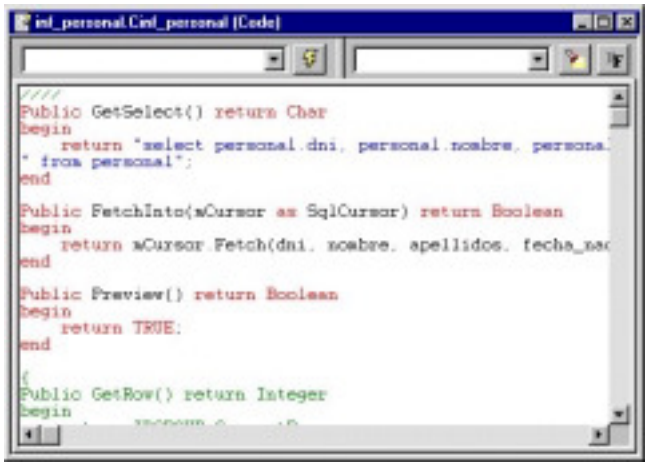


Figura 2.22. Código de la clase Cinf_personal.

5. Código generado para el módulo.

Fichero Inf_personal.smd

```
REPOSITORY mirepositorio
INCLUDES BEGIN
    TEMPLINC2
END
CONSTANTS  begin
End
```

```
CLASSES BEGIN
    Cinf_personal is absPage begin
        objects begin
            dni as personal.dni
            nombre as personal.nombre
            apellidos as personal.apellidos
            fecha_nacimiento as personal.fecha_nacimiento
        end
        INTERFACE
            SIZE 595 842
            MARGIN 24 33 24 66
        BEGIN
            CONTROL IDGROUP AS GROUP
                POSITION 11 39 537 719
                BACKGROUND COLOR transparent
                ATTACH ALL
                BORDER ALL
                GRID VERTICAL
                AS LINES
                REPEAT 0
            BEGIN
                CONTROL AS VAR
                    POSITION 2 0 71 22
                    LABEL "DNI y Letra"
                    VARIABLE dni
                    LIKEVAR
                CONTROL AS VAR
                    POSITION 78 0 146 22
                    LABEL "Nombre"
                    VARIABLE nombre
                    LIKEVAR
                CONTROL AS VAR
                    POSITION 228 0 183 22
                    LABEL "Apellidos"
                    VARIABLE apellidos
                    LIKEVAR
                CONTROL AS VAR
                    POSITION 413 0 123 20
                    FONT "Times New Roman" 12
                    LABEL "Fecha de Nacimiento"
                    VARIABLE fecha_nacimiento
                    ALIGNMENT CENTER
                    DATATYPE DATE MASK 4
            END
            CONTROL AS BOX
                POSITION 11 10 537 29
                BACKGROUND COLOR transparent
                NOLABEL
                ATTACH LEFT RIGHT
                BORDER ALL
            BEGIN
                CONTROL IDTITLE AS TEXT
                    POSITION 79 4 386 20
                    FONT "Times New Roman" 18 BOLD ITALIC
                    ATTACH LEFT RIGHT
                    ALIGNMENT CENTER
                    TAGS "Listado de personal"
                CONTROL IDDATE AS VAR
                    POSITION 11 10 68 10
                    DATATYPE DATE MASK 5
                CONTROL AS TEXT
```

```
                    POSITION 473 3 32 22
                    TAGS "Pag :"
                CONTROL IDPAGE AS VAR
                    POSITION 499 10 26 10
                    ALIGNMENT RIGHT
                    DATATYPE SMALLINT
            END
        END
    end
END
CODE CLASS Cinf_personal BEGIN
//{{CODEBEGIN
////
Public GetSelect() return Char
begin
    return "select personal.dni, personal.nombre,
personal.apellidos, personal.fecha_nacimiento"+
" from personal";
end

Public FetchInto(mCursor as SqlCursor) return Boolean
begin
    return mCursor.Fetch(dni, nombre, apellidos,
fecha_nacimiento).Found;
end

Public Preview() return Boolean
begin
    return TRUE;
end

{
Public GetRow() return Integer
begin
    return IDGROUP.CurrentRow;
end

Public SetRow(row as Integer)
begin
    IDGROUP.CurrentRow = row;
end
}
//{{CODEEND
END
CODE BEGIN
//{{CODEBEGIN
main
objects
begin
oCinf_personal as Cinf_personal
end
begin
Sql.AttachConnection("Conexion");
Sql.Connect("Datos");
oCinf_personal.Run;
Sql.Disconnect;
Sql.DettachConnection;
end
//{{CODEEND
END
```

Análisis del código

En la primera parte igual que en el capítulo 1 se incluye el repositorio que corresponde a nuestro proyecto, “mirepositorio”.

A continuación se realiza la operación de añadir a la sección ***Includes*** del proyecto el módulo “templinc2.smd” que contiene la funcionalidad necesaria para definir diferentes tipos de informes.

```
INCLUDES BEGIN
    TEMPLINC2
END
```

En la sección ***Classes*** se define la nueva clase “Cinf_personal” que hereda de la clase “absPage” definida en “templinc2” con los objetos que tiene internamente, que corresponden con las columnas de la tabla “personal”.

```
CLASSES BEGIN
    Cinf_personal is absPage begin
        objects begin
            dni as personal.dni
            nombre as personal.nombre
            apellidos as personal.apellidos
            fecha_nacimiento as personal.fecha_nacimiento
        end

(....)
END
```

En la parte marcada con puntos suspensivos se realiza la definición de los elementos que componen el informe: título, etiqueta con el número de páginas, fecha, etiquetas de las columnas y contenido de cada una de ellas.

A continuación se encuentra la sección ***Code Class*** que contiene el código de la clase “Cinf_personal” que vimos en la figura 2.22. Se comenta cada uno de los métodos:

```
CODE CLASS Cinf_personal BEGIN
//{{CODEBEGIN
////
Public GetSelect() return Char
begin
    return "select personal.dni, personal.nombre,
personal.apellidos, personal.fecha_nacimiento"+
" from personal";
end

Public FetchInto(mCursor as SqlCursor) return Boolean
begin
    return mCursor.Fetch(dni, nombre, apellidos,
fecha_nacimiento).Found;
end

Public Preview() return Boolean
begin
    return TRUE;
end
```

```
{
Public GetRow() return Integer
begin
    return IDGROUP.CurrentRow;
end

Public SetRow(row as Integer)
begin
    IDGROUP.CurrentRow = row;
end
}
//{{CODEEND
END
```

- GetSelect: realiza una operación de selección del SQL sobre la tabla personal de todas sus columnas.
- FetchInto: devuelve true si el cursor ha procesado alguna línea.
- Preview: devuelve true para que se ejecute la vista previa del informe
- GetRow: no se ejecuta porque se encuentra entre comentarios. Retornaría la fila en curso del control Group
- SetRow: asigna un número a la fila actual del control Group.

Para entender este código es necesario ver el anexo donde se encuentra explicado el código generado por el include “templinc2.smd”. Realmente al ser la clase “Cinf_personal” una clase que hereda de la clase “absPage” que a su vez hereda de “basicPage”, cuando ejecutamos el módulo “Inf_personal” se ejecuta el código definido para esas clases.

Lo siguiente que encontramos es el código del módulo, donde se realiza la conexión con la base de datos y se crea un objeto de la clase “Cinf_personal” y se ejecuta el método “Run” sobre ese objeto. Posteriormente se realiza la desconexión con la base de datos.

```
CODE BEGIN
//{{CODEBEGIN
main
objects
begin
oCinf_personal as Cinf_personal
end
begin
Sql.AttachConnection("Conexion");
Sql.Connect("Datos");
oCinf_personal.Run;
Sql.Disconnect;
Sql.DettachConnection;
end
//{{CODEEND
END
```

6. Archivos generados en este capítulo.

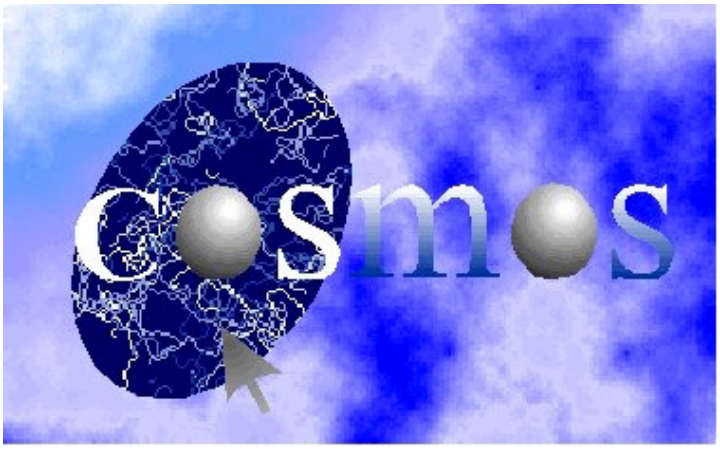
En este capítulo se han generado cuatro archivos nuevos dentro de la carpeta "miproyecto":

Archivo	Contenido
Inf_personal.omd	Fichero objeto generado por el compilador de Cosmos, para el módulo "inf_personal"
Inf_personal.smd	Fichero ASCII que contiene el código fuente del módulo "Inf_personal".
Templinc2.omd	Fichero objeto generado por el compilador de Cosmos, para el include "templinc2".
Templinc2.smd	Fichero ASCII con el código fuente del include "templinc2". Cuando creamos el módulo con el Wizard este realizo la copia de este archivo desde el directorio de "includes" de Cosmos.

Capítulo 3

Edición de los datos de una tabla mediante un control de tipo Grid

1. Introducción.
2. Creación del módulo “ABM_personal” mediante Wizard.
3. Análisis del módulo “ABM_personal” creado con Wizard.
4. Ejecución del módulo “ABM_personal” creado con Wizard.
5. Creación del módulo “ABM_personal” sin utilizar el Wizard de módulos.
6. Asociar el repositorio al módulo “ABM_personal”.
7. Creación de la clase “CABM_personal”.
8. Asociar una tabla a la clase “CABM_personal”.
9. Crear la pantalla de edición con el control Grid.
10. Conexión del módulo “ABM_personal” con la base de datos.
11. Ejecución del módulo “ABM_personal”.
12. Código generado para el módulo “ABM_personal”.
13. Archivos generados en este capítulo.



1. Introducción.

En la primera parte del capítulo, se va a utilizar la tabla "personal" que tenemos en nuestro repositorio, para crear con el asistente de módulos (Wizard) un módulo para editar los datos que contiene dicha tabla con un control de tipo Grid. El aspecto de este control es similar al de una tabla, en la cual se indicará las etiquetas con las que se desea identificar a sus columnas y el número de columnas del control. Para poder mostrar todas las filas y/o columnas del control, este podrá disponer de barra de desplazamiento horizontal y/o vertical.

En la segunda parte vamos a realizar el mismo módulo pero sin utilizar el Wizard de módulos. De esta forma se verá el comportamiento del asistente cuando crea el módulo.

Seguimos dentro del mismo proyecto que se creó en el capítulo primero.

2. Creación del módulo "ABM_Grid" mediante Wizard.

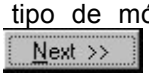
Este módulo realizará **A**ltas, **B**ajas, **M**odificaciones y consultas sobre la tabla "personal" en un control de tipo Grid. Se seguirán los pasos explicados en el apartado 10 del capítulo primero para crear un módulo mediante asistentes.

Seleccionamos de la paleta del proyecto la sección ***Sources*** y pulsamos el botón secundario de ratón, de la lista de opciones que se muestra pulsamos ***Add***. (Véase figuras similares en los anteriores capítulos, figura 1.25 y la figura 2.2)

Se muestra el cuadro de diálogo de las propiedades del módulo, los datos que introducimos en los diferentes campos son:

Propiedad	Contenido
Label	Edición personal (Grid)
Comment	Editar el contenido de la tabla personal mediante un control Grid
Type	Program
File name	ABM_Grid
Path	Se mantiene el mostrado por defecto

Pulsamos el botón  para invocar al Wizard de módulos. Los pasos que efectuamos en el Wizard son:

1. Seleccionamos el tipo de módulo que se quiere realizar, elegimos ***Standard Form***. Pulsamos el botón 

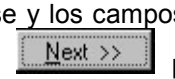
para continuar con el asistente. (Figura 3.1, paso 1)

2. Asociamos el repositorio "miRepositorio" al módulo y aceptamos el nombre que se asigna automáticamente a la clase que se creará para el nuevo módulo. Pulsamos el botón 

para continuar. (Figura 3.1, paso 2)

3. Seleccionamos el tipo de Form que queremos asignar al módulo, como queremos tener un control de Grid, seleccionamos Form de líneas y señalamos la casilla para que tenga barra de estado. Pulsamos el botón 

para continuar. (Figura 3.1, paso 3)

4. Se pide que se seleccione la tabla que estará asignada a la nueva clase y los campos de dicha tabla que aparecerán en el control Grid. Pulsamos el botón 

para continuar. (Figura 3.1, paso 4)

5. Seleccionamos que se establezca la conexión con la base de datos. Para ello marcamos la casilla de selección y en el cuadro que se muestra seleccionamos “Conexión” de la lista desplegable que contiene las diferentes conexiones especificadas en el Editor de Configuración. Seleccionamos la base de datos que queremos utilizar. Pulsamos el botón Finish para terminar con la ejecución del Wizard. (Figura 3.1, paso 5)

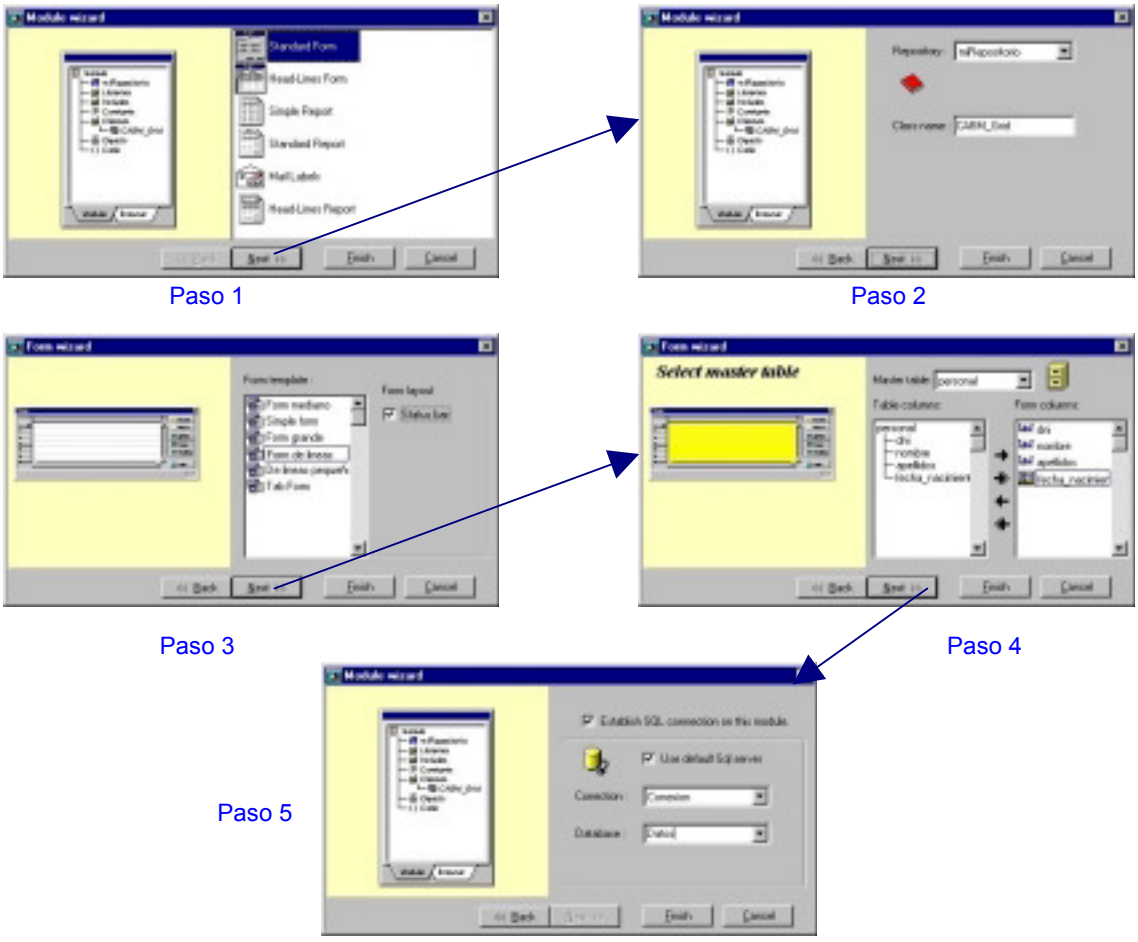


Figura 3.1. Pasos del Wizard de módulos.

3. Análisis del módulo “ABM_Grid” creado con Wizard.

En la paleta del proyecto seleccionamos el nuevo módulo y hacemos doble click de ratón sobre él para abrir su paleta. El aspecto de la paleta del nuevo módulo es el que se muestra en la figura 3.2.

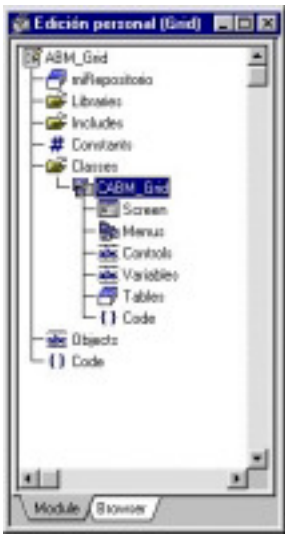


Figura 3.2. Aspecto de la paleta del nuevo módulo.

En la Figura 3.3. se muestran el contenido de la seccion ***Controls*** y la tabla personal que contiene sección ***Tables*** de la clase CABM_Grid mostrada en la figura 3.2.

Figura 3.3. Contenido de las secciones Controls y Tables.

En ***Controls*** se encuentra definido el control que hemos seleccionado en el Wizard, que era un Grid. En ***Tables*** se encuentran definidas las columnas de la tabla “personal” que aparecen en el control Grid.

La sección Screen de la clase CABM_Grid (clase del módulo ABM_Grid), que se muestra en la figura 3.4. tiene los siguientes componentes:

- El control Grid que tiene definidos una serie de campos edit con el contenido de las columnas de la tabla “personal”.
- Barras de desplazamientos para permitir moverse por los campos del control Grid.
- Conjunto de seis botones que permiten buscar, añadir, borrar, modificar, posicionarse en el última fila para añadir una persona y salir de la ejecución del módulo. Los botones tienen la misma funcionalidad y definición que los explicados en el capítulo primero de este manual.

Figura 3.4. Sección Screen de la clase CABM_Grid.

Las propiedades del control Grid se pueden ver en la figura 3.5. En la pestaña ***General*** se especifica el nombre o identificador que se ha dado a este control (IDMASTER), este nombre puede ser cualquier cadena de caracteres.

En la pestaña ***Special*** se especifica que el control tendrá Barra de desplazamiento vertical y horizontal. También se marca la característica ***Parent Grid***. Permite definir o no el control como padre de otros. Es necesario marcar esta casilla de verificación cuando desee asociar una tabla de líneas al control en un mantenimiento de cabeceras líneas como en este caso.

En la pestaña ***Storage*** se marca la opción de ***Form table*** para asociar una tabla al control Grid y se selecciona de la lista desplegable la tabla “personal”. Estas dos opciones aparecen sombreadas para no poder modificar el valor dado por el Wizard de módulos.

Figura 3.5. Propiedades del control Grid.

Por último en la sección ***Code*** del módulo se encuentran definidas las sentencias necesarias para la conexión con la base de datos y la ejecución del método Run de la clase CABM_Grid invocándole a través de un objeto que se define en la sección de objetos. (Figura 3.6).

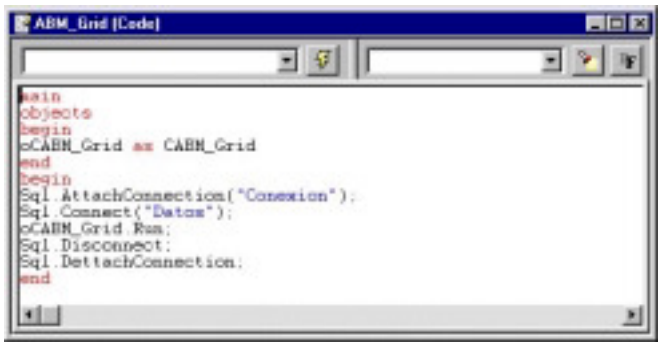


Figura 3.6. Código del módulo ABM_Grid.

4. Ejecución del módulo "ABM_Grid" creado con Wizard.

Vamos a ver ahora el aspecto del nuevo módulo ejecutándose. Para ello seleccionamos la sección Code del módulo y pulsamos  . También se puede ejecutar mediante las otras formas definidas en el capítulo primero de este manual.

Figura 3.7. Ejecución del módulo.

El funcionamiento es el mismo que el visto para la ejecución del módulo ABM del capítulo primero, con la diferencia de que se muestran los datos en el control de tipo Grid.

5. Creación del módulo "ABM_personal" sin utilizar el Wizard de módulos.

Vamos a realizar un módulo con la misma funcionalidad que el creado mediante el Wizard de módulos en los apartados anteriores.

Para crear el módulo procedemos de la siguiente forma, seleccionamos la sección ***Sources*** dentro de la paleta del proyecto y pulsando el botón secundario del ratón secundario del ratón seleccionamos la opción ***Add***, como se muestra en la figura 3.8.

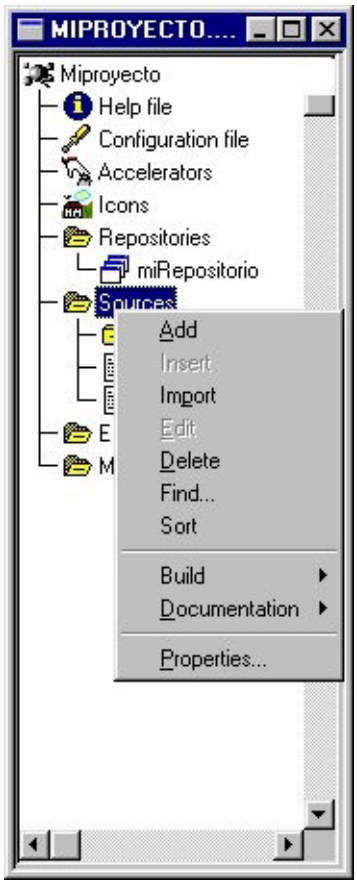


Figura 3.8. Añadir un nuevo módulo.

En el cuadro de diálogo que aparece se introducen las propiedades del nuevo modulo, que serán las siguientes:

Propiedad	Contenido
Label	Edición personal (sin Wizard)
Comment	Edita el contenido de la tabla personal
Type	Program
File name	ABM_personal
Path	Se mantiene el mostrado por defecto

Pulsando 

tenemos el nuevo módulo creado y se muestra en la paleta del proyecto con la etiqueta asignada en lugar del nombre del archivo.

6. Asociar el repositorio al módulo "ABM_personal".

Seleccionamos el nuevo módulo y hacemos doble click de ratón sobre él para abrir su paleta.(Figura 3.9)

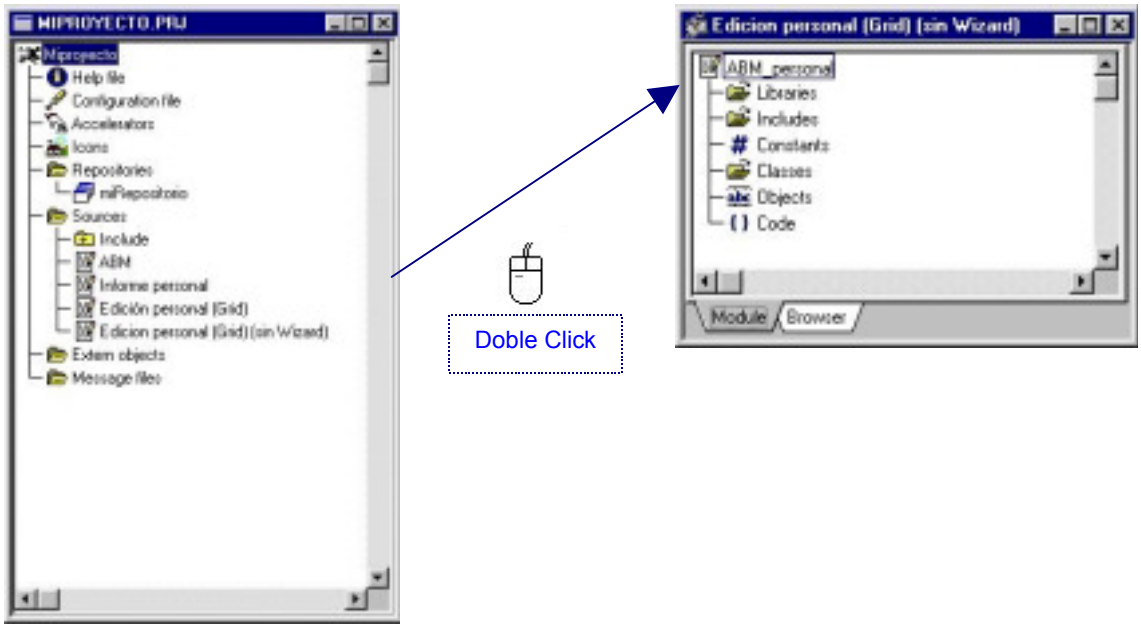


Figura 3.9. Paleta del nuevo módulo.

Vamos a asociar el repositorio “miRepositorio” al módulo “ABM_personal”. Para ello seleccionamos de la paleta del módulo el nombre del mismo y pulsamos el botón secundario del ratón, seleccionando de la lista el único campo activado ***Properties...***

Se muestra el cuadro de diálogo de la Figura 3.10. donde debemos seleccionar el nombre del repositorio que estará asignado a este módulo que será “miRepositorio”.

El campo Message file indica el fichero de mensaje que queremos asociar al módulo. Los ficheros de mensajes son ficheros ASCII que permiten definir textos fuera de los módulos de programación. Estos textos podrán ser utilizados posteriormente en la aplicación. En este momento no se especificará ninguno.

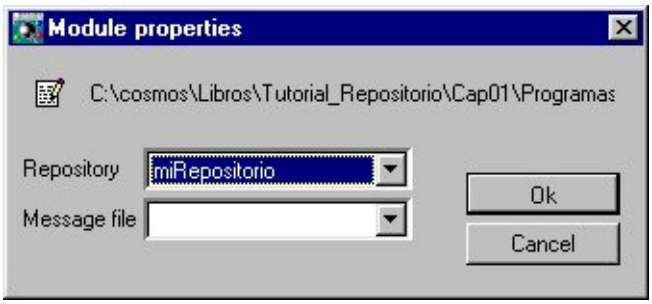


Figura 3.10. Seleccionar el repositorio.

Después de pulsar el botón 

en la paleta del módulo queda asignado el repositorio.

7. Creación de la clase “CABM_personal”.

Sobre la sección ***Classes*** del nuevo módulo pulsamos el botón secundario del ratón y seleccionamos ***Add*** para añadir una clase nueva a este módulo. (Figura 3.11)

Figura 3.11. Añadir una nueva clase al módulo.

En el cuadro de diálogo que se muestra para añadir la nueva clase se han rellenado los campos como se muestra en la figura 3.12. Se selecciona de la lista desplegable la clase Form para que la nueva clase herede de la misma con acceso publico.

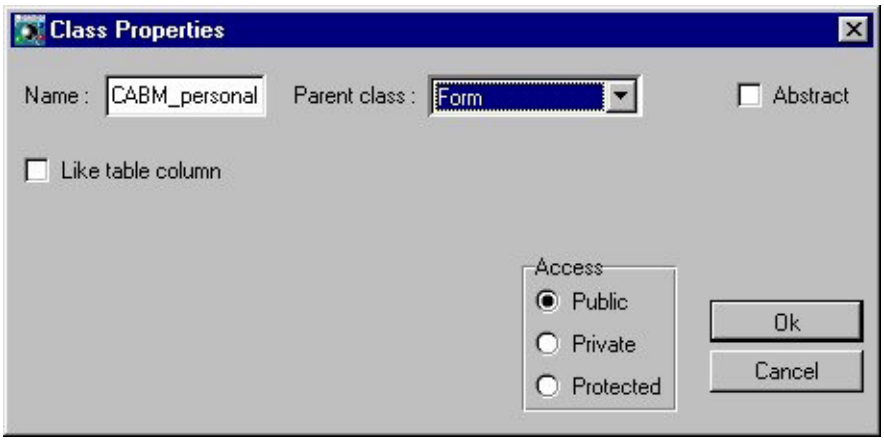


Figura 3.12. Propiedades de la nueva clase.

Se ha seleccionado la clase Form porque se va realizar una visualización de los datos de la tabla personal. La clase Form es la tiene definida toda la funcionalidad de manejo de pantallas. El tipo de acceso que se introduce para esta nueva clase es ***Public***. El contenido de la paleta del módulo después de añadir esta nueva clase se muestra en la figura 3.13.

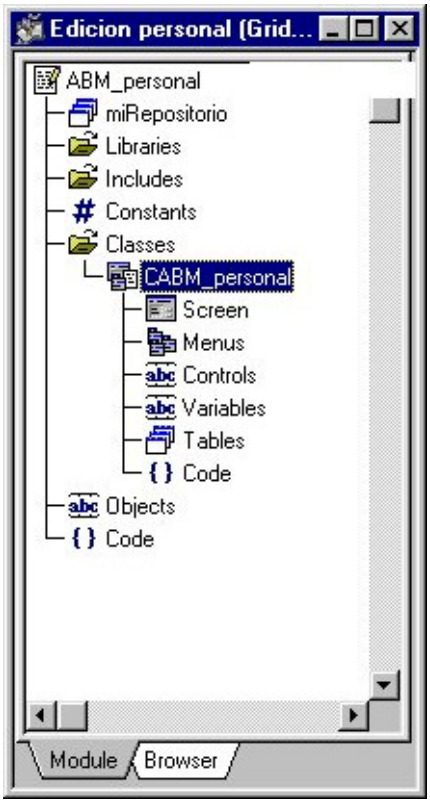


Figura 3.13. Contenido de la paleta del módulo.

8. Asignar una tabla a la clase "CABM_personal".

Ahora vamos a asignar una tabla a la clase CABM_personal para que la utilice como tabla de trabajo. Seleccionamos la sección ***Tables*** dentro de la clase CABM_personal en la paleta del módulo y pulsamos el botón secundario del ratón. De la lista que se nos muestra seleccionamos la única opción activada que es ***Add***. Se muestra un cuadro de diálogo donde se selecciona la tabla del repositorio que queremos asignar a la nueva tabla de la clase y el nombre de esta. Seleccionamos la tabla "personal" y dentro de la clase también la llamamos "personal". (Figura 3.14)

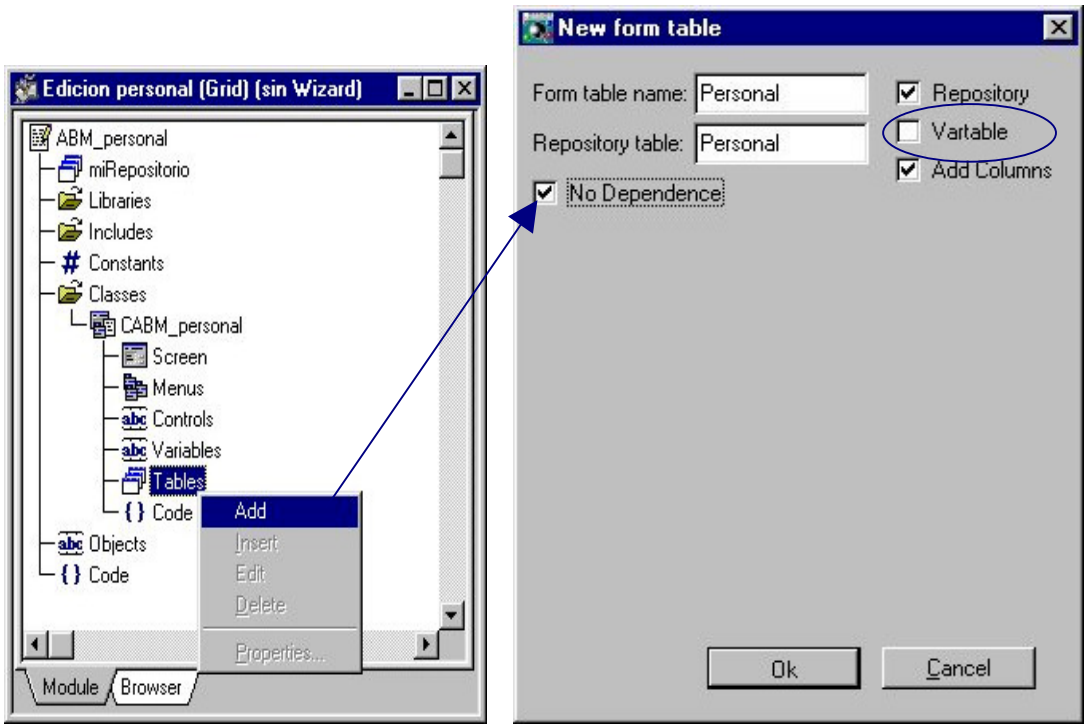


Figura 3.14. Añadir una tabla a la clase.

Las tablas de tipo ***Vartable*** son tablas que se mantienen en memoria y no se almacenan en ninguna base de datos. En este módulo queremos hacer la edición de una tabla de la base de datos luego desactivamos la casilla de selección de ***Vartable***.

La paleta del proyecto refleja esa nueva asignación y si hacemos doble click de ratón sobre la tabla “personal”, se muestran los campos de esta y el tipo de cada uno de ellos. (Figura 3.15).

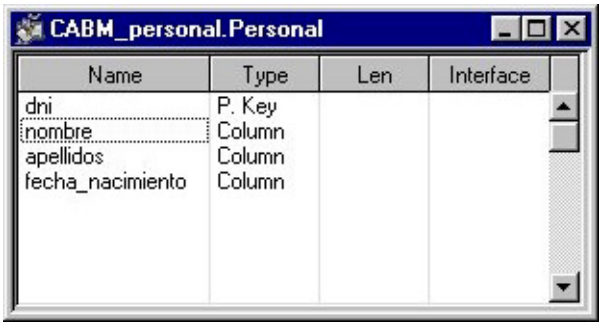


Figura 3.15. Campos de la tabla “personal”.

9. Crear la pantalla de edición con el control Grid.

En este momento ya se puede crear la pantalla para la visualización de los datos. Seleccionamos de la paleta del módulo la sección ***Screen***, aparece la screen vacía que se debe rellenar. (Figura 3.16)

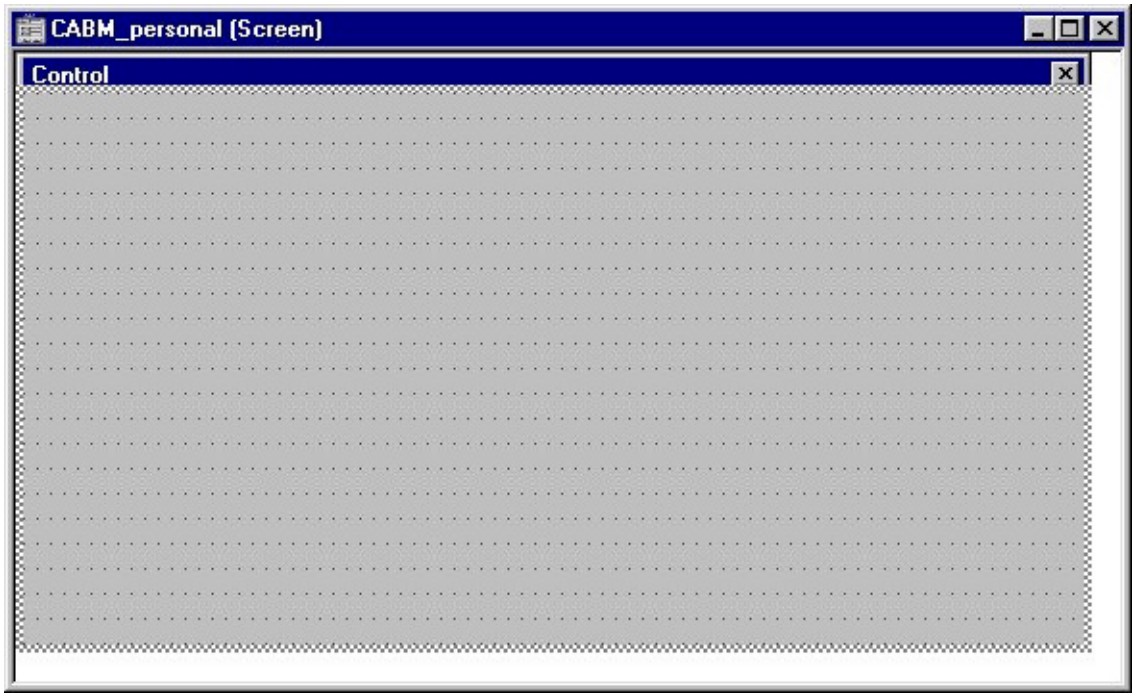


Figura 3.16. La sección Screen de la clase cVisual_personal.

Añadir el control Grid.

Para asignar este control a la screen lo seleccionamos en la paleta de objetos y lo arrastramos sobre el tapiz de la screen. (Figura 3.17)

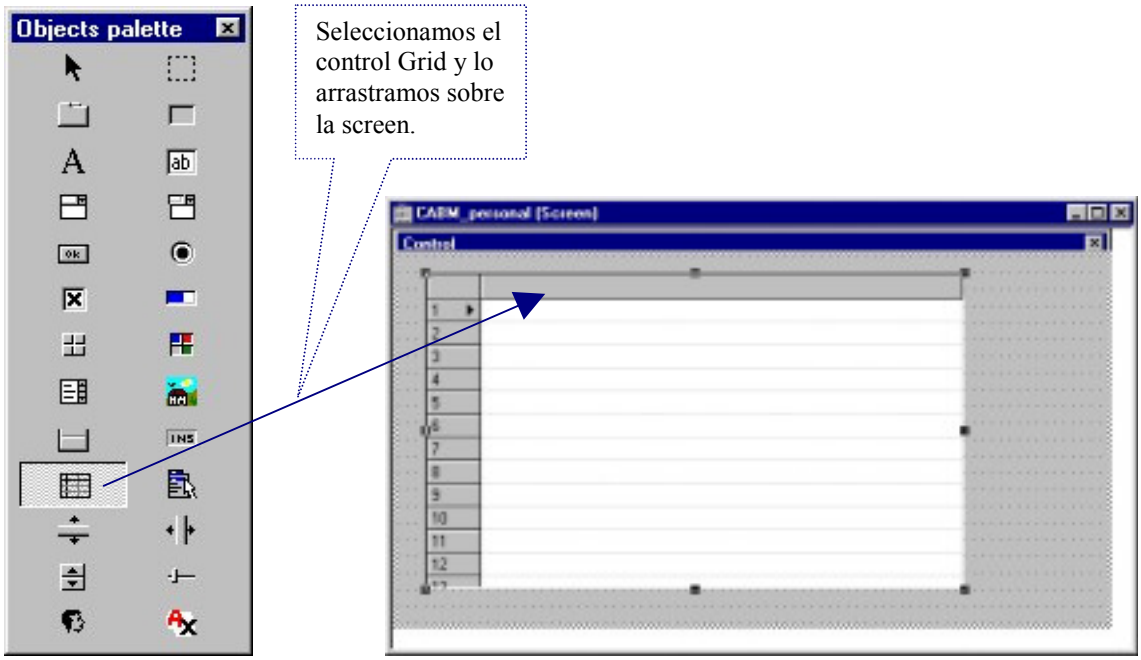


Figura 3.17. Añadir un control Grid a la screen.

Seleccionando este control, editamos sus propiedades haciendo doble click de ratón sobre él o pulsando el botón secundario del ratón y escogiendo la opción ***Properties.*** Se nos muestra un cuadro de diálogo con tres pestañas, el contenido de las dos primeras se muestra en la figura 3.18.

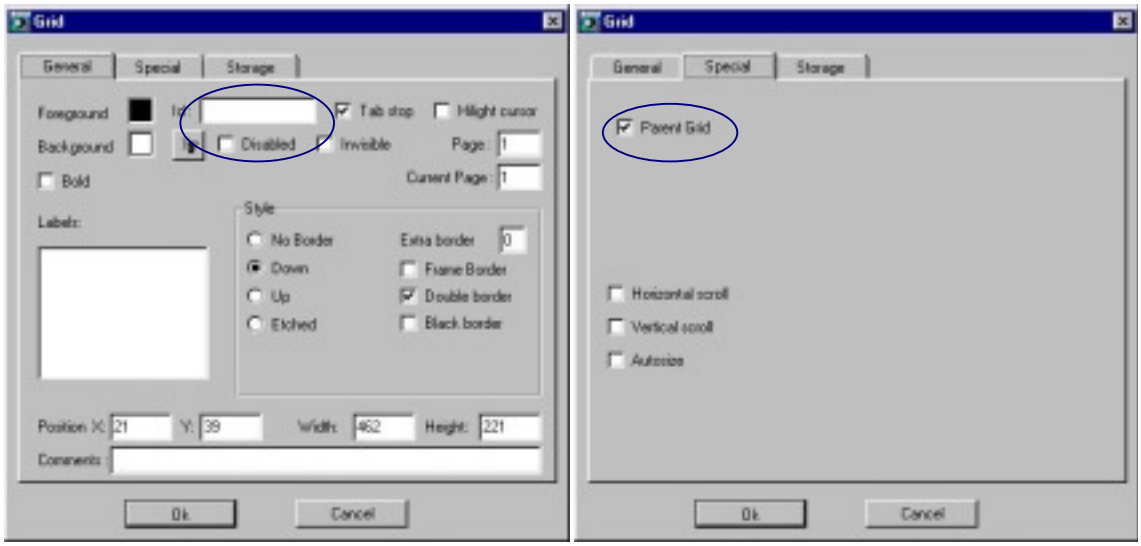


Figura 3.18. Propiedades del control Grid.

En la pestaña ***General*** añadimos el identificador para este control que será “miGrid” en el campo de edición llamado ***id***.

En la pestaña ***Special*** marcamos la casilla de selección con la etiqueta ***Parent Grid***, que permite que este control sea contenedor de otros.

En la pestaña ***Storage*** vamos a seleccionar la tabla del repositorio que vamos a asociar a este control. Como se ve en la figura 3.19. seleccionamos la casilla etiquetada como ***Form table*** y se muestra una lista desplegable ***Table*** donde se selecciona la tabla “Personal”

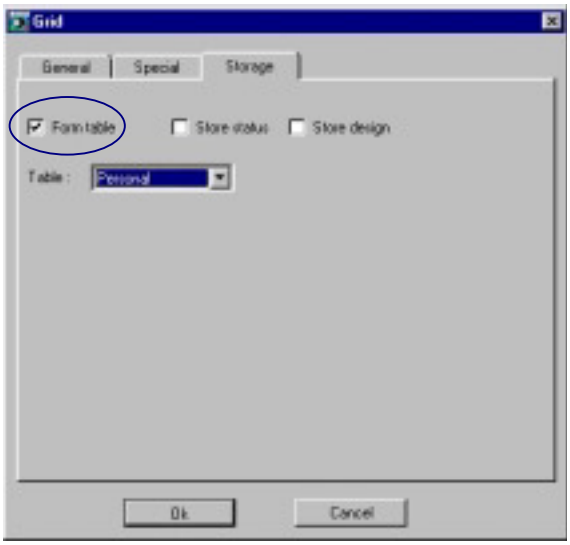


Figura 3.19. Propiedades de almacenamiento de miGrid.

Pulsamos  para hacer efectivos los cambios en las propiedades del control.

Añadir los controles Edit.

Para poder introducir datos en la tabla “Personal” utilizando el control ***Grid*** necesitamos añadir a este control una serie de controles de tipo ***Edit*** a los que se asigna cada una de las columnas de la tabla.

Para asignar estos controles seleccionamos de la paleta de objetos el control ***Edit*** y lo arrastramos sobre el control ***Grid*** como podemos ver en la figura 3.20.

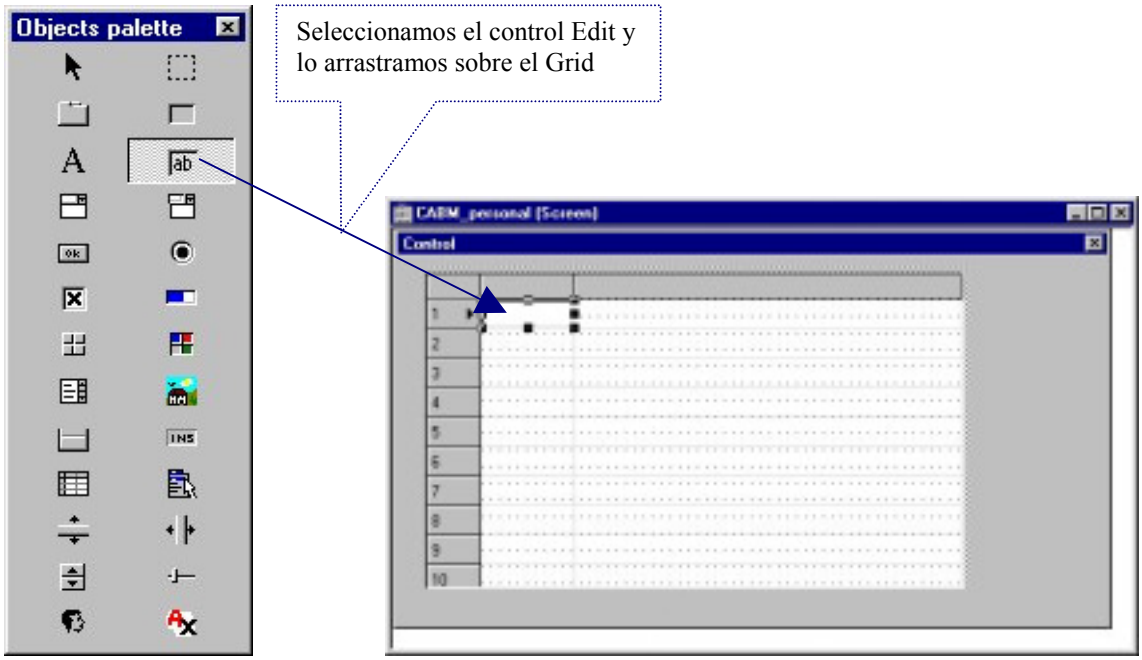


Figura 3.20. Añadir el control Edit al Grid.

En la figura 3.18. asignamos la propiedad ***Parent Grid*** que definía al control ***Grid*** como contenedor de otros controles, y es esta propiedad la que nos permite añadir controles de tipo ***Edit*** al ***Grid.***

Añadimos otros tres controles ***Edit*** de la misma forma que en la figura 3.20. Seleccionamos el primer ***Edit*** y hacemos doble click sobre él para abrir sus propiedades. En la figura 3.21. vemos dos de las tres pestañas de las propiedades de este control.

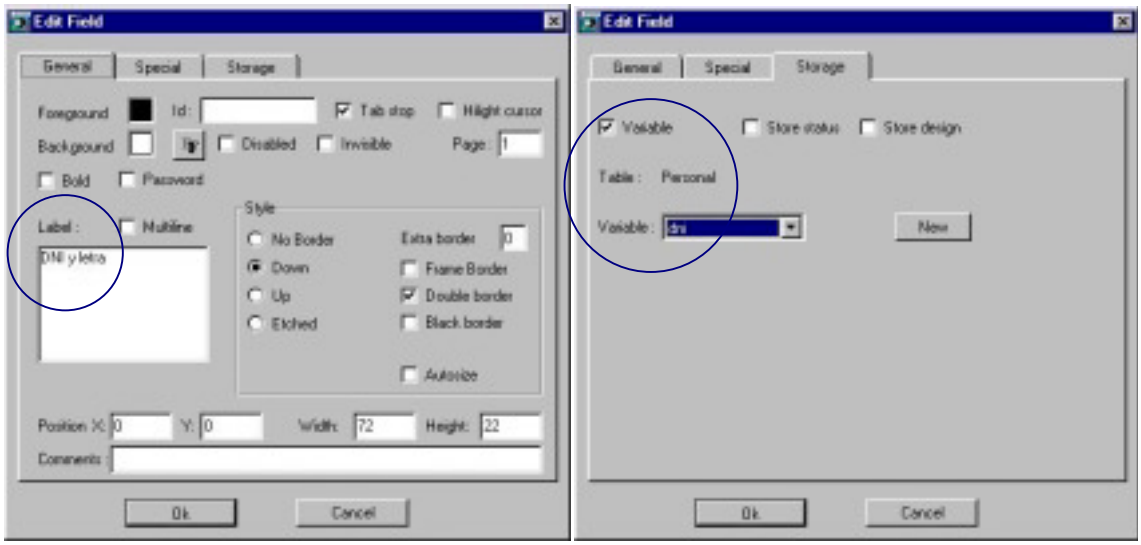


Figura 3.21. Propiedades del control Edit.

En la pestaña ***General*** añadimos en la propiedad ***Label*** la etiqueta que aparecerá sobre esa columna.

En la pestaña ***Special*** no realizamos ninguna modificación. Sobre la pestaña ***Storage*** seleccionamos la casilla etiquetada como ***Variable***, y se nos muestra la lista desplegable donde seleccionamos la columna de la tabla personal que queremos asigna a este control.

Estos pasos los repetimos para cada uno de los controles ***Edit*** que hemos añadido. A cada uno le asignamos la etiqueta y la columna de la tabla que queremos asociarle. El aspecto que tiene la ***Screen*** después de estos pasos es el que se muestra en la figura 3.22.

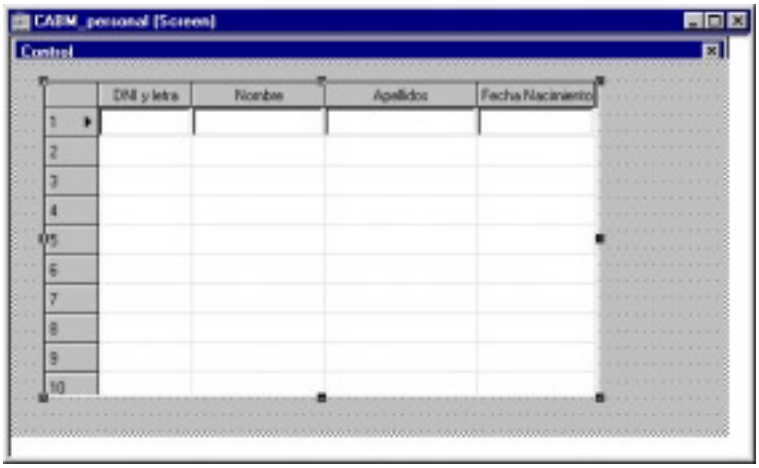


Figura 3.22. Aspecto actual de la Screen.

Añadir los botones.

Vamos a añadir los controles de tipo Button que contendrán la funcionalidad necesaria para realizar las altas, bajas, modificaciones y consultas sobre la tabla personal.

Seleccionamos el control Button de la paleta de objetos y le arrastramos sobre la Screen. Para poder añadir mas botones sin tener que hacer uso de la paleta de objetos podemos realizar una copia del botón que ya se encuentra en la Screen. Para realizar esto seleccionamos el botón y manteniendo pulsada la tecla CTRL se realiza la copia que podemos arrastrar hasta la posición deseada.(Figura 3.23)

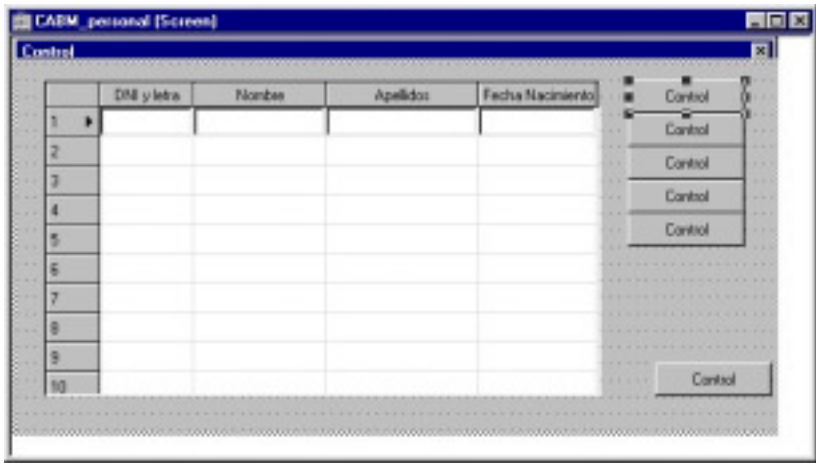


Figura 3.23. Añadir los controles de tipo botón.

Seleccionando el primer botón vamos a modificar sus propiedades para que realice la funcionalidad deseada. En la figura 3.24 se muestran las dos pestañas que se modifican.

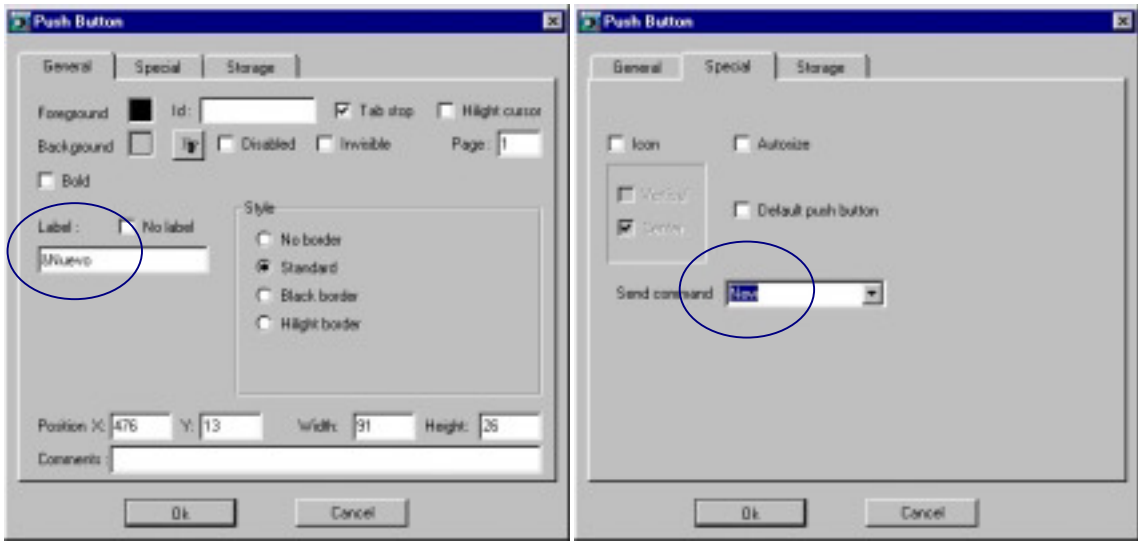


Figura 3.24. Propiedades del botón Nuevo.

En la pestaña ***General*** hemos añadido la etiqueta que se muestra sobre el botón. Si se escribe el símbolo **&** delante de una de las letras de la etiqueta se marca dicha letra como ***hot key*** para el acceso al botón.

En la pestaña ***Special*** se selecciona el comando que enviará este botón cuando se pulse. El comando se selecciona de la lista desplegable. En la pestaña ***Storage*** no se realiza ninguna modificación.

El resto de los botones se modifican de la misma forma según la siguiente tabla:

Botón	General (Label)	Special (Send Command)
Nuevo	&Nuevo	New
Añadir	&Añadir	Add
Modificar	&Modificar	Update
Borrar	&Borrar	Delete
Buscar	B&uscar	QueryByForm
Salir	&Salir	Close

El aspecto final de la Screen es el que se muestra en la figura 3.25.

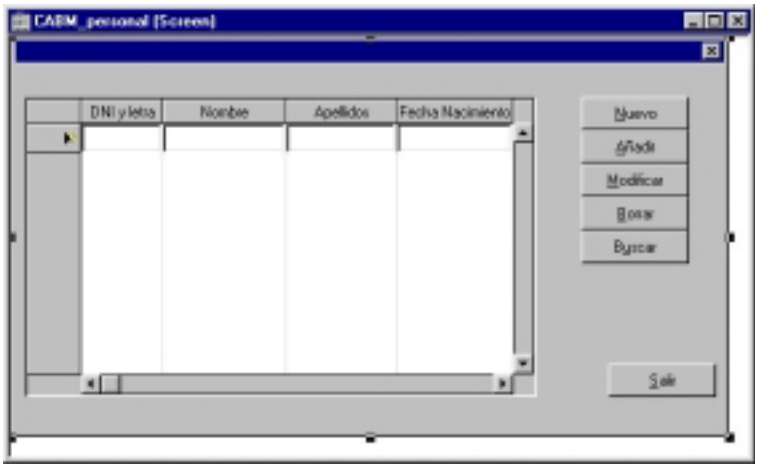


Figura 3.25. Aspecto de la Screen después de añadir los botones.

Añadir el título a la Screen.

Vamos a añadir el título a la Screen. Para ello seleccionamos el ***Frame*** que es el marco de la Screen. En la figura 3.25. se encuentra seleccionado dicho marco. Si hacemos doble click sobre el mismo o pulsamos el botón secundario del ratón y seleccionamos ***Properties...*** se muestra las propiedades del ***Frame*** de la ***Screen.*** (Figura 3.26)

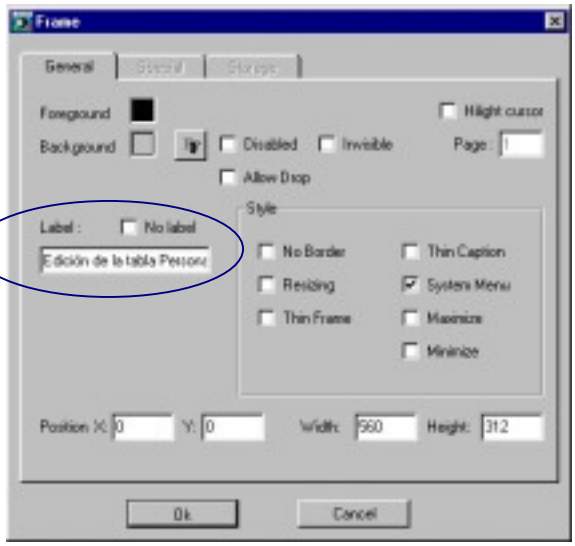


Figura 3.26. Propiedades de Frame.

En la única pestaña activada, ***General***, se añade la etiqueta "Edición de la tabla Personal con un control Grip" que será el título que se mostrará como título del Screen, como se puede ver en la figura 3.27.

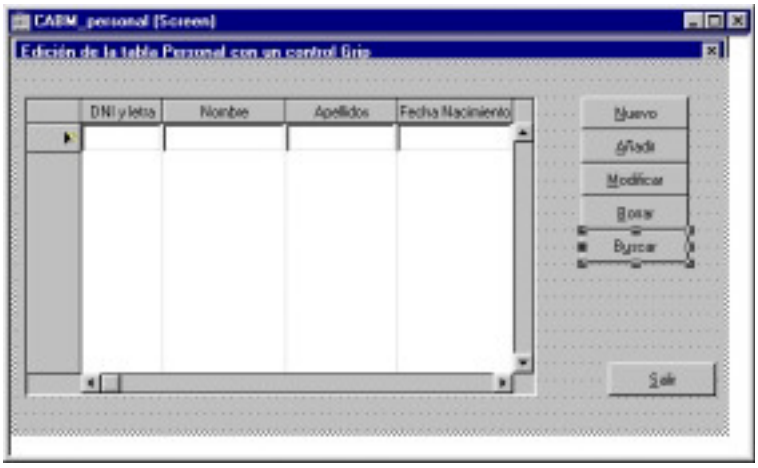


Figura 3.27. Screen final.

10. Conexión del módulo "ABM_personal" con la base de datos.

Cuando realizamos el módulo "ABM_grid" mediante el Wizard en el último paso de este (Paso 5 de la Figura 3.1) se seleccionaba conectar el módulo con una base de datos. En el módulo actual vamos a realizar esta conexión "a mano".

La conexión se realiza en la seccion Code del módulo para ello seleccionamos esta sección y hacemos doble click del ratón sobre ella para abrir el editor. (Figura 3.28)

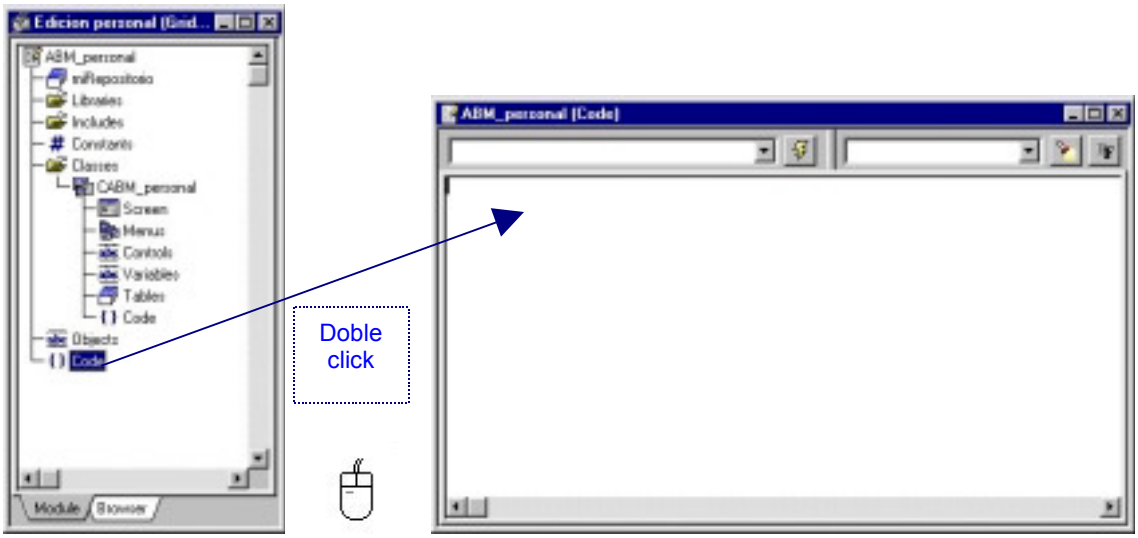


Figura 3.28. Editor del código del módulo “ABM_personal”.

En este editor escribimos las siguientes líneas, que es donde se realiza la conexión a la base de datos, y se invoca al método Run de la clase CABM_personal mediante un objeto. Este método Run realiza la visualización de la Screen creada en la clase.

```
main
objects
begin
 oCABM_personal as CABM_personal
end
begin
 Sql.AttachConnection("Conexion");
 Sql.Connect("Datos");
 oCABM_personal.Run;
 Sql.Disconnect;
 Sql.DettachConnection;
end
```

El aspecto del editor de código del módulo “ABM_personal” después de introducir este código es el que se muestra en la Figura 3.29.

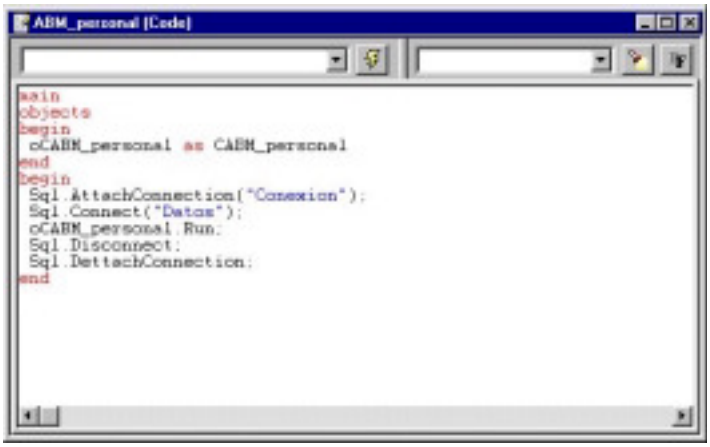


Figura 3.29. Sección de código del módulo “ABM_personal”.

11. Ejecución del módulo "ABM_personal".

Vamos a ver ahora el aspecto del nuevo módulo ejecutándose. Para ello seleccionamos la sección Code del módulo "ABM_personal" y pulsamos  . También se puede ejecutar mediante las otras formas definidas en el capítulo primero de este manual. El aspecto del módulo ejecutandose es el que se muestra en la Figura 3.30.

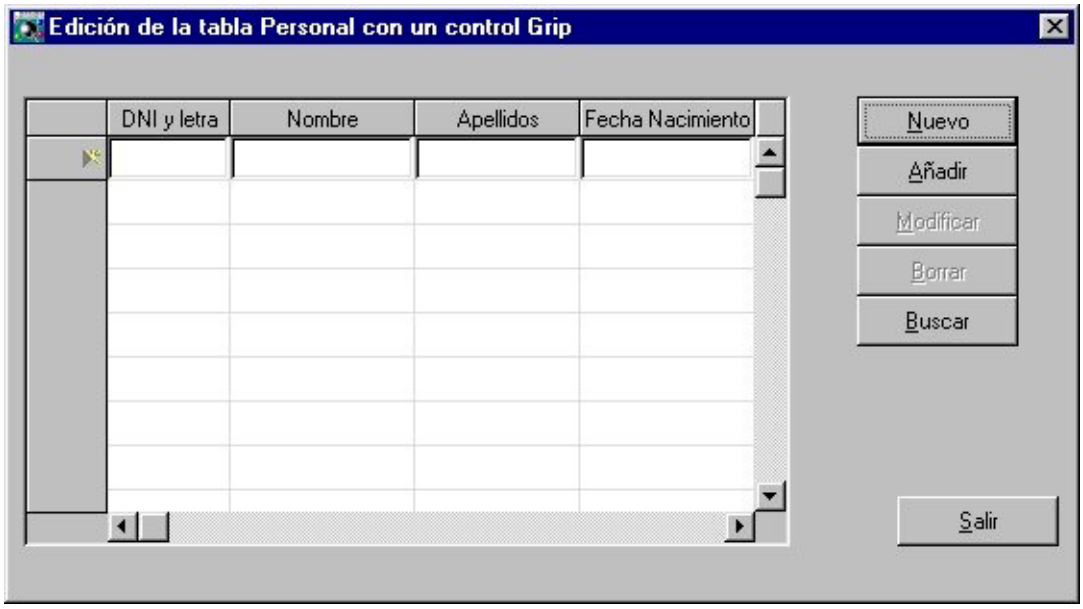


Figura 3.30. Ejecución del módulo ABM_personal.

12. Código generado para el módulo "ABM_personal".

El código que se ha generado para el modulo que se acaba de crear es el que se muestra a continuación. En este código no existe ninguna característica especial al comentado en el capítulo primero, excepto que se utiliza un control de tipo grid contenedor de una serie de controles edit.

```
REPOSITORY mirepositorio
CONSTANTS  begin
end
CLASSES BEGIN
    CABM_personal is Form begin
        objects begin
        end
        Table Personal is personal
        begin
            dni as column required
            nombre as column
            apellidos as column
            fecha_nacimiento as column
        end
        INTERFACE
            POSITION 0 0 560 312
            LABEL "Edición de la tabla Personal con un control
                  Grip"
            SYSMENU
        BEGIN
```

```
CONTROL AS BOX
    POSITION 0 0 552 287
    FOREGROUND RGB 0 0 0
    ATTACH ALL
    NOLABEL
    NOBORDER
BEGIN
    CONTROL AS BUTTON
        POSITION 441 26 84 26
        LABEL "&Nuevo"
        COMMAND New
    CONTROL AS BUTTON
        POSITION 441 52 84 26
        LABEL "&Añadir"
        COMMAND Add
    CONTROL AS BUTTON
        POSITION 441 78 84 26
        LABEL "&Modificar"
        COMMAND Update
    CONTROL AS BUTTON
        POSITION 441 130 84 26
        LABEL "B&uscar"
        COMMAND QueryByForm
    CONTROL AS BUTTON
        POSITION 441 104 84 26
        LABEL "&Borrar"
        COMMAND Delete
    CONTROL AS BUTTON
        POSITION 462 234 84 26
        LABEL "&Salir"
        COMMAND Close
    CONTROL miGrid AS GRID
        POSITION 7 26 399 234
        BORDER DOUBLE DOWN
        SCROLL VERTICAL HORIZONTAL
        TABLE Personal
        PARENTGRID
    BEGIN
        CONTROL AS EDIT
            POSITION 0 0 63 22
            LABEL "DNI y letra"
            BORDER DOUBLE DOWN
            VARIABLE Personal.dni
            DATATYPE CHAR
        CONTROL AS EDIT
            POSITION 64 0 95 22
            LABEL "Nombre"
            BORDER DOUBLE DOWN
            VARIABLE Personal.nombre
            DATATYPE CHAR
        CONTROL AS EDIT
            POSITION 160 0 85 22
            LABEL "Apellidos"
            BORDER DOUBLE DOWN
            VARIABLE Personal.apellidos
            DATATYPE CHAR
        CONTROL AS EDIT
            POSITION 246 0 90 22
            LABEL "Fecha Nacimiento"
            BORDER DOUBLE DOWN
            VARIABLE Personal.fecha_nacimiento
```

```
                        DATATYPE CHAR
                    END
                END
            END
        end
    END
    CODE CLASS CABM_personal BEGIN
    //{{CODEBEGIN

    //{{CODEEND
    END
    CODE BEGIN
    //{{CODEBEGIN
    main
    objects
    begin
     oCABM_personal as CABM_personal
    end
    begin
     Sql.AttachConnection("Conexion");
     Sql.Connect("Datos");
     oCABM_personal.Run;
     Sql.Disconnect;
     Sql.DettachConnection;
    end
    //{{CODEEND
    END
```

13. Archivos generados en este capítulo.

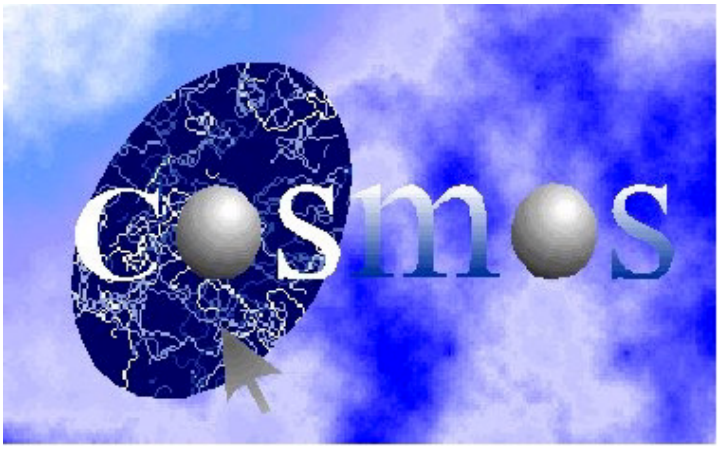
En este capítulo se han generado cuatro archivos nuevos dentro de la carpeta "miproyecto":

Archivo	Contenido
ABM_grid.omd	Fichero objeto generado por el compilador de Cosmos, para el módulo "ABM_grid"
ABM_grid.smd	Fichero ASCII que contiene el código fuente del módulo "ABM_grid".
ABM_personal.omd	Fichero objeto generado por el compilador de Cosmos, para el módulo "ABM_personal".
ABM_personal.smd	Fichero ASCII que contiene el código fuente del módulo "ABM_personal".

Capítulo 4

Utilización de tablas maestro-detalle

1. Introducción.
2. Campos de las diferentes tablas.
3. Diagrama entidad-relación.
4. Creación de las tablas mediante el Editor de Repositorio.
5. Definición de las claves primarias.
6. Definición de claves referenciales/Joins.
7. Crear la base de datos.
8. Crear el proyecto.
9. Compilación del proyecto.
10. Ejecución del proyecto.
11. Pantalla e informe de cabecera-líneas.
12. Personalización del menú.
13. Código generado.
14. Archivos generados en este capítulo.



1. Introducción.

En este capítulo vamos a crear un proyecto que contenga una aplicación con dos tablas, empleados y departamentos de una empresa. En un departamento hay varios empleados, y un empleado únicamente pertenece a un departamento.

En los primeros pasos se creara mediante el Editor de Repositorios la estructura de tablas que necesitamos, y después mediante el Wizard de módulos crearemos el proyecto de la aplicación.

Las tablas son empleados y departamento. Entre ambas tablas existe una relación 1 a n, es decir un empleado pertenece a un único departamento y en un departamento trabajan varios (n) empleados. Esta relación es la que establece la estructura maestro-detalle entre las dos tablas.

2. Campos de las diferentes tablas.

Empleados

Contiene los datos personales de los diferentes empleados de la empresa, así como el departamento al que pertenecen.

- NIF: número de identificación fiscal del empleado.
- Nombre: nombre del empleado.
- Apellidos: apellidos del empleado.
- Domicilio: calle, número y piso de la dirección del empleado.
- Teléfono: teléfono de contacto del empleado.
- Código del departamento: código del departamento al que pertenece el empleado.

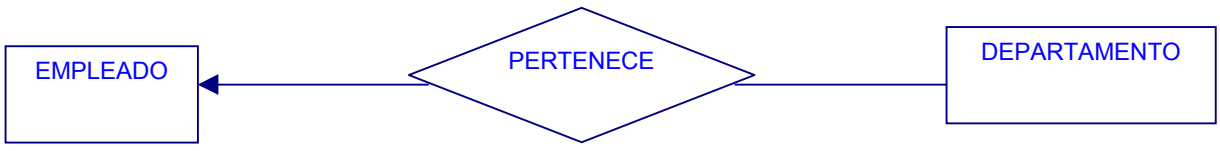
Departamento

Datos de los diferentes departamentos de la empresa.

- Código del departamento: número que identifica al departamento dentro de la empresa.
- Nombre: nombre del departamento dentro de la empresa.

3. Diagrama entidad-relación.

El diagrama entidad-relación de esta sencilla aplicación será:



4. Creación de las tablas mediante el Editor de Repositorio.

Vamos a ejecutar el Editor Visual y seleccionamos la opción ***New*** del menú desplegable como vemos en la figura 4.1. o pulsamos el icono  .

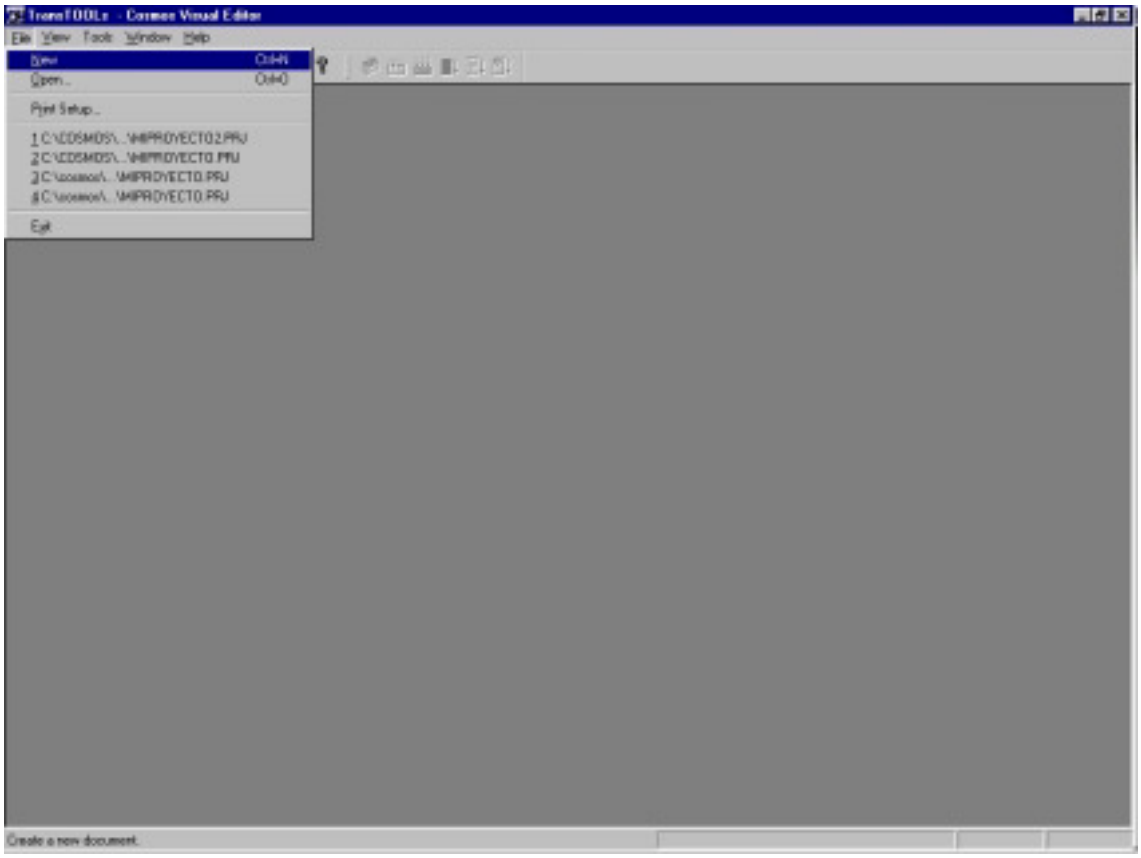


Figura 4.1. Seleccionar la opción New.

Se muestra un cuadro de diálogo que nos permite seleccionar el tipo de documento que se quiere crear. Se selecciona ***Repository Document*** para crear el repositorio del proyecto. (Figura 4.2)

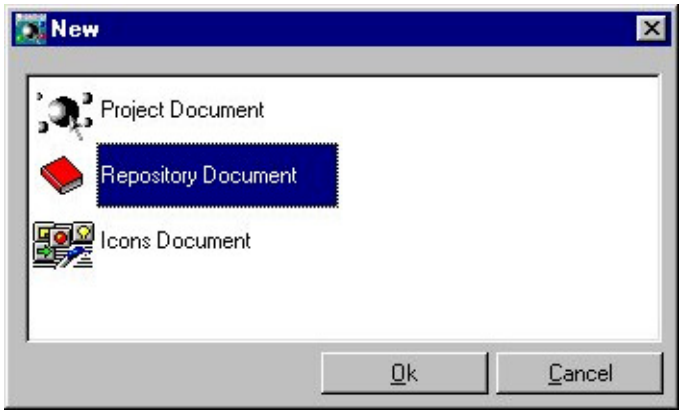


Figura 4.2. Seleccionar la creación de un repositorio.

Se ejecuta el Editor de Repositorio para poder crear el repositorio del proyecto. (Figura 4.3). Para la creación de las tablas seguiremos los pasos dados en el apartado cuarto del capítulo primero de este manual.

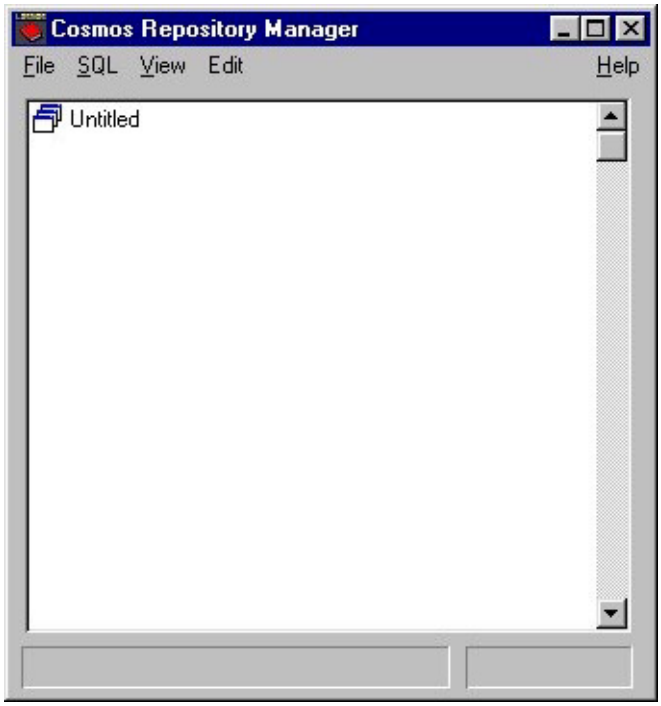


Figura 4.3. Editor de Repositorio.

Empleados

Creamos esta tabla con los campos especificados en el apartado 3 de este capítulo. Añadimos las columnas dni, nombre, apellidos como ya conocemos. Para añadir la dirección vamos a utilizar una estructura. Una estructura es un conjunto de columnas pero que no representa una entidad dentro de la base de datos, solo tiene representación en el repositorio. En la figura 4.4. se ven los campos introducidos hasta el momento y la selección mediante la pulsación del botón secundario del ratón de la opción ***New structure***.

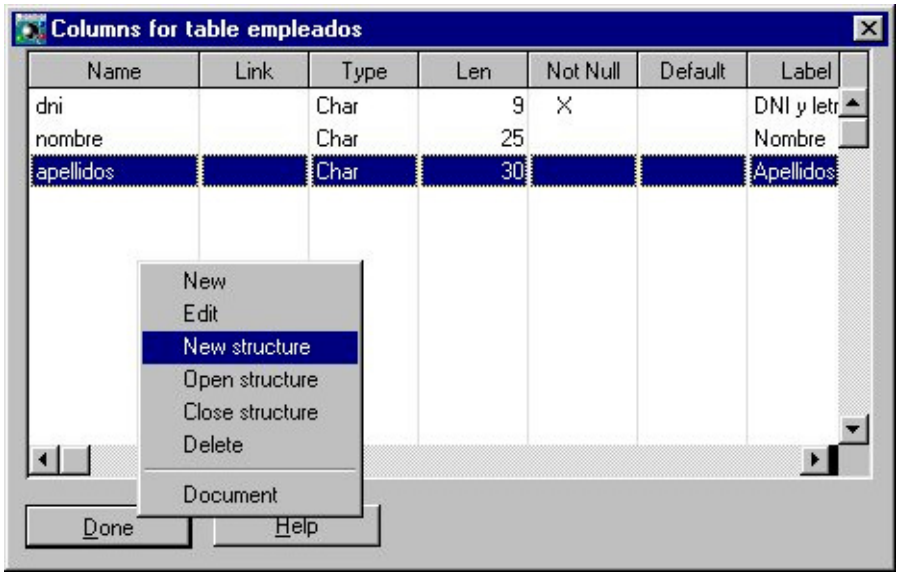


Figura 4.4. Crear una nueva estructura.

Se muestra el cuadro de diálogo donde se introduce el nombre de la nueva estructura y su etiqueta, pulsando el botón Ok se muestra en el cuadro de diálogo de las columnas de la tabla de empleados la nueva estructura. (Figura 4.5)

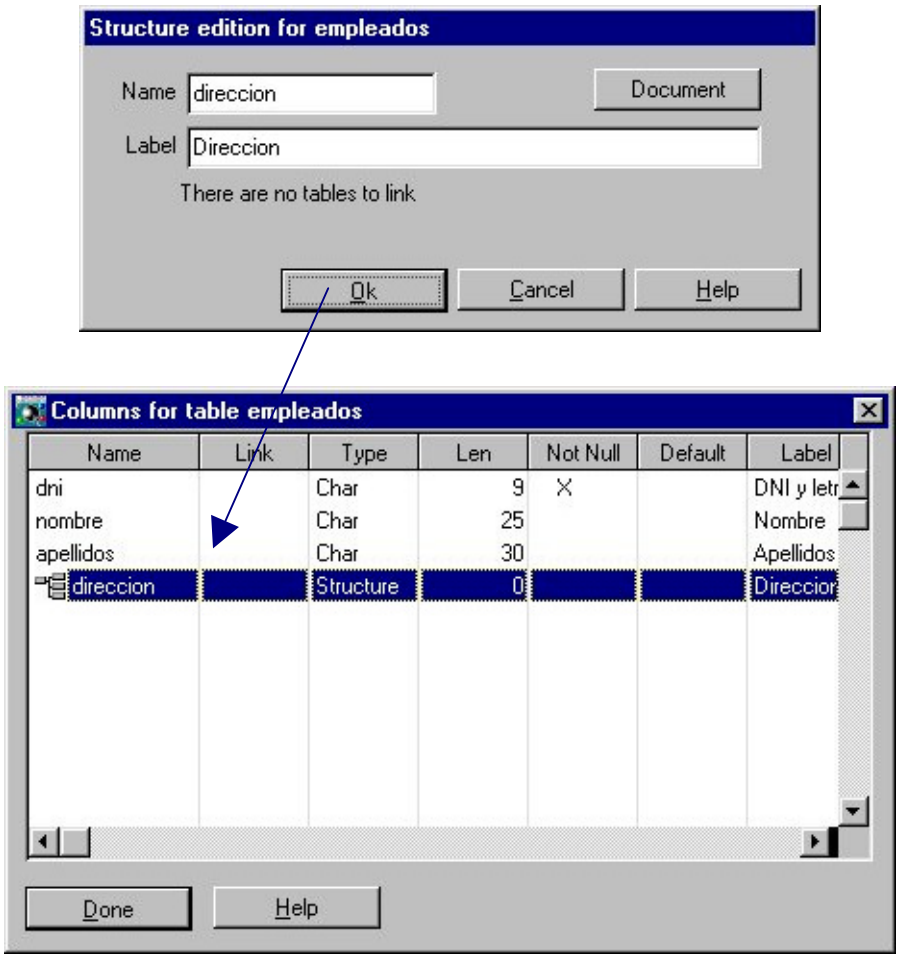


Figura 4.5. Añadir la nueva estructura.

Si hacemos doble click de ratón sobre “dirección” o seleccionamos el opción ***Open Structure*** que se muestra cuando pulsamos el botón secundario del ratón sobre “dirección” se muestra el cuadro de ***Columns for structure direccion***. Para añadir columnas en la estructura “direccion” se realizan los mismos pasos que cuando se añaden columnas a la tabla.Las columnas que se han añadido a la estructura se pueden ver en la figura 4.6. con sus características.

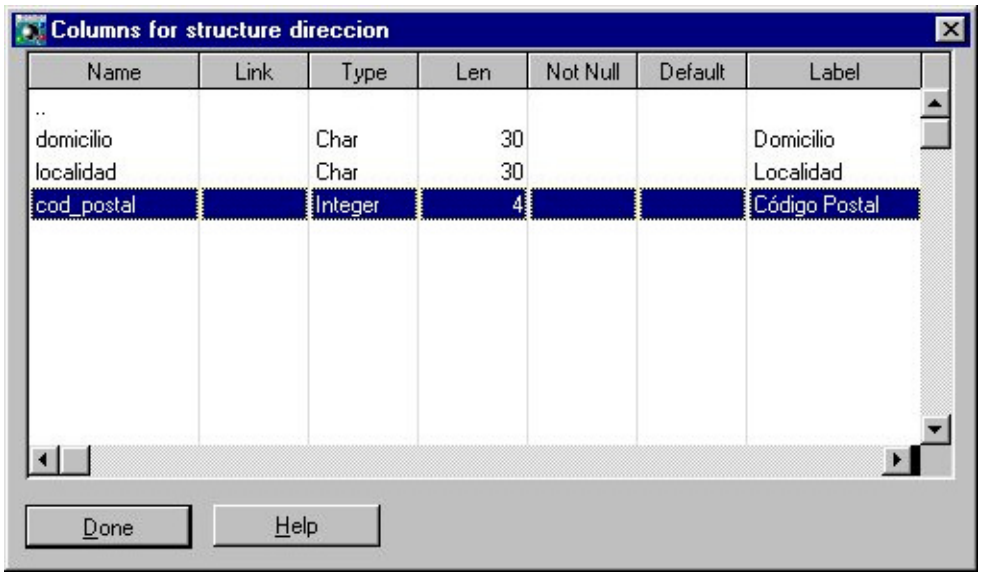


Figura 4.6. Columnas de la estructura dirección.

Después de pulsar el botón 

terminamos de rellenar las columnas de la tabla “empleados” con las columnas teléfono y código de departamento. La tabla “empleados” finalmente tendrá las columnas que se muestran en la figura 4.7.

Figura 4.7. Columnas de la tabla “empleados”.

Pulsamos el botón 

para realizar efectivamente la definición de las columnas de la tabla “empleados”. La definición de claves la realizaremos cuando tengamos definidas el resto de las tablas.

Departamentos

La tabla de departamentos tendrá las columnas que se muestran en la figura 4.8.

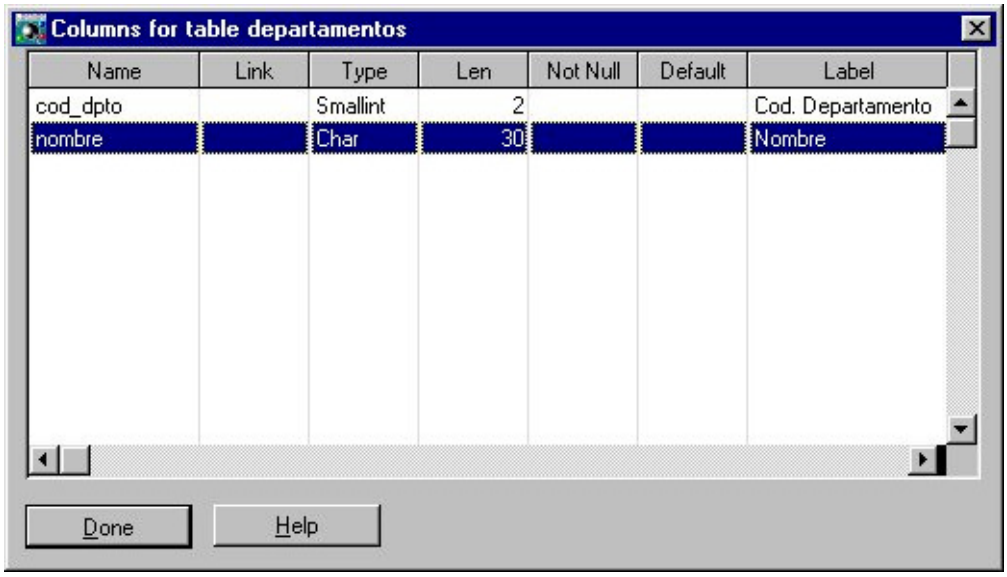


Figura 4.8. Columnas de la tabla departamentos.

Como se ha terminado de introducir las columnas de las dos tablas que forman el repositorio seleccionamos del menú ***File*** la opción ***Save*** y se guarda el repositorio con el nombre de "Repositorio2". En este momento el aspecto de la estructura en forma de árbol del repositorio es la que se muestra en la figura 4.9.

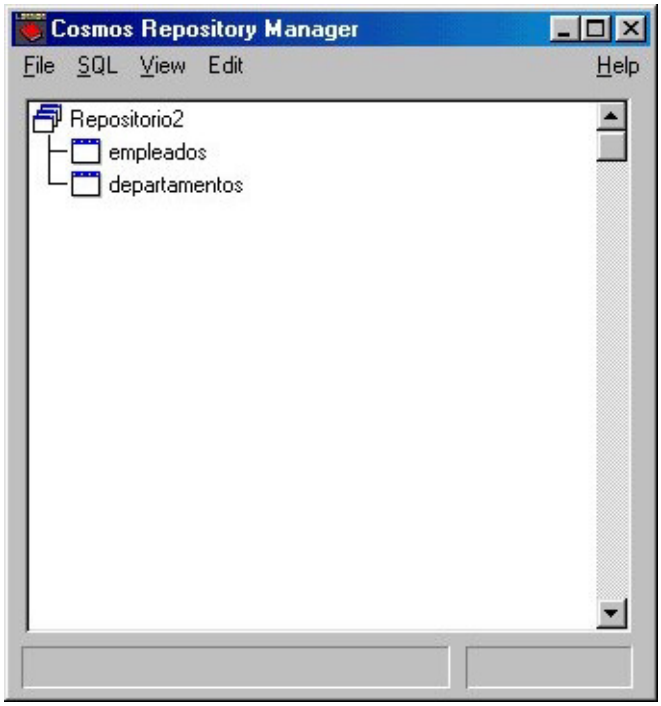


Figura 4.9. Contenido del repositorio.

5. Definición de las claves primarias.

En este apartado vamos a realizar la definición de las claves primarias de las dos tablas. Una clave primaria es un conjunto de columnas, ninguna de las cuales ha de admitir valores nulos (NOT NULL), que va a identificar de forma unívoca cada una de las filas de la tabla. Una tabla admite únicamente una clave primaria.

Empleados

Seleccionamos la tabla y pulsamos el botón secundario del ratón seleccionando ***Primary Keys***, se nos muestra el cuadro de diálogo donde podemos seleccionar las columnas que formaran parte de la clave primaria de la tabla. (Figura 4.10).

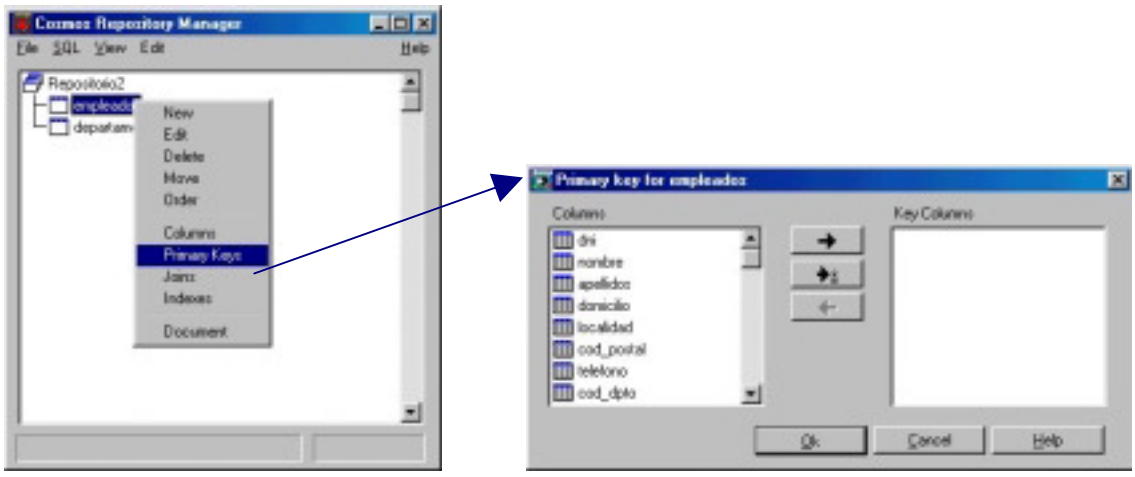


Figura 4.10. Seleccionar la clave primaria para la tabla empleados.

Seleccionamos la columna dni y pulsamos el botón  para añadir esta columna como la única que formara parte de la clave primaria de la tabla.(Figura 4.11)

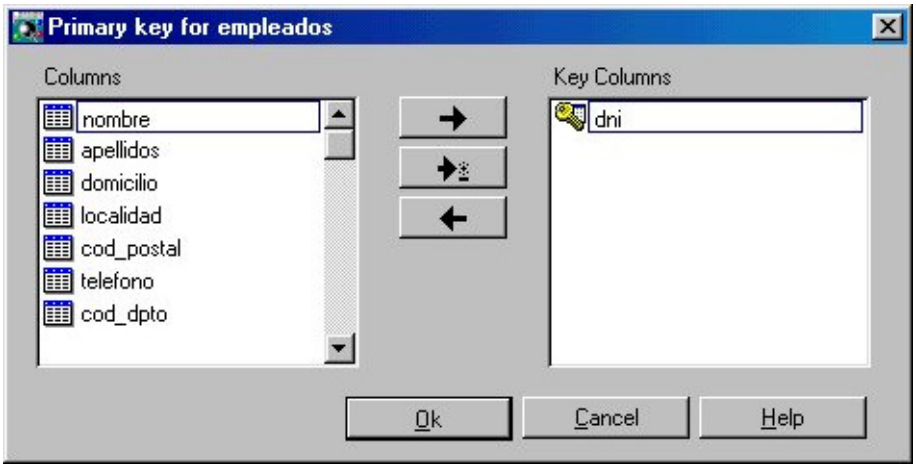


Figura 4.11. Clave primaria de la tabla empleados.

Para terminar pulsamos el botón  . La clave primaria se refleja en las columnas de la tabla empleados mediante el símbolo  sobre el nombre de la tabla, como podemos ver en la figura 4.12. Recuérdese que el campo con clave primaria no puede dejarse en blanco en la introducción de datos en la tabla.

Figura 4.12. Aspecto de las columnas de la tabla empleados con su clave primaria.

Departamentos

En esta tabla se selecciona como clave primaria la columna de cod_dpto como se muestra en la figura 4.13.

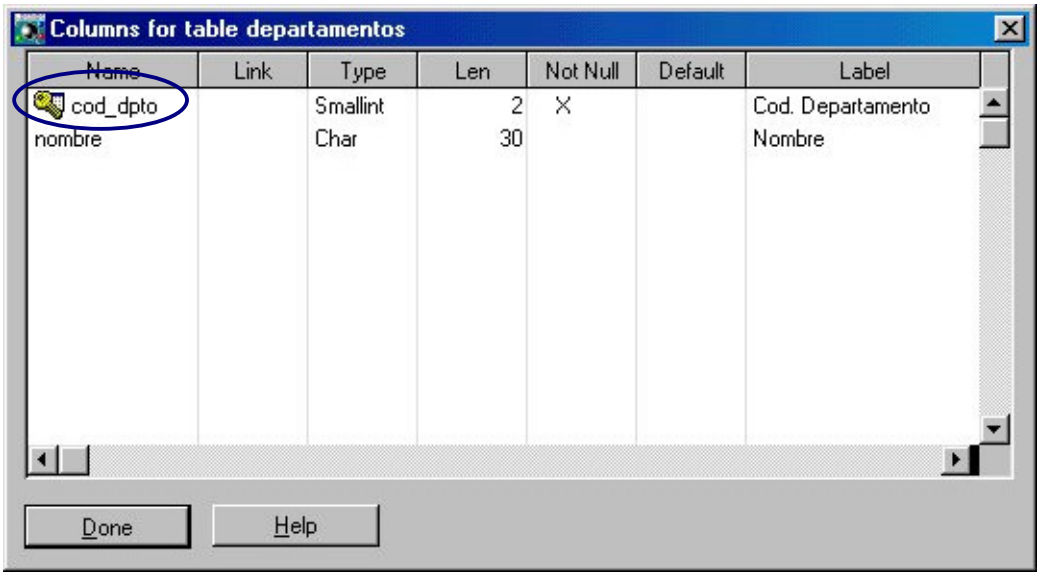


Figura 4.13. Aspecto de las columnas de la tabla departamentos con su clave primaria.

6. Definición de claves referenciales/Joins.

Una clave referencial establece la relación entre la tabla para la que se crea y la tabla que referencia. Mediante esta relación el usuario nunca podrá actualizar la columna o columnas que componen la clave primaria en la tabla referenciada o borrar la fila si existe información de ella en la tabla referencial

El concepto de join es equivalente al de clave referencial, la única diferencia es que un join solo existirá en el repositorio, no existirá en la base de datos. Las claves referenciales son joins que existen en el repositorio y en la base de datos. Todas las claves referenciales son joins pero no todos los joins son claves referenciales.

Se va a definir un join entre la tabla empleados y departamentos. La relación entre ambas tablas se establece a través de la columna cod_dpto de forma que en la tabla empleados definimos un join que relaciona esa columna con la del mismo nombre y tipo de la tabla departamentos.

Se selecciona en la paleta del repositorio la tabla empleados y pulsamos el botón secundario del ratón, seleccionando la opción Joins del menú que se muestra. (Figura 4.14)

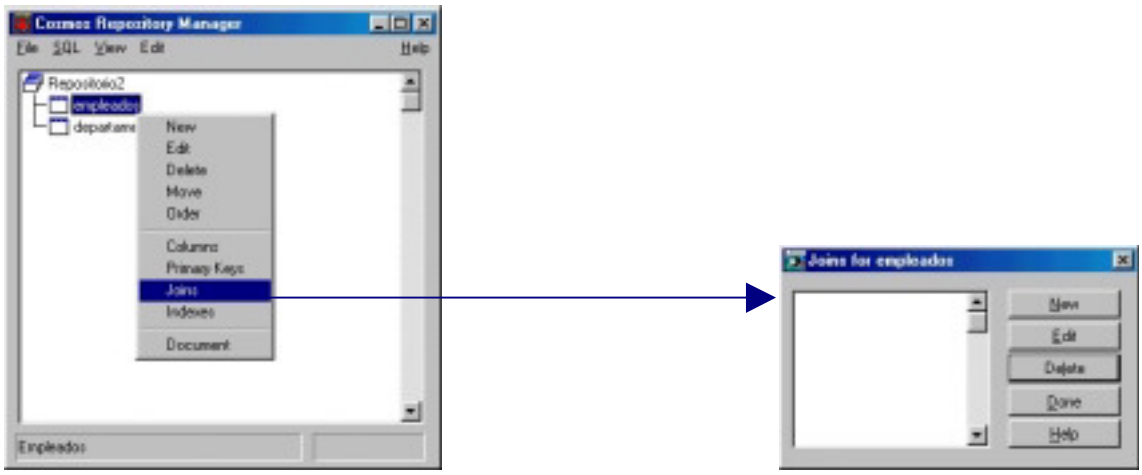


Figura 4.14. Añadir un join a la tabla empleados.

Pulsamos el botón  para añadir el nuevo join. Se muestra el cuadro de diálogo donde se define el nombre y las características de este join. (Figura 4.15)

Figura 4.15. Características del join para la tabla empleados.

Se debe introducir el nombre y seleccionar de la lista desplegable etiquetada como ***Table Referenced*** la tabla a que se referencia mediante este join, y asociar la columna de la tabla de referencia con la columna de la tabla referida.

Pulsamos el botón Ok y en el cuadro de diálogo denominado ***Joins for empleados*** aparece el nuevo join definido. Pulsando el botón Done en este cuadro se realiza la definición efectivamente. (Figura 4.16)

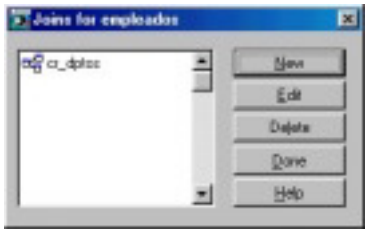


Figura 4.16. Nuevo join.

Si se selecciona ahora la tabla de empleados y se hace doble click de ratón sobre ella se muestra el cuadro de diálogo de Columns for table empleados y el nuevo join queda visible por el icono situado inmediatamente antes del nombre de la columna cod_dpto. (Figura 4.17)

Figura 4.17. Icono de join sobre la columna cod_dpto.

En el apartado 10 de este manual se verá la ejecución de los maestro-detalle.

7. Crear la base de datos.

En este momento se encuentra definido completamente el repositorio y podemos crear la base de datos correspondiente. Se selecciona dentro del menú ***SQL*** la opción ***Database Options*** para seleccionar la conexión definida mediante el Editor de Configuración que se desea utilizar. Se muestra un cuadro de diálogo donde, mediante una lista desplegable podemos seleccionar el tipo de conexión. (Figura 4.18)

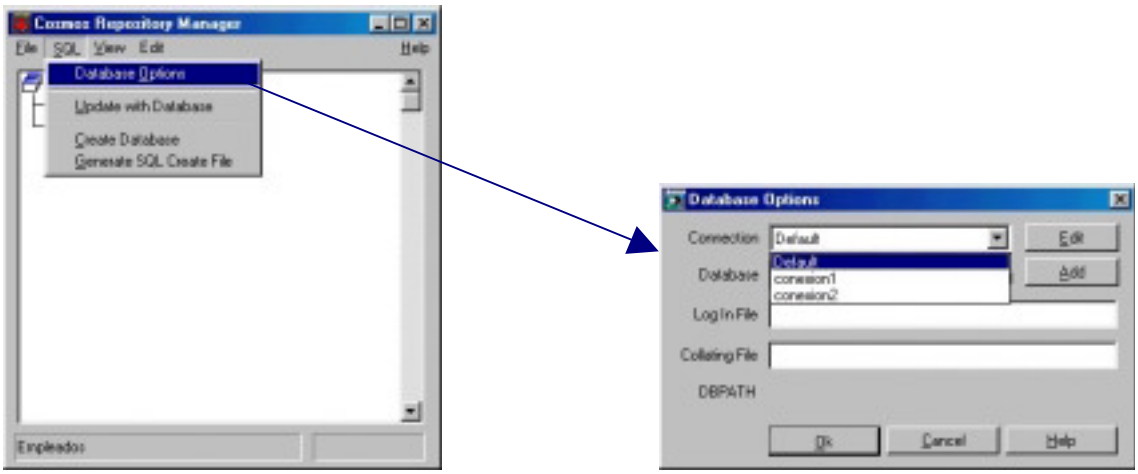


Figura 4.18. Definir las opciones de la base de datos.

Se selecciona “conexion2” que tiene asociada un nombre de base de datos, “datos2”, y un path para la misma.

Del mismo menú se selecciona la opción ***Create Database*** para que se cree efectivamente la base de datos con el repositorio. Se muestra nuevamente el cuadro de diálogo de ***Database Options*** y después de pulsar el botón Ok, se crea la base de datos y se muestra el cuadro de diálogo de la figura 4.19.

Figura 4.19. Se ha realizado la creación de la base de datos.

8. Crear el proyecto.

En este momento se tiene definido el repositorio y la base de datos de las dos tablas que van a componer la aplicación. Se selecciona dentro del Editor Visual de Cosmos la opción New, se muestra el cuadro de diálogo de la figura 4.20 donde se selecciona crear un nuevo documento de proyecto.

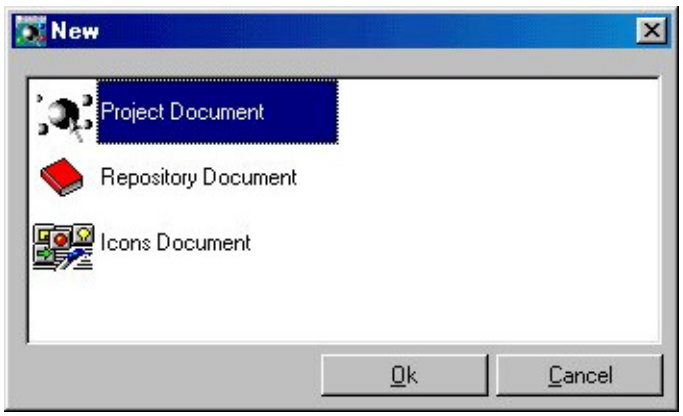


Figura 4.20. Crear un nuevo proyecto.

Se muestra el cuadro de diálogo de la figura 4.21 donde se define el nombre y ubicación del nuevo proyecto, si pulsamos el botón ... se muestra una ventana para poder seleccionar el lugar donde se desea guardar el nuevo proyecto.

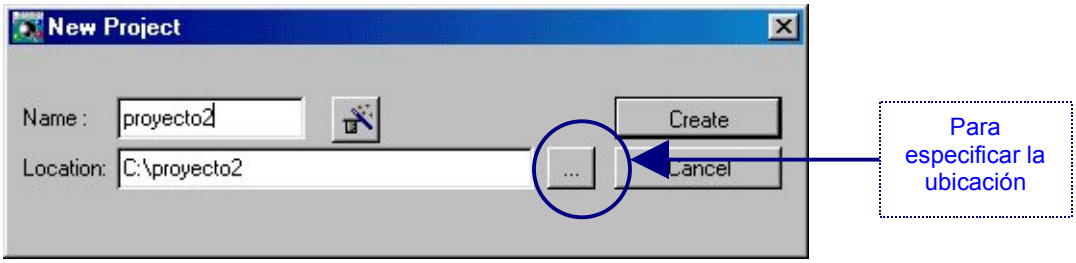


Figura 4.21. Definir nombre y ubicación del nuevo proyecto.

Se va a hacer uso de otro tipo de asistente que nos proporciona Cosmos, el Wizard de Proyectos. En el capítulo primero de este manual se explicaron los diferentes tipos de asistentes que existían en Cosmos. Se pulsa el botón para comenzar la ejecución del Wizard de proyecto.

La primera ventana que se muestra es la selección del repositorio que queremos asociar al proyecto, pulsando el botón Browse se muestra una ventana donde podemos seleccionar el repositorio como se ve en la Figura 4.22. Pulsamos el botón Next >> para continuar con el Wizard.

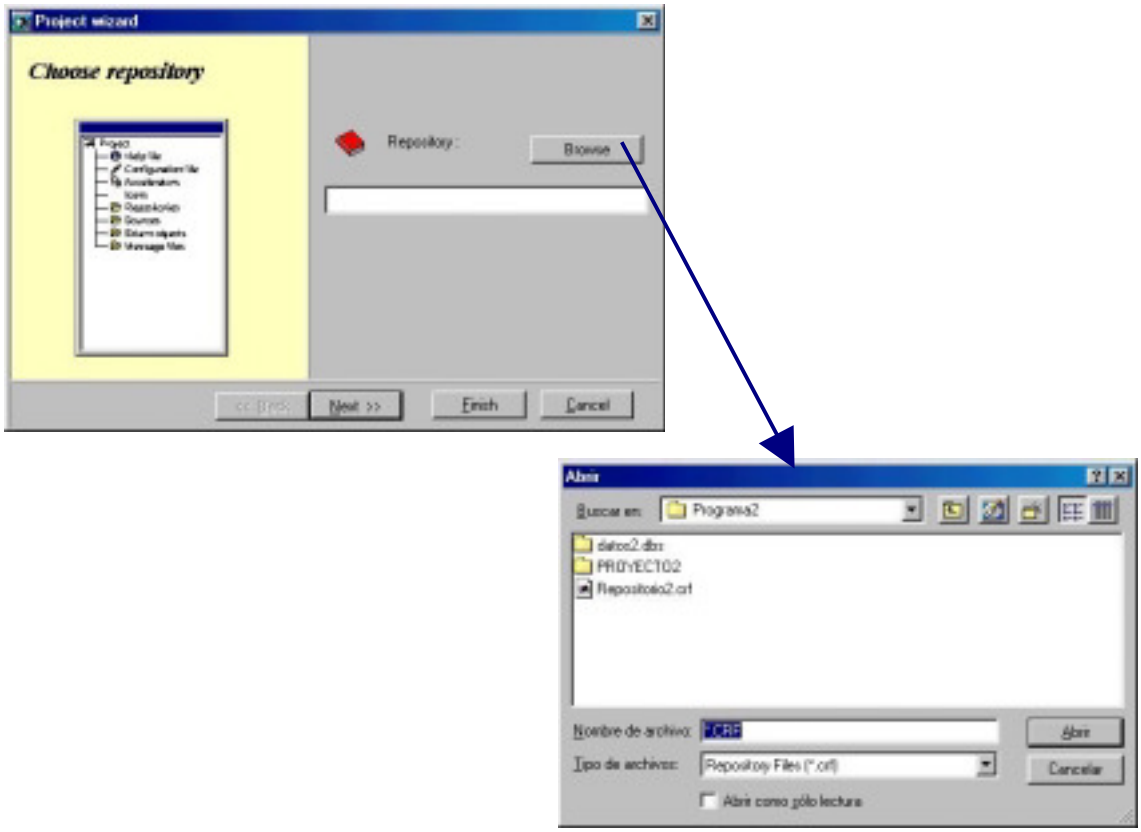


Figura 4.22. Selección del repositorio.

El siguiente paso del Wizard es seleccionar los includes que formarán parte de la aplicación. Un include exporta su declaración de constantes, su definición de clases públicas, objetos públicos y métodos públicos. Sus métodos y clases privados y protegidos, no se exportan y por tanto no pueden ser utilizados en el programa que lo incluya. En nuestra aplicación no añadimos ningún include, pulsamos el botón 

. (Figura 4.23)

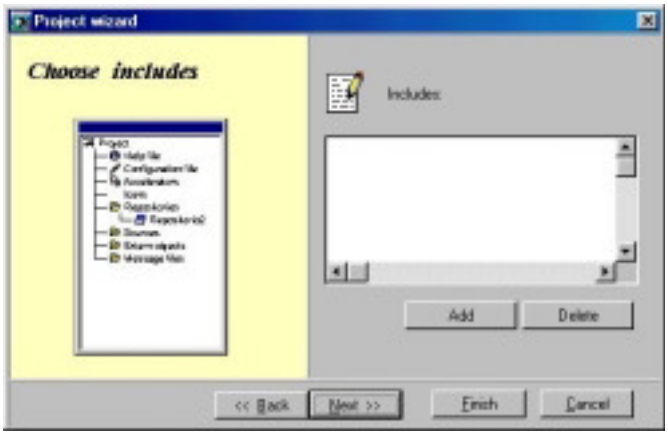


Figura 4.23. Seleccionar los includes.

En el siguiente paso del Wizard se muestra el tipo de módulos que se realizarán en el proyecto por defecto, se pueden modificar seleccionando de los menús desplegables el tipo que mejor se adapte a las necesidades de la aplicación. (Figura 4.24)

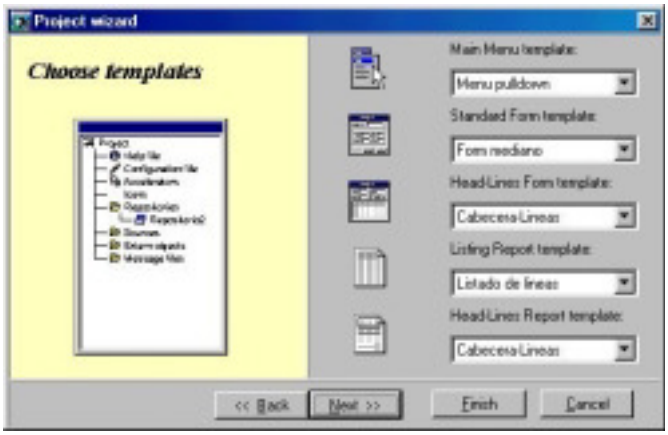


Figura 4.24. Seleccionar el tipo de módulos.

Se mantiene los tipos que se muestran por defecto. En los siguientes apartados se verá el aspecto y contenido de cada uno de los módulos.

Después de pulsar el botón 

se muestra el cuadro de diálogo que se muestra en la figura 4.25. donde se selecciona la conexión y la base de datos que se va a asociar al proyecto.

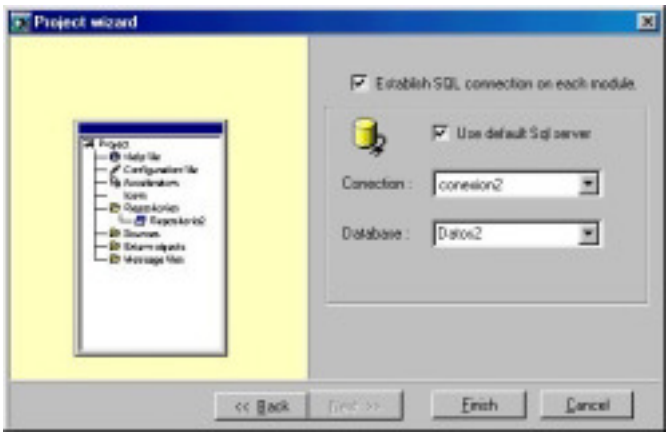


Figura 4.25. Seleccionar la conexión y base de datos del proyecto.

Pulsando  termina la ejecución del Wizard de módulos. El aspecto que presenta la paleta del proyecto que se construye es el que se muestra en la figura 4.26. Del mismo modo que en el capítulo dos de este manual para realizar el informe de la tabla "personal" el Wizard de módulos añadió el include "templinc2", en este proyecto también se ha realizado la misma inclusión. En la sección de anexos se incluye el código completo de este include que tiene definida toda la funcionalidad necesaria para realizar informes.

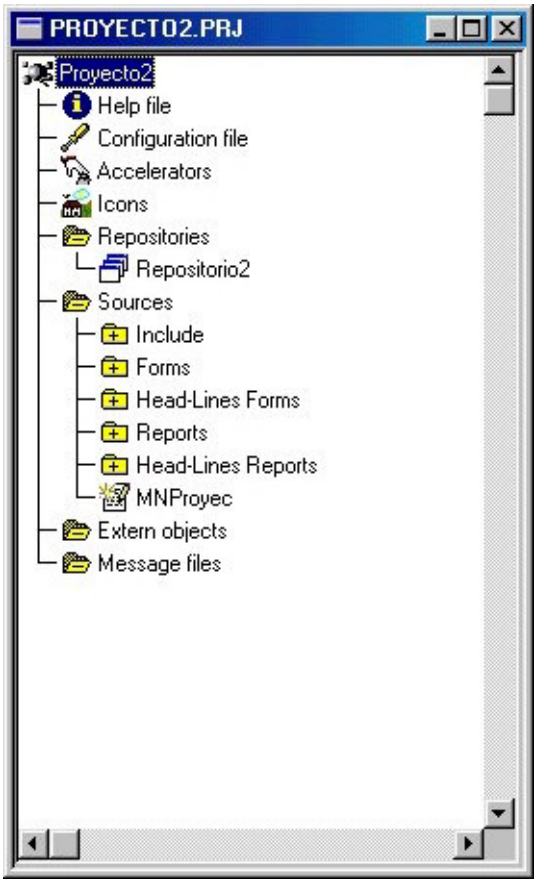


Figura 4.26. Paleta del proyecto creado mediante por Wizard.

El módulo MNProyec esta marcado mediante un icono especial que indica que se trata del módulo principal de inicio de la aplicación. Si seleccionamos este módulo y pulsamos el botón secundario del ratón seleccionando la opción Properties se muestra el cuadro de diálogo donde se encuentra seleccionada la casilla de verificación que indica que se tratará de un Main module, en un proyecto únicamente puede existir un módulo principal. (Figura 4.27).

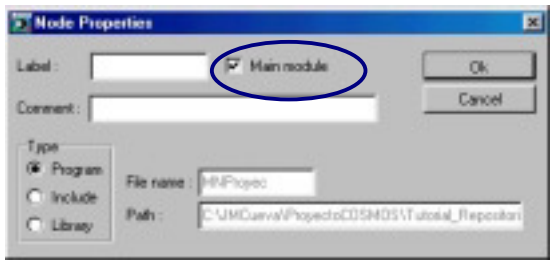


Figura 4.27. Propiedades del módulo.

En este cuadro de diálogo se puede añadir una Label al módulo y un comentario, si se escribe como label “inicio” se muestra esta label en la paleta del proyecto. (Figura 4.28)

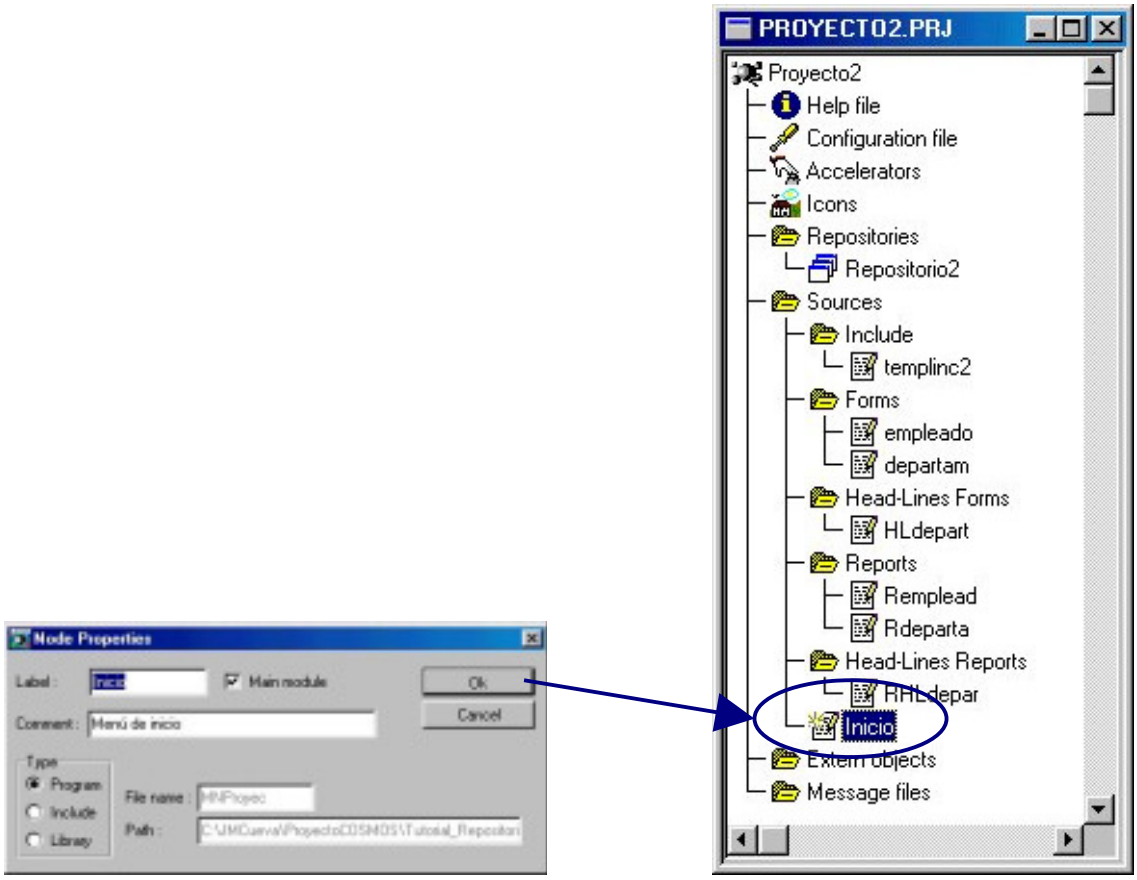


Figura 4.28. Añadir una Label al módulo MNProyec.

Se puede observar que en la figura 4.26 había una serie de carpetas que se han abierto para mostrar su contenido. De la misma forma que se ha añadido una Label al módulo MNProyec, se puede añadir al resto de los módulos.

9. Compilación del proyecto.

Un paso importante a realizar en este momento es seleccionar la opción ***Build all*** del menú ***Tools.*** Mediante esta opción se compila cada uno de los módulos que componen el proyecto para poder ejecutar como se puede ver en la figura 4.29.

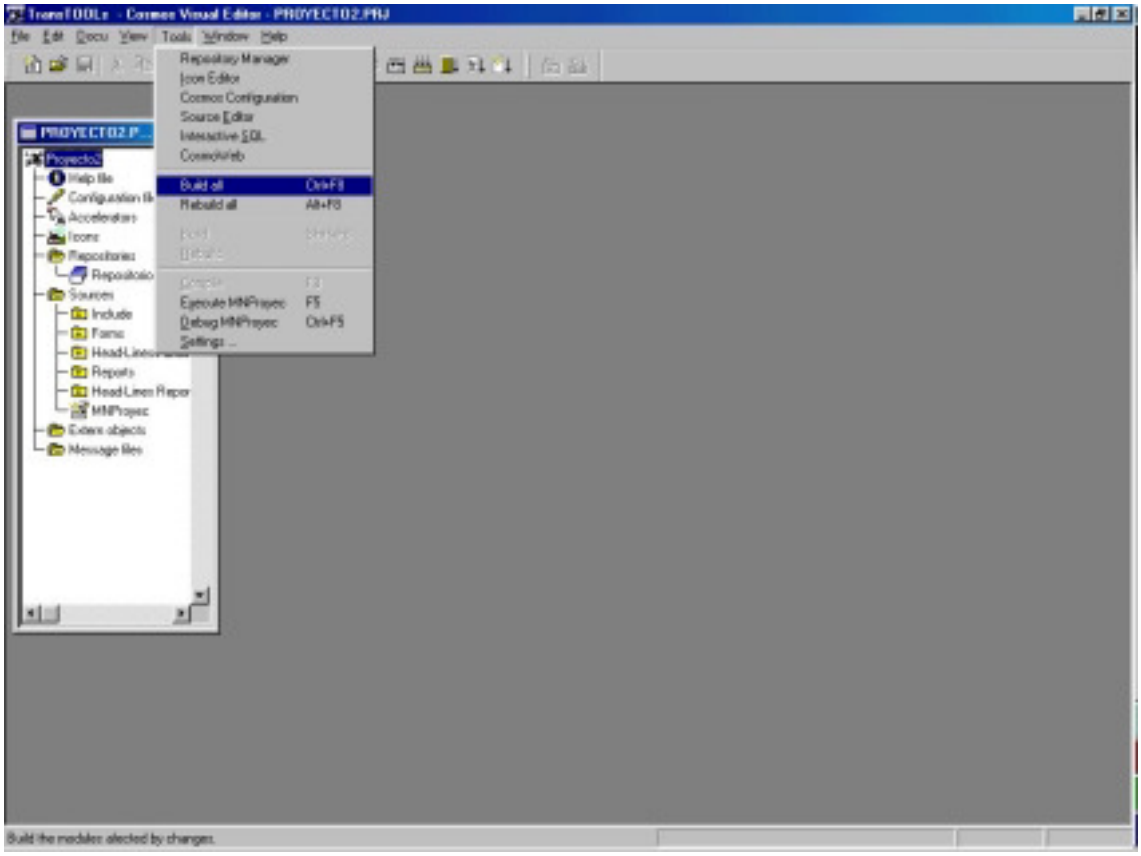
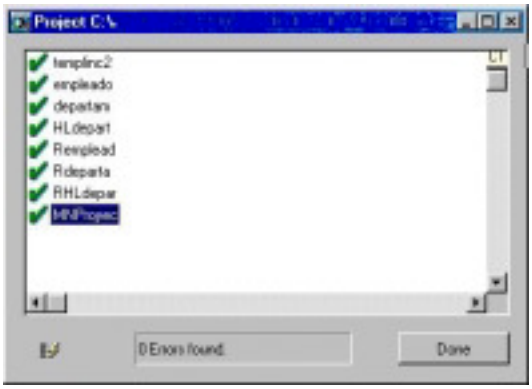


Figura 4.29. Seleccionar la opcion Build all.

La pantalla que se muestra durante la compilación del proyecto es la de la figura 4.30 donde se especifican cada uno de los módulos que componen el proyecto y el número de errores, si existen, detectados durante la compilación.



4.30. Evolución de la compilación del proyecto.

En este momento ya se puede ejecutar el proyecto.

10. Ejecución del proyecto.

Al existir un módulo principal en el proyecto al pulsar el icono de ejecución se ejecuta dicho módulo, siempre que no tengamos seleccionado un módulo en concreto dentro de la paleta del proyecto.

Cuando pulsamos el icono  se muestra el aspecto de la screen del módulo de inicio, como se puede ver en la figura 4.31.

Figura 4.31. Pantalla principal de la aplicación.

Esta screen esta compuesta de un menú y un control de tipo bitmap para mostrar el logotipo de Cosmos. Mediante el menú se puede acceder al resto de los módulos del proyecto. El contenido de las diferentes opciones del menú desplegable y su función, son las siguientes:

Opción	Función
Forms	Contiene las dos pantallas necesarias para realizar el mantenimiento de las dos tablas del repositorio. Tendrán el mismo aspecto y funcionalidad que la realizada en el capítulo primero para el mantenimiento de la tabla personal.
Head-Lines Forms	Mantenimiento de las dos tablas simultáneamente mediante una screen donde figuran en campos edit las columnas de la tabla departamentos y en un control de tipo Grid las columnas de la tabla empleados. Veremos mas adelante el aspecto de este nuevo tipo de screen.
Reports	Informes de las dos tablas que tenemos en el repositorio, tienen el mismo aspecto y funcionalidad que el descrito en el capítulo segundo para la tabla personal.
Head-Lines Reports	Informe de cabeceras-detalle para las dos tablas de la aplicación.

11. Pantalla e informe de cabecera-líneas.

En este apartado vamos a ver el aspecto de la form y el informe de cabeceras-líneas que ha generado el Wizard para el mantenimiento conjunto de las dos tablas que contiene el repositorio. Estas dos tablas tienen una relación 1 a n como vimos en los apartados 2 y 3 de este capítulo. Esto quiere decir que en cada departamento existen varios empleados y que cada empleado únicamente pertenece a un departamento concreto.

Módulo HLdepart

En la carpeta Head-Lines Forms se encuentra el módulo Hldepart que contiene toda la funcionalidad del mantenimiento maestro-detalle de las tablas de repositorio. Se ha definido una clase llamada **CHLdepart** en esta clase la sección screen tiene el aspecto que se muestra en la figura 4.32.

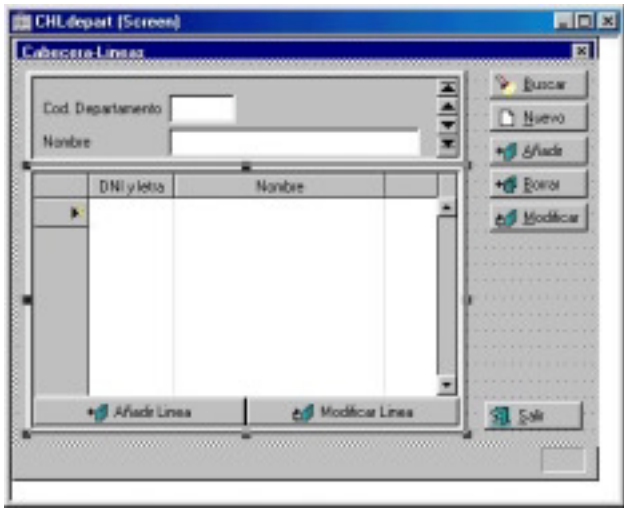


Figura 4.32. Screen del módulo Head

En la parte superior de la form se encuentran los campos de la tabla departamentos, el contenido de la tabla empleados se encuentra definida mediante un control de tipo grid. En la parte derecha de la screen aparecen los botones necesarios para el mantenimiento de la tablas. Estos botones son similares a los que se explicaron en el capítulo primero. Bajo el control de tipo grid existen dos botones que permiten añadir una línea a la tabla empleados y modificar una línea ya existente. Si se extiende el tamaño de esta screen aparece un nuevo botón relacionado con la tabla empleados: borrar línea. (Figura 4.33).

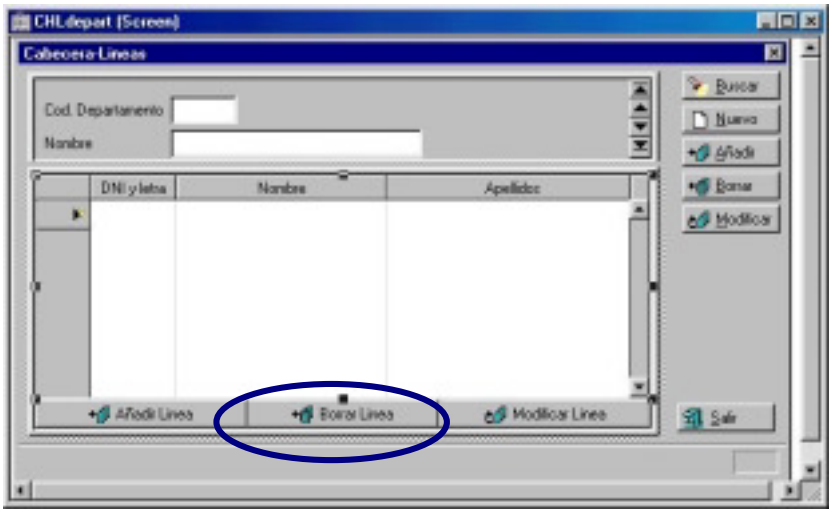


Figura 4.33. Ampliación del tamaño de la screen.

En la figura 4.34. se muestran la pestaña **Special** de las propiedades del botón Borrar Línea, para editar estas propiedades se hace doble click de ratón sobre el botón

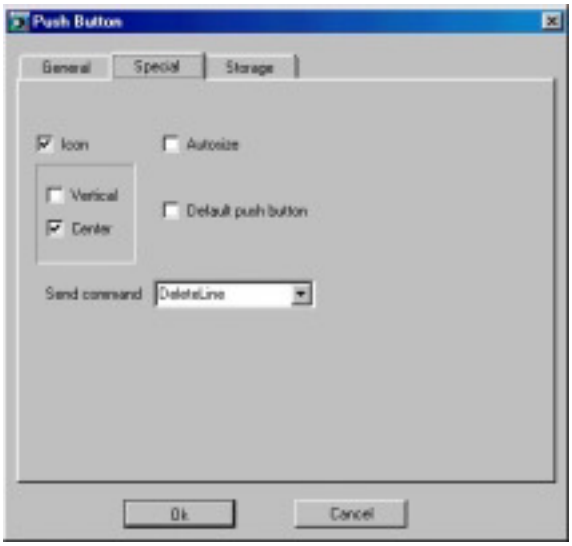


Figura 4.34. Pestaña Special del botón Borrar línea.

Como se puede ver en la opción **Send Command** se encuentra seleccionado el comando **DeleteLine**. Este comando se encuentra en la lista desplegable porque dentro de la sección **Code** de la clase "CHLdepart" se han definido una serie de comandos. Si dentro de la paleta del módulo "Hldepart" se selecciona la sección de código de la clase definida se muestra los comandos que el Wizard ha generado para esa clase. (Figura 4.35).

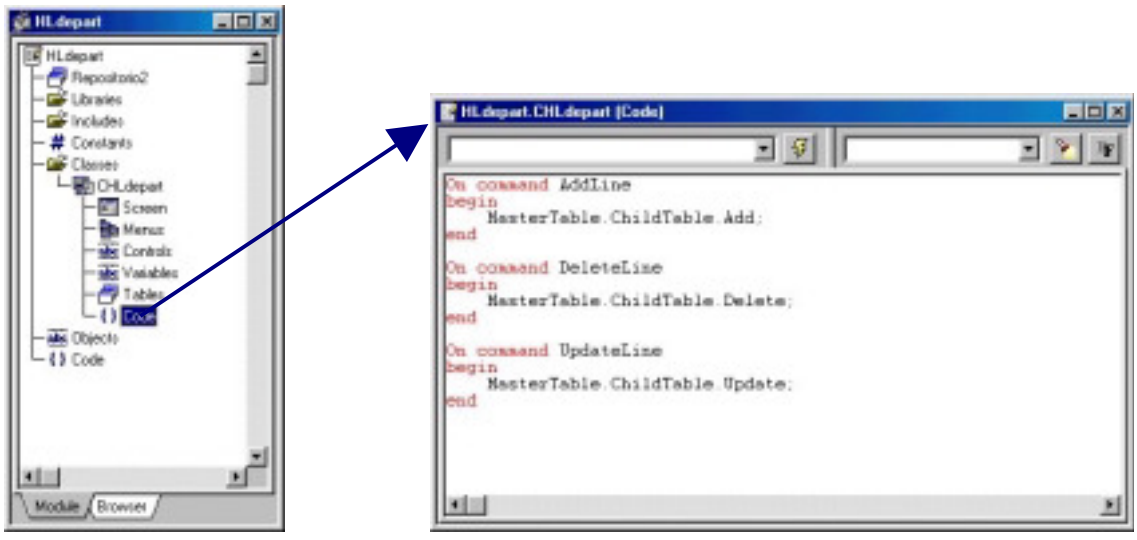


Figura 4.35. Código de la clase CHLdepart.

Mediante estos comandos se añaden, modifican y borran datos a la tabla de empleados que depende del departamento que se encuentre en cada momento activo. En cada uno de ellos se ejecuta unos métodos que se definen a continuación:

MasterTable: Este método de la clase Form retorna la tabla maestra del Form actual.

ChildTable: Es un método de la clase FormTable. Esta clase deriva de la clase Complex. Esta clase virtual permite manipular los datos de las tablas del Form actual. Una tabla del Form suele estar asociada con una tabla de la base de datos, por tanto, esta clase encapsula la funcionalidad necesaria para agregar, modificar, hacer consultas a dichas tablas, etc. Este método retorna la primera tabla de líneas de la tabla sobre la que se aplica el método, es decir se muestra la primera tabla de líneas de la tabla departamento que es sobre la que se aplica el método.

Se puede modificar el aspecto de esta screen para que se adapte mejor a las necesidades. Por ejemplo si se ejecuta el módulo, los campos de la tabla personal que aparecen en el control grid son unicamente DNI y Nombre. Para poder ver el resto de las columnas de esta tabla se va a añadir la barra de desplazamiento horizontal al grid. Se selecciona el control grid y pulsando el botón secundario del ratón y seleccionando **Properties** se muestra el cuadro de diálogo donde podemos seleccionar las diferentes características de este control, como vimos en capítulo tercero. Seleccionamos la pestaña Special y marcamos la casilla de selección Horizontal scroll (Figura 4.36)

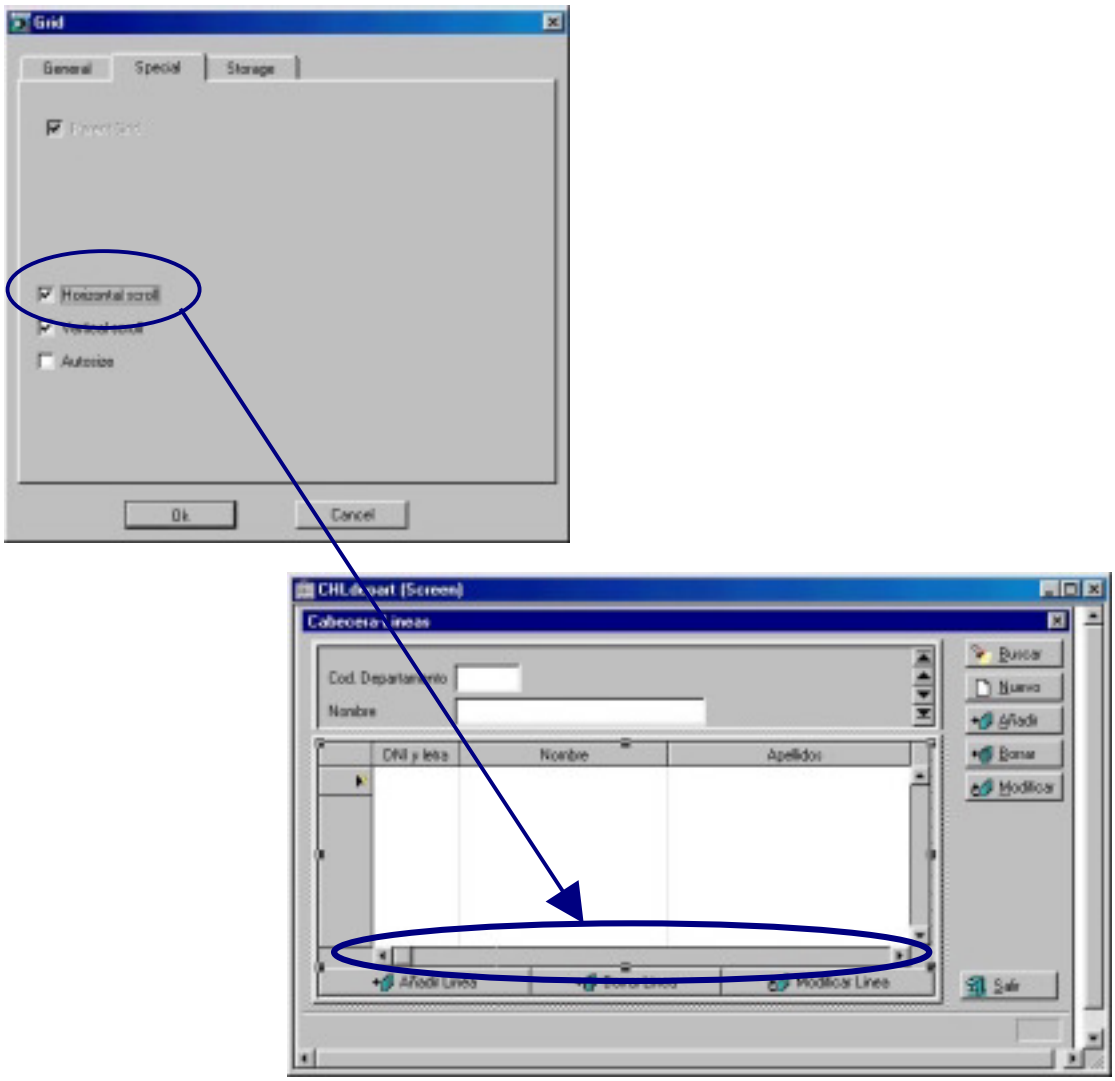


Figura 4.36. Añadir la barra de desplazamiento horizontal al grid.

También en la sección Tables de esta clase se puede ver la forma en la que están relacionadas las tablas del repositorio. En la figura 4.37 se puede observar a la izquierda las propiedades de la tabla departamentos en la pestaña Table que relaciona la tabla del proyecto con la tabla del repositorio, y a la derecha se puede ver la pestaña Depend de las propiedades de la tabla empleados.

Tabla Departamentos

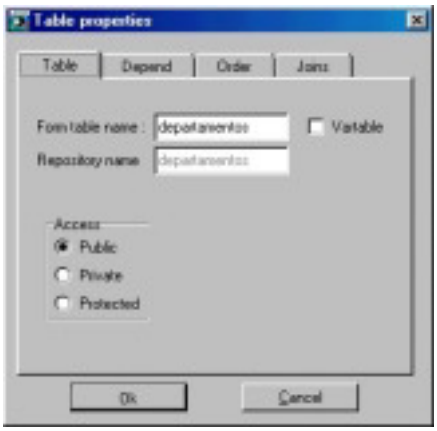


Tabla Empleados

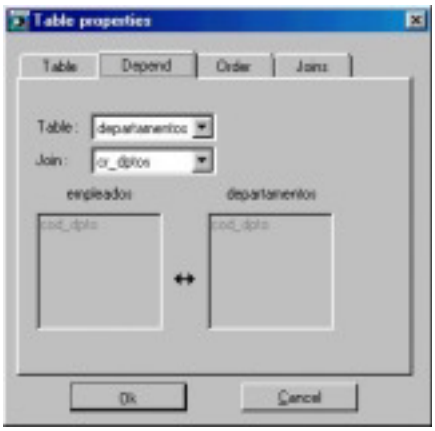


Figura 4.37. Propiedades de las tablas del proyecto.

Módulo RHLdepart

En este módulo se realiza el informe del tipo maestro-detalle de la tabla departamento y su tabla asociada empleados. Dentro de la paleta del módulo se puede observar la clase generada por el Wizard: CRHLdepart. Esta clase esta derivada de la clase Page de Cosmos. (Figura 4.38)

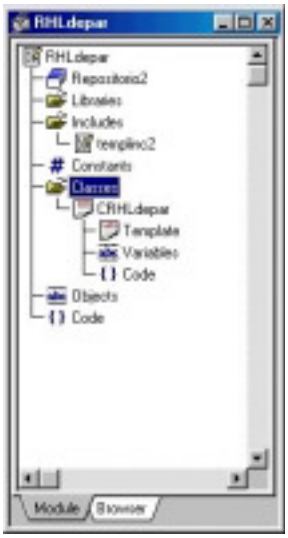


Figura 4.38. Paleta del módulo RHLdepar.

El Wizard ha añadido el include **templic2** para hacer uso de la utilidad definida en él de la misma forma que se explicó en el capítulo segundo de este manual.

El aspecto de la sección Template de esta nueva clase definida se muestra en la figura 4.39. Podemos modificar su aspecto para que se adapte mejor a las necesidades de la aplicación actual.

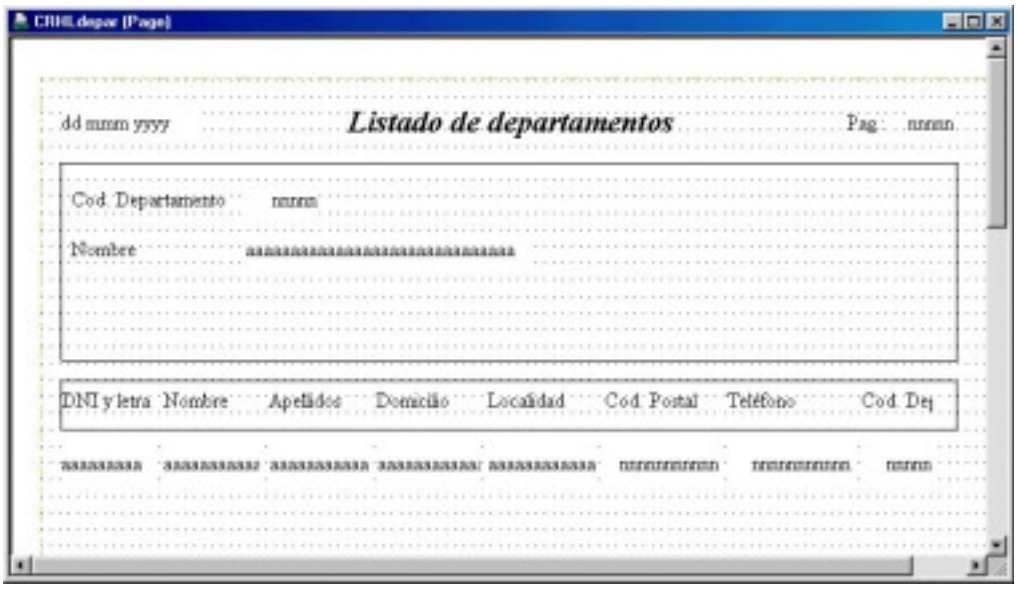


Figura 4.39. Aspecto del informe de los departamentos.

12. Personalización del menú.

En esta sección se va a realizar la personalización del menú que ha generado el Wizard con objeto de adaptar mejor las diferentes opciones de nuestra aplicación.

Seleccionamos de la paleta de la aplicación el módulo denominado Inicio y haciendo doble click de ratón sobre él abrimos su paleta. (Figura 4.40)

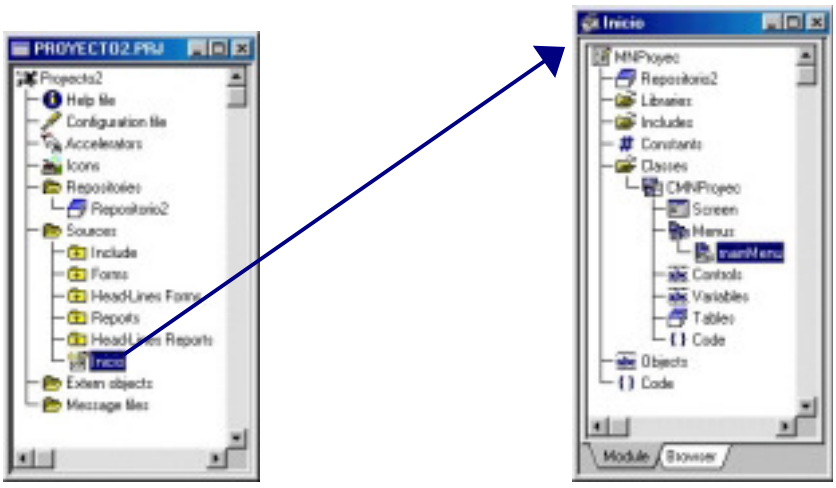


Figura 4.40. Se abre la paleta del módulo Inicio.

Se encuentra en este módulo una clase denominada **CMNProyec** y dentro de la sección **Menus** de esta clase esta definido un menú llamado **mainMenu**. Este menú deriva de la clase Menu de Cosmos. Si se hace doble click sobre este, se muestra el contenido de las diferentes opciones del menú principal, donde se van a realizar las modificaciones. (Figura 4.41)

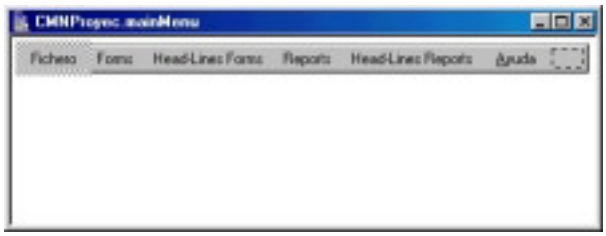


Figura 4.41. Contenido del menú.

Se selecciona el componente Forms y se hace doble click de ratón sobre él y se muestra las propiedades del mismo. Dentro de el campo de edición marcado como Caption podemos modificar el nombre que aparece para este componente en el menú. Se cambia por Mantenimiento. (Figura 4.42)

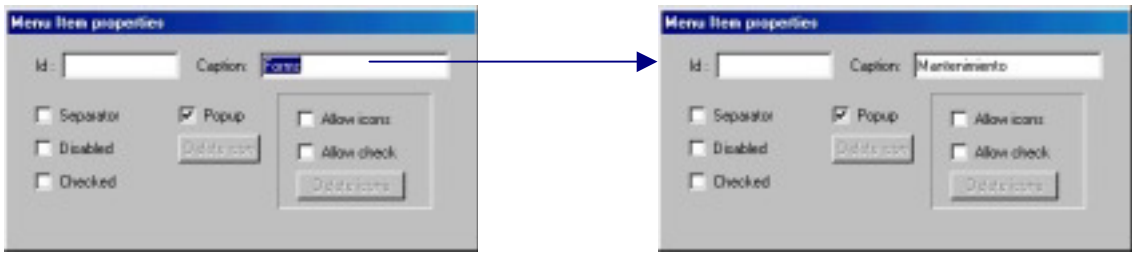


Figura 4.42. Modificación del nombre.

El resultado de esta modificación se muestra en la figura 4.43. donde también se muestra en contenido del menú desplegable, que también se modificará de la misma forma, para completar el nombre de la opción departamento.

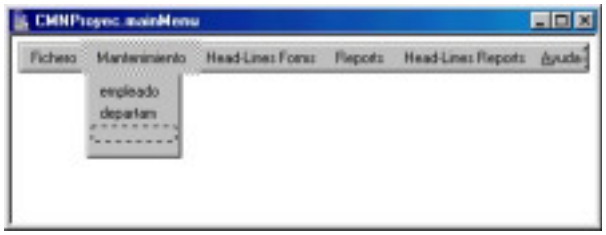


Figura 4.43. Resultado de la modificación.

Después de modificar casi todos los campos del menú de la misma forma que se ha realizado con la entrada Forms, el aspecto del menú es el que se muestra en la figura 4.44.

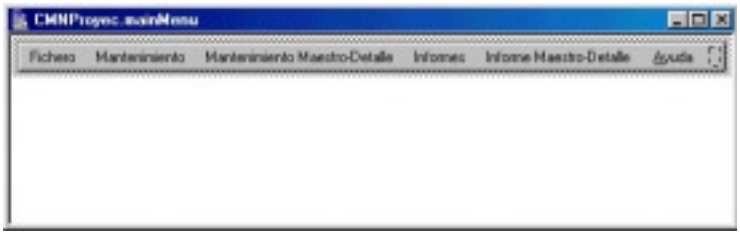


Figura 4.44. Aspecto del menú después de la modificación completa.

Si se ejecuta nuevamente la aplicación las modificaciones quedan reflejadas en el menú que se muestra junto al bitmap asociado. (Figura 4.45)

Figura 4.45. Ejecución final de la aplicación.

13. Código generado.

Del código que se ha generado para la aplicación completa, únicamente se va a mostrar el generado para los módulos analizados en los dos apartados anteriores, el de pantalla de maestro-detalle, el informe de maestro-detalle y el módulo inicio que contiene el menú. El resto de los módulos son similares a los analizados en los capítulos anteriores.

Código del módulo Inicio

```
REPOSITORY repositorio2
CONSTANTS  begin
end
CLASSES BEGIN
    CMNProyec is Form begin
        default menu mainMenu
        objects begin
        end
        menu mainMenu
        PULLDOWN
            BEGIN
                MENU
                    "Fichero"
                    ICONFILE "_MBSTDBMP_"
                    ALLOWICON
                    BEGIN
                        OPTION
                            "Salir"
                            ICON 4
                            COMMAND Close
                    END
                MENU
                    "Mantenimiento"
                    BEGIN
                        OPTION
                            "empleado"
                            COMMAND empleado
```

```
                OPTION
                    "departamento"
                    COMMAND departam
            END
        MENU
            "Mantenimiento Maestro-Detalle"
            BEGIN
                OPTION
                    "Cabecera-Líneas"
                    COMMAND HLdepart
            END
        MENU
            "Informes"
            BEGIN
                OPTION
                    "empleados"
                    COMMAND Remplead
                OPTION
                    "departamento"
                    COMMAND Rdeparta
            END
        MENU
            "Informe Maestro-Detalle"
            BEGIN
                OPTION
                    "Cabecera-Líneas"
                    COMMAND RHLdepar
            END
        MENU
            "&Ayuda"
            ICONFILE "_MBSTDBMP_"
            ALLOWICON
            BEGIN
                OPTION
                    "Ayuda de la aplicación"
                    ICON 32
                    COMMAND CHelp
                OPTION
                    "Acerca de ..."
                    ICON 58
                    COMMAND CAbout
            END
    END
INTERFACE
    POSITION 0 0 554 395
    LABEL "Menú principal de la aplicación"
    SYSMENU
BEGIN
    CONTROL AS BOX
        POSITION 0 0 546 351
        FOREGROUND RGB 0 0 0
        ATTACH ALL
        NOLABEL
        NOBORDER
    BEGIN
        CONTROL logo AS BITMAP
            POSITION 0 0 546 351
            ATTACH ALL
            BORDER DOUBLE FRAME
            BITMAP FILE "C:\cosmos\Etc\Logo1.bmp"
            STRETCH
```

```
                BEGIN
                END
            END
        END
    end
END
CODE CLASS CMNProyec BEGIN
//{{CODEBEGIN
On Command RHLdepar
begin
    module.Run( "RHLdepar" );
end
On Command Rdeparta
begin
    module.Run( "Rdeparta" );
end
On Command Remplead
begin
    module.Run( "Remplead" );
end
On Command HLdepart
begin
    module.Run( "HLdepart" );
end
On Command departam
begin
    module.Run( "departam" );
end
On Command empleado
begin
    module.Run( "empleado" );
end
on command CAbout
begin
    MessageBox( "Aplicacion " , "Acerca de ..." , 64);
end

On Open
begin
    logo.Bitmap=CosmosDir+ "\etc\logo1.bmp";
end
//{{CODEEND
END
CODE BEGIN
//{{CODEBEGIN
main
objects
begin
oCMNProyec as CMNProyec
end
begin
oCMNProyec.Run;
end
//{{CODEEND
END
```

Código del módulo HLdepar

```
REPOSITORY repositorio2
CONSTANTS  begin
end
CLASSES BEGIN
    CHLdepart is Form begin
        objects begin
        end
        Table departamentos is departamentos
        begin
            cod_dpto as column required
            nombre as column
        end
        Table empleados is empleados
            depend on departamentos join cr_dptos
        begin
            dni as column required
            nombre as column
            apellidos as column
            domicilio as column
            localidad as column
            cod_postal as column
            telefono as column
        end
        INTERFACE
            POSITION 2 1 605 347
            LABEL "Cabecera-Lineas"
            SYSMENU
        BEGIN
            CONTROL AS BOX
                POSITION 0 0 597 295
                FOREGROUND RGB 0 0 0
                ATTACH ALL
                NOLABEL
                NOBORDER
            BEGIN
                CONTROL IDMASTER AS BOX
                    POSITION 7 5 491 74
                    ATTACH ALL
                    NOLABEL
                    BORDER DOUBLE FRAME
                    TABLE departamentos
                BEGIN
                    CONTROL AS SPIN
                        POSITION 462 0 17 62
                        ATTACH TOP RIGHT BOTTOM
                        NOBORDER
                        VERTICAL
                        QUADRUPLE
                        COMMAND
                    CONTROL AS TEXT
                        POSITION 7 13 92 22
                        NOBORDER
                        DATATYPE CHAR
                        TAGS "Cod. Departamento"
                    CONTROL AS EDIT
                        POSITION 105 13 52 22
                        LABEL "Cod. Departamento"
                        BORDER DOUBLE DOWN
                        VARIABLE departamentos.cod_dpto
```

```
            LIKEVAR
        CONTROL AS TEXT
            POSITION 7 39 37 22
            NOBORDER
            DATATYPE CHAR
            TAGS "Nombre"
        CONTROL AS EDIT
            POSITION 105 39 196 22
            LABEL "Nombre"
            BORDER DOUBLE DOWN
            VARIABLE departamentos.nombre
            LIKEVAR
    END
    CONTROL AS BUTTON
        POSITION 516 5 76 22
        ATTACH RIGHT
        LABEL "&Buscar"
        ICON 6 FILE "Small Buttons"
        COMMAND QueryByForm
    CONTROL AS BUTTON
        POSITION 516 31 76 22
        ATTACH RIGHT
        LABEL "&Nuevo"
        ICON 84 FILE "Small Buttons"
        COMMAND New
    CONTROL AS BUTTON
        POSITION 516 57 76 22
        ATTACH RIGHT
        LABEL "&Añadir"
        ICON 157 FILE "Small Buttons"
        COMMAND Add
    CONTROL AS BUTTON
        POSITION 516 83 76 22
        ATTACH RIGHT
        LABEL "&Borrar"
        ICON 159 FILE "Small Buttons"
        COMMAND Delete
    CONTROL AS BUTTON
        POSITION 516 109 76 22
        ATTACH RIGHT
        LABEL "&Modificar"
        ICON 158 FILE "Small Buttons"
        COMMAND Update
    CONTROL AS BUTTON
        POSITION 512 264 76 22
        ATTACH RIGHT BOTTOM
        LABEL "&Salir"
        ICON 4 FILE "Small Buttons"
        COMMAND Close
    CONTROL AS BOX
        POSITION 7 78 490 210
        ATTACH LEFT RIGHT BOTTOM
        NOLABEL
        BORDER DOUBLE FRAME
    BEGIN
        CONTROL IDLINES AS GRID
            POSITION 0 0 476 169
            ATTACH ALL
            NOBORDER
            SCROLL VERTICAL HORIZONTAL
            TABLE empleados
```

```
    PARENTGRID
BEGIN
    CONTROL AS EDIT
        POSITION 0 0 66 22
        LABEL "DNI y letra"
        NOBORDER
        VARIABLE empleados.dni
        LIKEVAR
    CONTROL AS EDIT
        POSITION 67 0 162 22
        LABEL "Nombre"
        NOBORDER
        VARIABLE empleados.nombre
        LIKEVAR
    CONTROL AS EDIT
        POSITION 230 0 192 22
        LABEL "Apellidos"
        NOBORDER
        VARIABLE empleados.apellidos
        LIKEVAR
    CONTROL AS EDIT
        POSITION 423 0 192 22
        LABEL "Domicilio"
        NOBORDER
        VARIABLE empleados.domicilio
        LIKEVAR
    CONTROL AS EDIT
        POSITION 616 0 192 22
        LABEL "Localidad"
        NOBORDER
        VARIABLE empleados.localidad
        LIKEVAR
    CONTROL AS EDIT
        POSITION 809 0 90 22
        LABEL "Cod. Postal"
        NOBORDER
        VARIABLE empleados.cod_postal
        LIKEVAR
    CONTROL AS EDIT
        POSITION 900 0 90 22
        LABEL "Teléfono"
        NOBORDER
        VARIABLE empleados.telefono
        LIKEVAR
END
CONTROL AS BUTTON
    POSITION 313 176 165 22
    ATTACH RIGHT BOTTOM
    LABEL "Modificar Linea"
    ICON 158 FILE "Small Buttons" CENTER
    COMMAND UpdateLine
CONTROL AS BUTTON
    POSITION 168 176 148 22
    ATTACH LEFT RIGHT BOTTOM
    LABEL "Borrar Linea"
    ICON 159 FILE "Small Buttons" CENTER
    COMMAND DeleteLine
CONTROL AS BUTTON
    POSITION 0 176 165 22
    ATTACH LEFT BOTTOM
    LABEL "Añadir Linea"
```

```
                        ICON 157 FILE "Small Buttons" CENTER
                        COMMAND AddLine
                END
            END
            CONTROL AS BAR
                POSITION 0 295 597 27
                FOREGROUND RGB 0 0 0
                ATTACH LEFT RIGHT BOTTOM
                BORDER ETCHED EXTRA 3
                BARTYPE STATUS
            BEGIN
                CONTROL AS PANEL
                    POSITION 3 0 547 19
                    FOREGROUND RGB 0 0 0
                    ATTACH ALL
                    NOBORDER
                    PANELTYPE COMMENT
                CONTROL AS PANEL
                    POSITION 553 0 35 19
                    FOREGROUND RGB 0 0 0
                    ATTACH TOP RIGHT BOTTOM
                    BORDER DOWN
                    PANELTYPE EDITMODE
            END
        END
    end
END
CODE CLASS CHLdepart BEGIN
//{{CODEBEGIN
On command AddLine
begin
    MasterTable.ChildTable.Add;
end

On command DeleteLine
begin
    MasterTable.ChildTable.Delete;
end

On command UpdateLine
begin
    MasterTable.ChildTable.Update;
end
//{{CODEEND
END
CODE BEGIN
//{{CODEBEGIN
main
objects
begin
oCHLdepart as CHLdepart
end
begin
Sql.AttachConnection( "conexion2" );
Sql.Connect( "Datos2" );
oCHLdepart.Run;
Sql.Disconnect;
Sql.DettachConnection;
end
//{{CODEEND
END
```

Código del módulo RHLdepart

```
REPOSITORY repositorio2
INCLUDES BEGIN
    TEMPLINC2
END
CONSTANTS  begin
end
CLASSES BEGIN
    CRHLdepar is absHLPage begin
        objects begin
            cod_dpto as departamentos.cod_dpto
            nombre as departamentos.nombre
            dni as empleados.dni
            empleados_nombre as empleados.nombre
            apellidos as empleados.apellidos
            domicilio as empleados.domicilio
            localidad as empleados.localidad
            cod_postal as empleados.cod_postal
            telefono as empleados.telefono
            empleados_cod_dpto as empleados.cod_dpto
        end
        INTERFACE
            SIZE 595 842
            MARGIN 24 33 24 66
        BEGIN
            CONTROL IDMASTER AS BOX
                POSITION 11 49 524 117
                BACKGROUND COLOR transparent
                NOLABEL
                ATTACH LEFT RIGHT
                BORDER ALL
            BEGIN
                CONTROL AS TEXT
                    POSITION 5 10 96 22
                    TAGS "Cod. Departamento"
                CONTROL AS VAR
                    POSITION 107 10 42 22
                    LABEL "Cod. Departamento"
                    VARIABLE cod_dpto
                    LIKEVAR
                CONTROL AS TEXT
                    POSITION 5 39 44 22
                    TAGS "Nombre"
                CONTROL AS VAR
                    POSITION 107 39 168 22
                    LABEL "Nombre"
                    VARIABLE nombre
                    LIKEVAR
            END
            CONTROL IDDATE AS VAR
                POSITION 11 14 79 22
                DATATYPE DATE MASK 5
            CONTROL IDTITLE AS TEXT
                POSITION 87 14 376 20
                FONT "Times New Roman" 18 BOLD ITALIC
                ATTACH LEFT RIGHT
                ALIGNMENT CENTER
                TAGS "Listado de departamentos"
            CONTROL AS TEXT
                POSITION 471 14 32 22
```

```
    TAGS "Pag :"
CONTROL IDPAGE AS VAR
    POSITION 507 14 26 22
    ALIGNMENT RIGHT
    DATATYPE SMALLINT
CONTROL IDGROUP AS GROUP
    POSITION 11 214 524 554
    BACKGROUND COLOR transparent
    NOLABEL
    ATTACH ALL
    AUTOSIZE
    REPEAT 0
BEGIN
    CONTROL IDLINES AS BOX
        POSITION 0 0 524 23
        BACKGROUND COLOR transparent
        NOLABEL
        STORE DESIGN
        ATTACH LEFT RIGHT
    BEGIN
        CONTROL AS VAR
            POSITION 0 0 55 22
            LABEL "DNI y letra"
            VARIABLE dni
            LIKEVAR
        CONTROL AS VAR
            POSITION 60 0 56 22
            LABEL "Nombre"
            VARIABLE empleados_nombre
            LIKEVAR
        CONTROL AS VAR
            POSITION 122 0 58 22
            LABEL "Apellidos"
            VARIABLE apellidos
            LIKEVAR
        CONTROL AS VAR
            POSITION 185 0 60 22
            LABEL "Domicilio"
            VARIABLE domicilio
            LIKEVAR
        CONTROL AS VAR
            POSITION 250 0 63 22
            LABEL "Localidad"
            VARIABLE localidad
            LIKEVAR
        CONTROL AS VAR
            POSITION 318 0 67 22
            LABEL "Cod. Postal"
            VARIABLE cod_postal
            LIKEVAR
        CONTROL AS VAR
            POSITION 390 0 72 22
            LABEL "Teléfono"
            VARIABLE telefono
            LIKEVAR
        CONTROL AS VAR
            POSITION 467 0 42 22
            LABEL "Cod. Departamento"
            VARIABLE empleados_cod_dpto
            LIKEVAR
    END
```

```
                END
                CONTROL IDCOLTITLE AS BOX
                    POSITION 11 176 524 29
                    BACKGROUND COLOR transparent
                    NOLABEL
                    ATTACH LEFT RIGHT
                    BORDER ALL
                BEGIN
                    CONTROL AS TEXT
                        POSITION 0 0 55 22
                        TAGS "DNI y letra"
                    CONTROL AS TEXT
                        POSITION 59 0 56 22
                        TAGS "Nombre"
                    CONTROL AS TEXT
                        POSITION 121 0 58 22
                        TAGS "Apellidos"
                    CONTROL AS TEXT
                        POSITION 184 0 60 22
                        TAGS "Domicilio"
                    CONTROL AS TEXT
                        POSITION 249 0 63 22
                        TAGS "Localidad"
                    CONTROL AS TEXT
                        POSITION 317 0 67 22
                        TAGS "Cod. Postal"
                    CONTROL AS TEXT
                        POSITION 389 0 72 22
                        TAGS "Teléfono"
                    CONTROL AS TEXT
                        POSITION 467 0 42 22
                        TAGS "Cod. Departamento"
                END
            END
        end
END
CODE CLASS CRHLdepar BEGIN
//{{CODEBEGIN
Public Preview() return Boolean
begin
    return TRUE;
end

{
Public GetRow() return Integer
begin
    return IDGROUP.CurrentRow;
end

Public SetRow(row as Integer)
begin
    IDGROUP.CurrentRow = row;
end

Public SetBand(num as smallint)
begin
    IDLINES.Page = num;
end
}
////////////////////////////////////////////////////////////////
```

```
// Wizard generated functions

Public GetMasterSelect() return Char
begin
    return "select departamentos.cod_dpto,
departamentos.nombre"+
" from departamentos" ;
end

Public GetLinesSelect() return Char
begin
    return "select empleados.dni, empleados.nombre,
empleados.apellidos, empleados.domicilio,"+
" empleados.localidad, empleados.cod_postal, empleados.telefono,
empleados.cod_dpto"+
" from empleados"+
" where empleados.cod_dpto = ?" ;
end

Public MasterFetchInto(mCursor as SqlCursor) return Boolean
begin
    return mCursor.Fetch(cod_dpto, nombre).Found;
end

Public LinesFetchInto(lCursor as SqlCursor) return Boolean
begin
    return lCursor.Fetch(dni, empleados_nombre, apellidos,
domicilio, localidad, cod_postal, telefono,
empleados_cod_dpto).Found;
end

Public OpenLinesCursor(lCursor as SqlCursor) return Boolean
begin
    return lCursor.Open(cod_dpto).Error;
end

Public GroupsNumber() return smallint
begin
    return 0;
end

Public SetGroups(mCursor as SqlCursor)
begin
    mCursor.GroupBy();
    mCursor.Totalize();
end

Public BeforeGroup(groupNumber as smallint)
begin
    switch groupNumber begin

    end
end

Public SetGroupTotals(mCursor as SqlCursor, groupNumber as
smallint)
begin
    switch groupNumber begin

    end
```

```
end

Public SetTotals(mCursor as SqlCursor)
begin

end
//{{CODEEND
END
CODE BEGIN
//{{CODEBEGIN
main
objects
begin
oCRHLdepar as CRHLdepar
end
begin
Sql.AttachConnection( "conexion2" );
Sql.Connect( "Datos2" );
oCRHLdepar.Run;
Sql.Disconnect;
Sql.DettachConnection;
end
//{{CODEEND
END
```

14. Archivos generados en este capítulo.

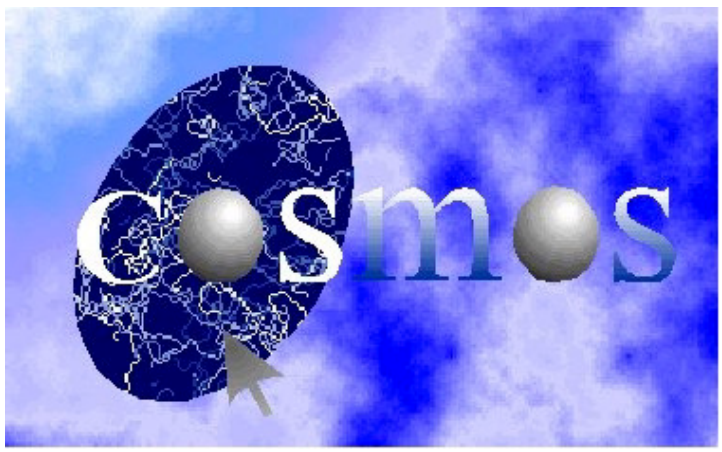
En este capítulo se han generado los siguientes archivos nuevos dentro de la carpeta "Proyecto2". Unicamente se reflejan los archivos de extensión .smd y por cada uno de ellos existirá un archivo del mismo nombre y extensión .omd que son los que contienen el fichero objeto.

Archivo	Contenido
departam.smd	Fichero ASCII que contiene el código fuente del módulo de mantenimiento de la tabla departamento.
empleado.smd	Fichero ASCII que contiene el código fuente del módulo de mantenimiento de la tabla empleados.
HLdepart.smd	Fichero ASCII que contiene el código fuente del módulo de mantenimiento de la tabla de departamentos y empleados mediante el maestro-detalle.
Mnproyec.smd	Fichero ASCII que contiene el código fuente del módulo de inicio de la aplicación.
Rdeparta.smd	Fichero ASCII que contiene el código fuente del módulo del informe de la tabla departamentos.
Remplea.smd	Fichero ASCII que contiene el código fuente del módulo del informe de la tabla empleados.
RHLdepar.smd	Fichero ASCII que contiene el código fuente del módulo del informe mediante la estructura de mestro-detalle, de las tablas de departamentos y empleados.
Templinc2.smd	Fichero ASCII con el código fuente del include "templinc2". Cuando creamos el módulo con el Wizard este realizo la copia de este archivo desde el directorio de "includes" de Cosmos.
Proyecto2.prj	Fichero que contiene el código global de la aplicación.

Capítulo 5

Utilización de varias tablas

1. Introducción.
2. Descripción de requisitos.
3. Campos de las nuevas tablas.
4. Diagrama entidad-relación completo.
5. Importar un repositorio.
6. Añadir una tabla a un repositorio.
7. Actualización de la base de datos.
8. Añadir las restantes tablas al repositorio y a la base de datos.
9. Añadir nuevos campos a las tablas existentes.
10. Crear el nuevo proyecto.
11. Ejecución de la aplicación.
12. Archivos generados en este capítulo.



1. Introducción.

En este capítulo se va a modificar el repositorio creado en el capítulo cuarto añadiendo una serie de tablas que permitirán realizar una aplicación completa para la gestión de proyectos de una empresa. También se incidirá en el trabajo con las relaciones entre tablas 1 a n.

2. Descripción de requisitos.

Una empresa se compone de diferentes departamentos, cada departamento tiene varios empleados. Esta parte ya esta realizada en el capítulo cuarto.

La empresa realiza proyectos con empresas, también existen cargos de dinero a proyectos. En un proyecto pueden trabajar varios empleados y un empleado únicamente esta asociado a un proyecto. En el apartado 3 de este capítulo se realiza el diagrama entidad-relación de esta aplicación.

Se ha de realizar la definición completa de las tablas que nos permitan realizar altas, bajas y modificaciones sobre, empresas, proyectos y cargos a un proyecto. También se deben mostrar los informes de cada una de las tablas.

3. Campos de las nuevas tablas.

Empresas

Contiene la información relativa a las empresas que solicitan la realización de algún proyecto.

- Código empresa: contiene el código interno de la empresa.
- Nombre: denominación social de la empresa.
- Dirección: domicilio social de la empresa.
- Teléfono: teléfono de la empresa.
- Fax: número de fax de la empresa.
- E-mail: dirección de correo electrónico de la empresa.
- Página Web: dirección de la página web de la empresa.
- Código proyecto: definición del proyecto. Sirve para definir la clave relacional con la tabla Proyectos.

Proyectos

Datos de los proyectos que están siendo realizados por la empresa.

- Código del proyecto: código con el que el proyecto se identifica dentro de la empresa.
- Título: nombre del proyecto.
- Descripción: breve descripción del contenido del proyecto.
- Fecha inicio: fecha cuando comienza el proyecto.
- Fecha finalización: fecha de finalización del proyecto.
- Presupuesto: presupuesto del proyecto.

Cargos

Contiene una serie de cargos en dinero a uno de los proyectos en curso.

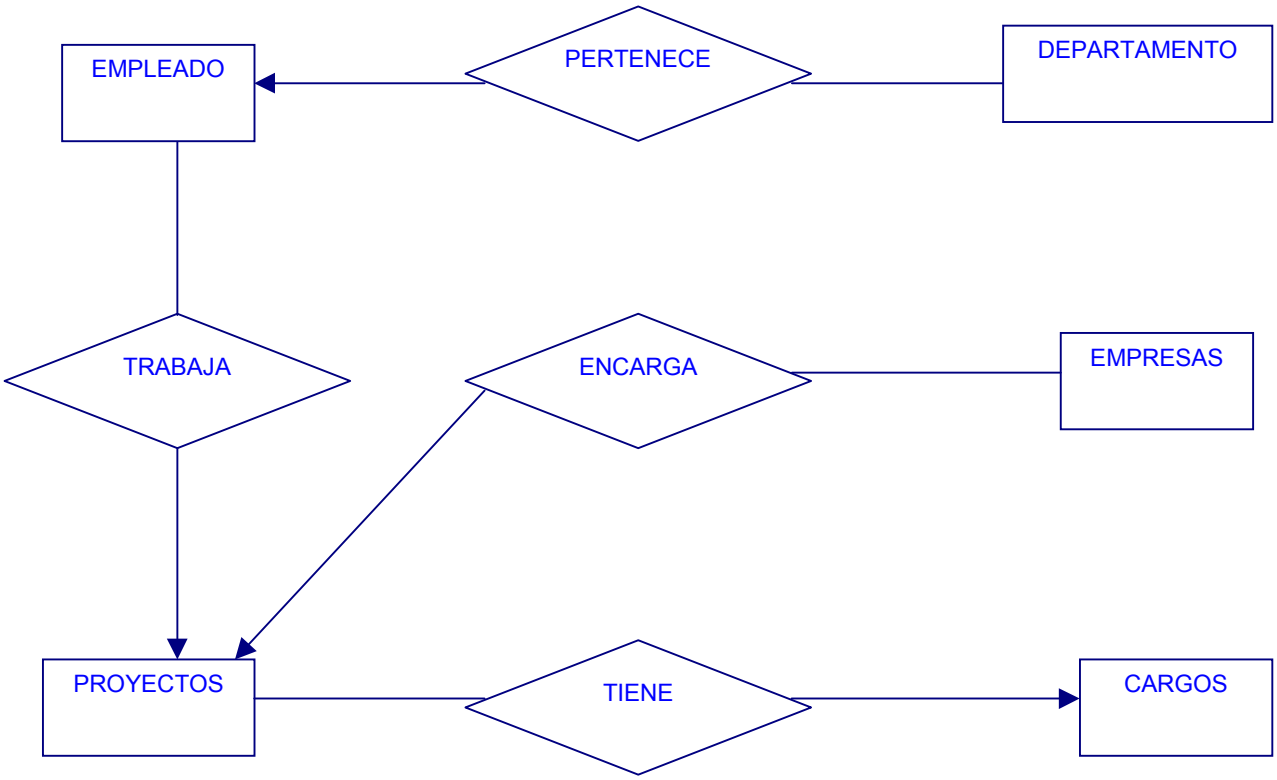
- Código Proyecto: código del proyecto al que se imputa ese cargo.
- Descripción: breve comentario sobre el objeto del cargo.
- Cuantía: cantidad en pesetas del cargo.
- Fecha: fecha en la que se realiza el cargo.

Departamentos y Empleados

En estas tablas hay que añadir un campo que será Código Proyecto, para relacionar ambas tablas con los proyectos que se desarrollan.

4. Diagrama entidad-relación completo.

El diagrama entidad-relación de esta sencilla aplicación se observa los diferentes tipos de relaciones que existen entre las tablas. Todas las relaciones son del tipo 1 a n.



5. Importar un repositorio.

Como un paso previo en el Editor de Configuración se ha definido una nueva conexión denominada “Conexion3” cuya variable de entorno DBNAME tiene como contenido “datos3”. (Véase Anexos)

En esta sección se va a realizar la importación de las tablas desarrolladas en el capítulo cuarto. Se ejecuta el Editor de Repositorios y se selecciona en el menú desplegable **View** la opcion **Import Palete**. Se muestra entonces una cuadro de diálogo donde se selecciona la opción **Load Repository** para cargar la paleta del repositorio creado en el capítulo anterior. (Figura 5.1)

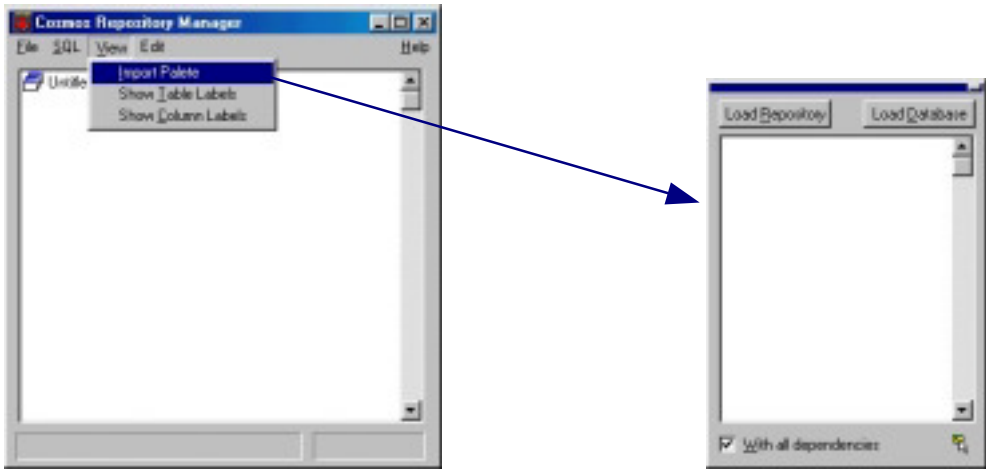


Figura 5.1. Importar tablas de otro repositorio.

Después de seleccionar el Repositorio2 se muestran las dos tablas que componen este repositorio. Como vamos a realizar una ampliación de las tablas del repositorio anterior, seleccionamos las dos tablas que existen y las arrastramos sobre la paleta del nuevo repositorio que se esta creando. (Figura 5.2)

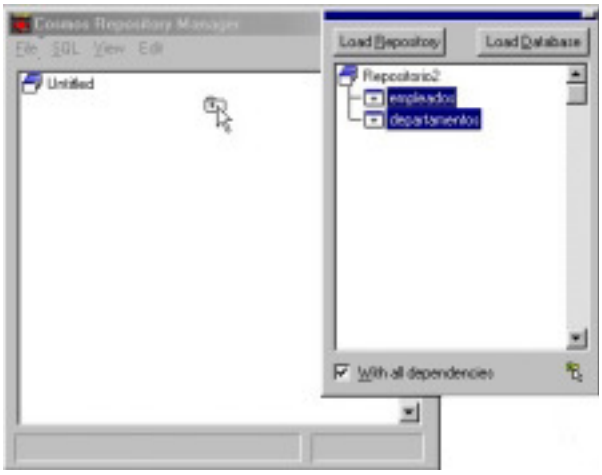


Figura 5.2. Arrastrar las tablas sobre el nuevo repositorio.

El resultado de esta acción que se ha realizado se refleja en la paleta del nuevo repositorio. En este momento seleccionamos la opcion **Save** del menú **File** y guardamos el nuevo repositorio con el nombre Repositorio3. Este nombre encabeza ahora la paleta del nuevo repositorio. (Figura 5.3)

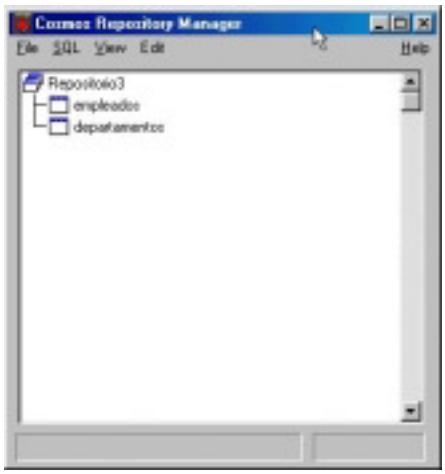


Figura 5.3. Aspecto de la paleta del repositorio.

Creamos la base de datos asociada al repositorio, para ello se dan los mismos pasos que en los capítulos anteriores.

Aunque el repositorio aún no se ha terminado de definir, la creación en este momento de la base de datos, permitirá en las siguientes secciones utilizar algunas de las opciones del Editor de Repositorios para la actualización de una base de datos a partir de un repositorio.

6. Añadir una tabla a un Repositorio.

Añadimos la nueva tabla de empresas siguiendo los mismos pasos que se dieron en el capítulo cuarto. Se añade la clave primaria de esta nueva tabla, que será cod_empresa. Las columnas que componen la tabla empresas son la que se muestra en la figura 5.4.

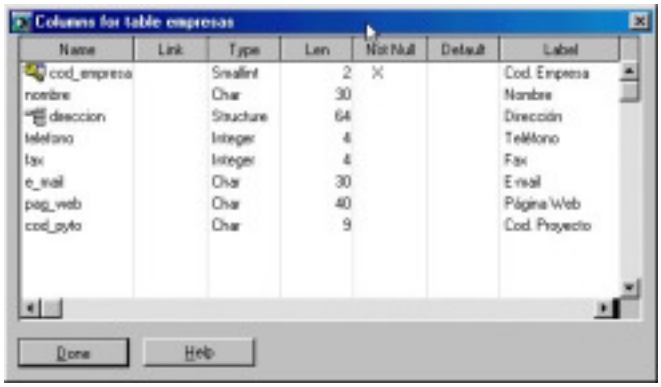


Figura 5.4. Columnas de la tabla empresa.

Hemos añadido una estructura con la dirección de la empresa que es similar a la definida en la tabla “empleados”. Después de añadir esta tabla la paleta del repositorio se muestra en la figura 5.5.

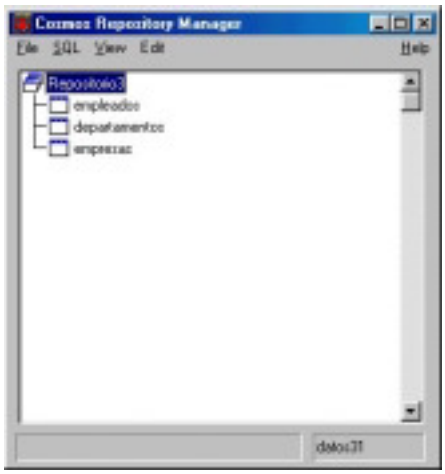


Figura 5.5. Aspecto actual del repositorio.

7. Actualización de la base de datos.

En este momento el repositorio se compone de tres tablas y la base de datos asociada únicamente tiene dos. Para comprobar si un repositorio y una base de datos tienen la misma definición podemos acudir a la opción SQL del menú y seleccionar la opción **Update with Database**. (Figura 5.6)

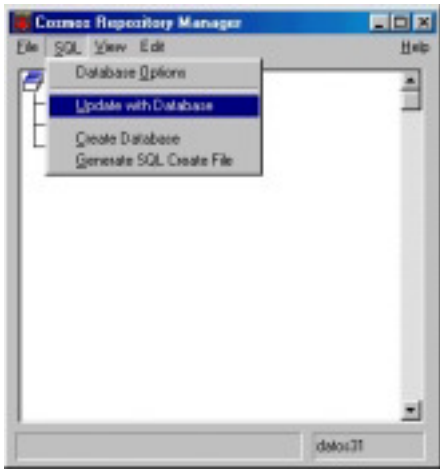


Figura 5.6. Selección de la opción Update with Database.

Se muestra un cuadro de diálogo que permite seleccionar la conexión que se desea utilizar dentro de las que se tienen definidas en el Editor de Configuración de Cosmos, también se muestra la base de datos asociada a ella. (Figura 5.7.)

Figura 5.7. Selección de la conexión y la base de datos.

Dentro de esta opción de menú se pueden realizar cuatro opciones distintas que relacionan la base de datos con el repositorio, son las siguientes:

- Mostrar las diferencias existentes entre ambos, sin actualizar.
- Actualizar el repositorio desde la base de datos.
- Actualizar la base de datos desde el repositorio.
- Actualizar preguntando todas las diferencias.

En un primer momento se va ejecutar la primera opción para comprobar que efectivamente en la base de datos falta la definición de la tabla empresas. Se selecciona por tanto la opción **Show differences (No update)**, y se muestra el cuadro de diálogo con dichas diferencias. (Figura 5.8)

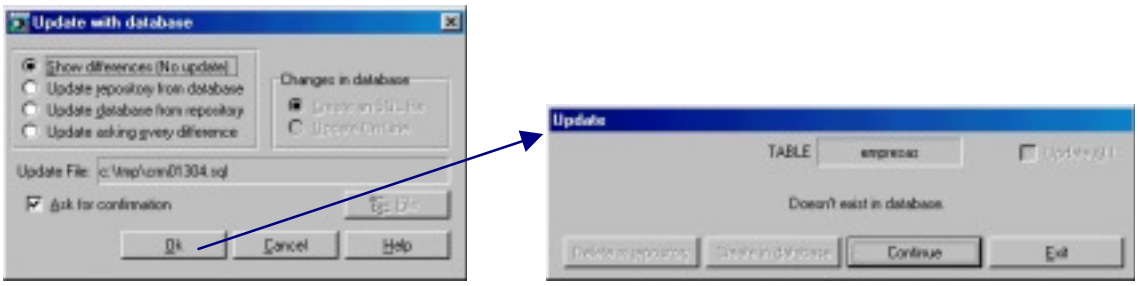


Figura 5.8. Diferencias entre el repositorio y la base de datos.

Como no existen mas diferencias cuando se pulsa **Continue**, se muestra el cuadro de diálogo de **Update with database**. En este momento seleccionamos la opción **Update database from repository**. Se activa la zona derecha del cuadro de diálogo donde se pueden seleccionar el tipo de cambios que se desean realizar sobre la base de datos, son dos:

- Crear un fichero SQL con los cambios a realizar sobre la base de datos.
- Actualizar en línea la base de datos, realizando los cambios necesarios directamente sobre la base de datos.

En este caso se selecciona la segunda opción y se crea la tabla que no existe en la base de datos. Se mostrará nuevamente el cuadro de las diferencias existentes y un botón que nos permitirá seleccionar la opción **Update**.(Figura 5.9)

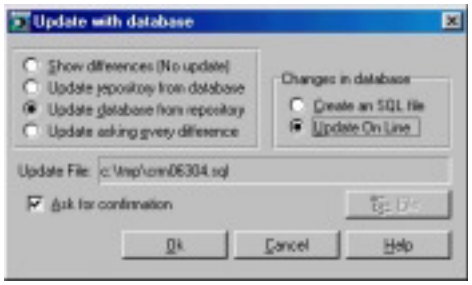


Figura 5.9. Actualización de la base de datos en línea.

8. Añadir las restantes tablas al repositorio y a la base de datos.

Se añaden las tablas “proyectos” y “cargos” al repositorio. Las columnas que componen la tabla “proyecto” se muestran en la figura 5.10. Se ha definido como clave primaria de esta tabla la columna etiquetada “Cod. Proyecto”.

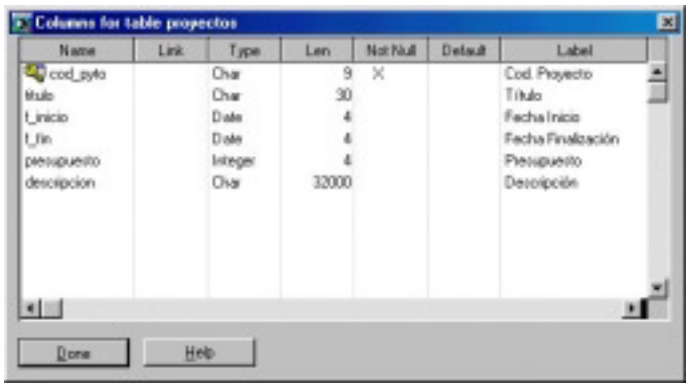


Figura 5.10. Columnas de la tabla Proyectos.

En la tabla “cargos” se ha definido un **join** denominado “cr_cargo_pyto” cuya definición se muestra en la figura 5.11. El cuadro de diálogo de las restantes columnas que componen la tabla se muestra en la Figura 5.12.

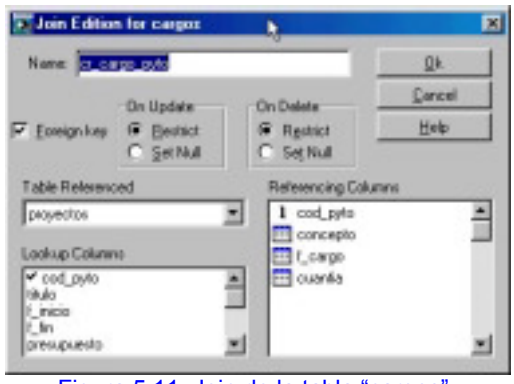


Figura 5.11. Join de la tabla “cargos”.

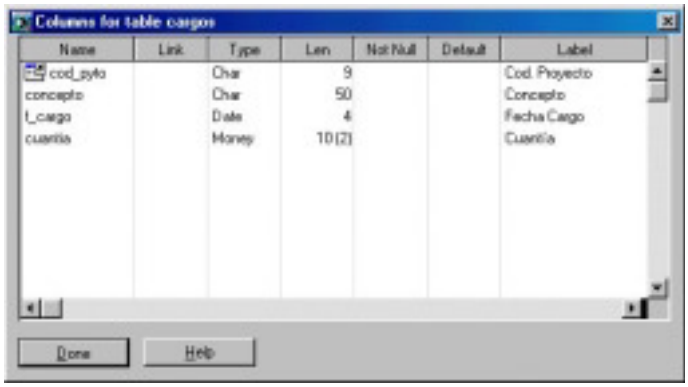


Figura 5.11. Columnas de la tabla “cargos”.

9. Añadir nuevos campos a las tablas existentes.

En la tabla departamento y en la tabla empleados se debe añadir la columna “Código de Proyecto” y la clave referencial que relaciona cada una de estas tablas con la tabla Proyectos definida en el punto anterior.

En la figura 5.12 se muestra el aspecto actual de la paleta de repositorio. Si se hace doble click sobre la tabla “departamentos” se muestra el contenido de las columnas de esta tabla.

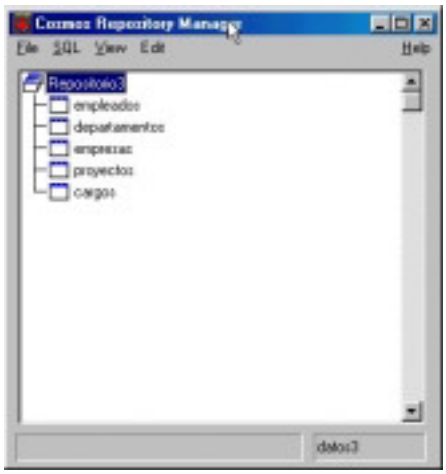


Figura 5.12. Paleta del repositorio.

Pulsando el botón derecho del ratón sobre las columnas de la tabla “departamentos” se selecciona la opción **New** para añadir la nueva columna de “Código proyecto” sobre esta tabla. (Figura 5.13)

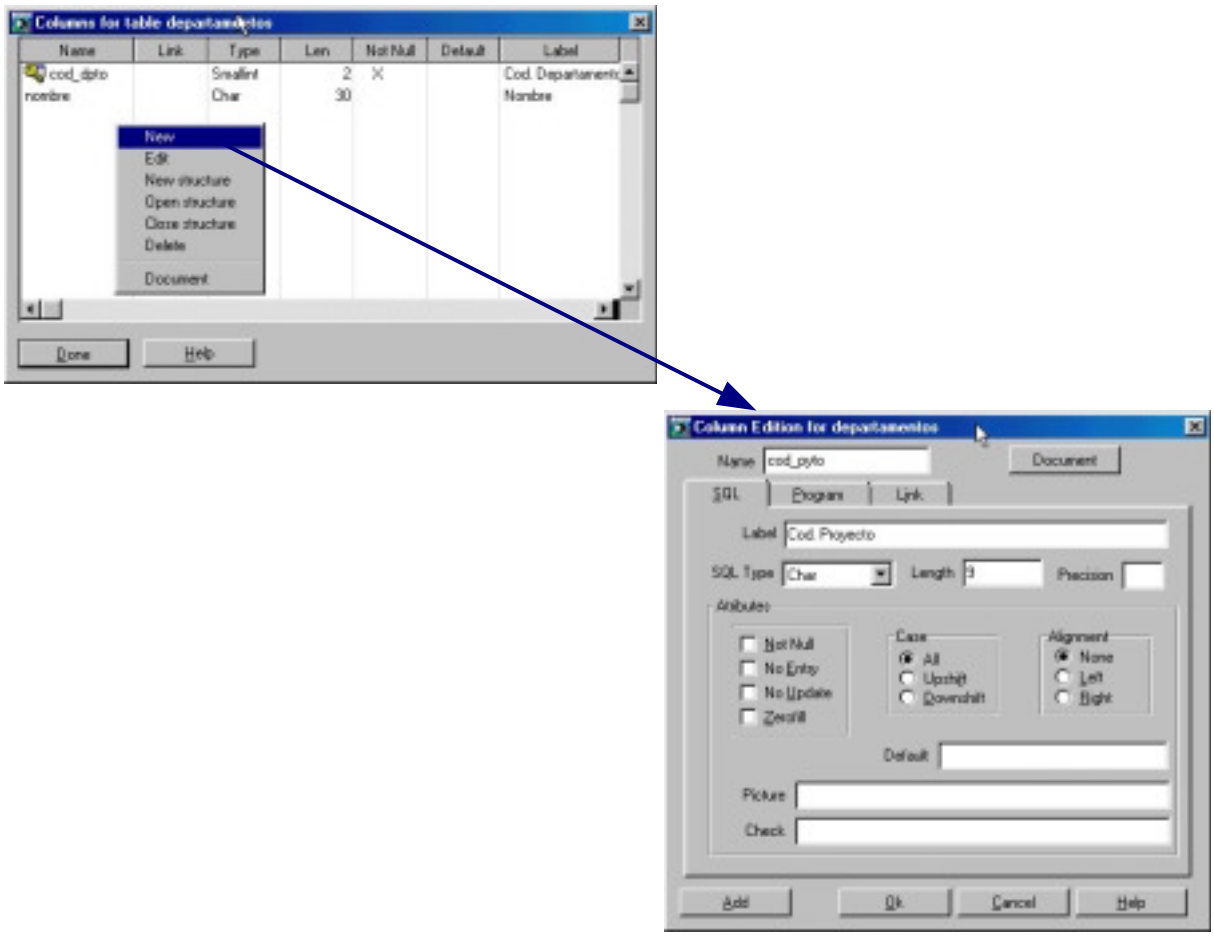


Figura 5.13. Añadir la columna “cod_pyto” a la tabla departamentos.

Sobre esta misma tabla se define una clave referencial llamada “cr_dpto_pyto” que relaciona la columna añadida con la clave primaria de la tabla “proyectos”. De este modo se consigue definir la relación 1 a n entre la tabla “departamentos” y la tabla “proyectos”. (Figura 5.14)

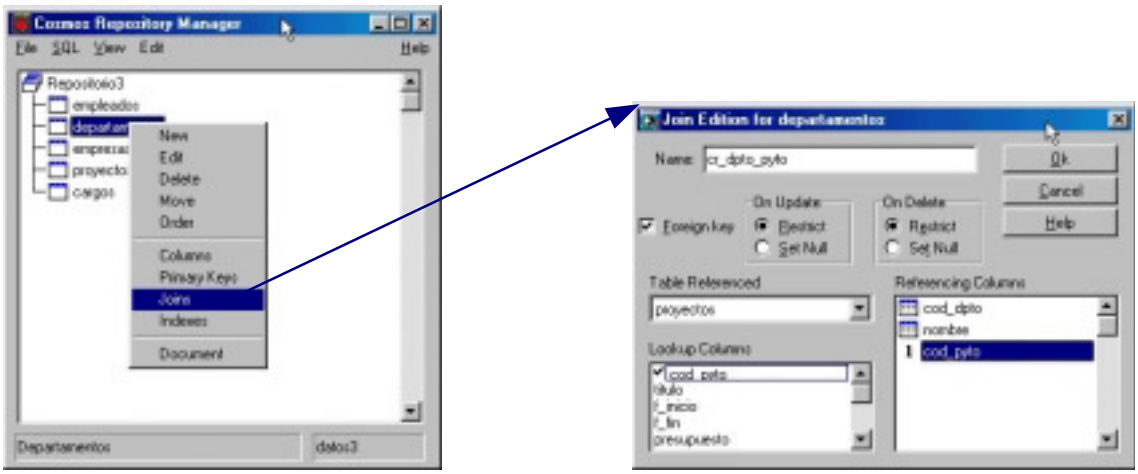


Figura 5.14. Definición de la clave referencial.

Sobre la tabla “empleados” se siguen los mismos pasos dados sobre “departamentos” definiendo la clave referencial con el nombre “cr_emp_pytos”. Las columnas de ambas tablas con la definición de las claves referenciales respectivas se muestran en la figura 5.15.

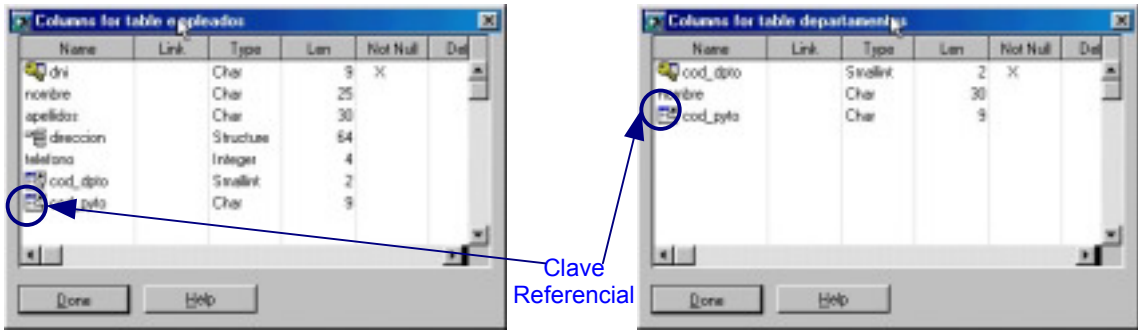


Figura 5.15. Columnas de las tablas “departamentos” y “empleados” modificadas.

En este momento se realiza la actualización de la base de datos con los nuevos cambios realizados sobre el repositorio. Para realizar esto se siguen los mismos pasos que se dieron en el punto 7 de este capítulo.

10. Crear el nuevo proyecto.

En este momento tenemos modificado el repositorio con las nuevas tablas que necesitamos en la aplicación.

En este momento podemos generar la aplicación completa del forma similar a lo realizado en el capítulo cuarto, mediante el asistente de proyectos. En este capítulo para adaptar mejor la aplicación a nuestros requerimientos se hará uso de los wizards de módulos.

Lo primero que hay que hacer es seleccionar la opción **New** dentro del menú desplegable **File** del Editor Visual de Cosmos. En el cuadro de diálogo se muestra seleccionamos Project Document, se solicita la ubicación del nuevo proyecto y el nombre del mismo. (Figura 5.16)

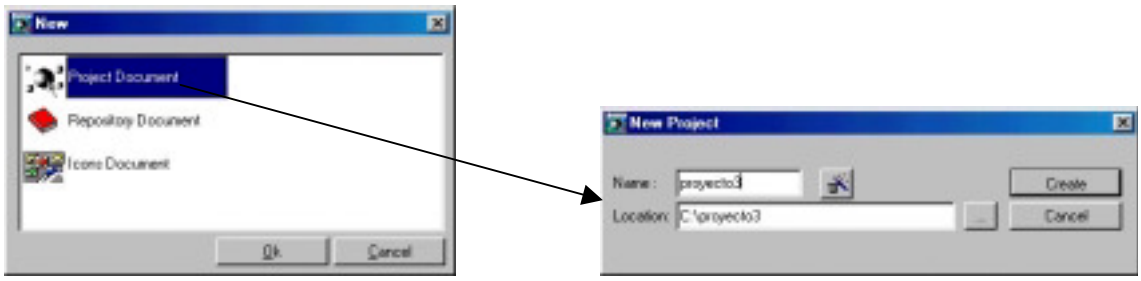


Figura 5.16. Crear un nuevo proyecto.

En este momento se tiene ya creada la definición básica del nuevo proyecto. El aspecto actual de la paleta del nuevo proyecto se muestra en la figura 5.17.

Figura 5.17. Paleta del proyecto.

Los pasos que se van a realizar a partir de este momento son los siguientes:

- Añadir el repositorio al proyecto.
- Crear las carpetas necesarias para la aplicación.
- Crear los módulos que contendrá cada carpeta.
- Crear el menú que enlace los módulos.
- Compilación y construcción de la aplicación completa.

En los siguientes apartados se va a ir desarrollando cada una de estas partes.

Añadir el repositorio al proyecto.

Se va añadir el repositorio modificado en este capítulo al nuevo proyecto que se ha definido. Para ello se selecciona en la paleta del proyecto la sección **Repositories** y se pulsa el botón secundario del ratón. Seleccionamos la opción **Import** y seleccionamos el repositorio que queremos añadir al proyecto. En este caso se trata de Repositorio3. Toda la secuencia de pasos seguida se muestra en la figura 5.18.

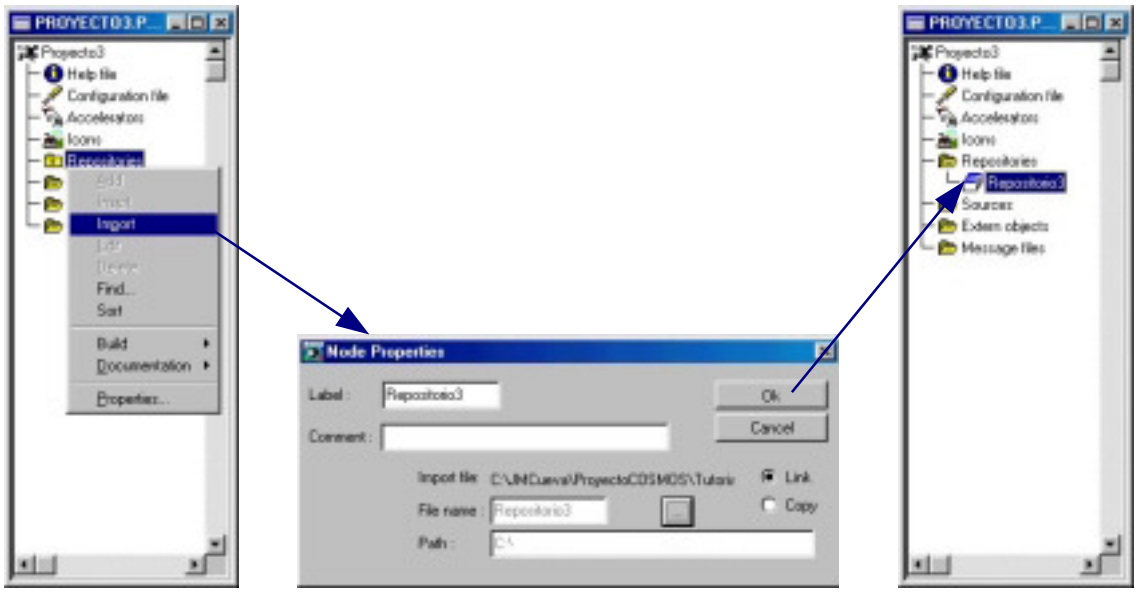


Figura 5.18. Importar el Repositorio al proyecto.

Crear las carpetas necesarias para la aplicación.

En esta aplicación va a haber dos tipos de pantallas, las que permiten acceder y actualizar los datos de una tabla en particular y las que permiten realizar el mantenimiento de una de las relaciones que se han definido, mediante controles de tipo grid.

Para tener diferenciados ambos tipos se van a crear dos carpetas dentro de la sección Sources de la paleta del proyecto. Una se llamará Forms y la otra Maestro-detalle.

Seleccionamos la sección **Sources** dentro de la paleta del proyecto y pulsamos el botón secundario del ratón sobre la misma. De las diferentes opciones que se muestran se selecciona **Add** que permite añadir tanto un módulo como un grupo. Un grupo es una utilidad que proporciona Cosmos con objeto de tener asociados los diferentes módulos que tienen relación entre sí.

En el cuadro de diálogo que se muestra se tiene que señalar la casilla **Group** y escribir la etiqueta y el comentario del nuevo grupo o carpeta. De la misma forma que se ha realizado la definición de la carpeta de Forms se actúa para crear la carpeta de Maestro-detalle. En la figura 5.19 se muestra el proceso seguido para la definición del primer grupo y como se refleja la definición en la paleta del proyecto.

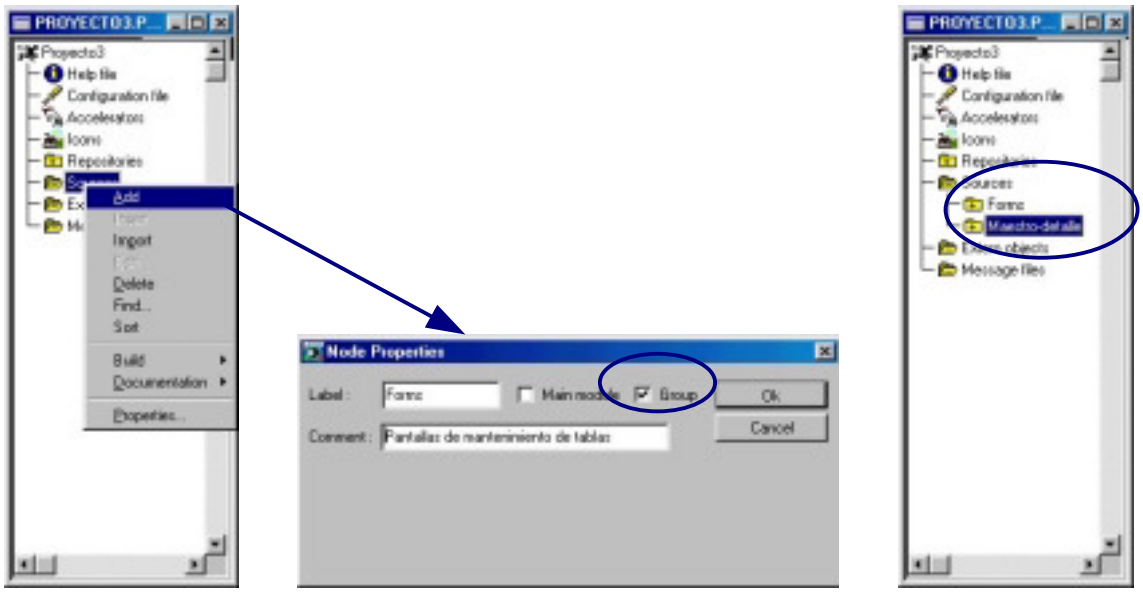


Figura 5.19. Definición de grupos.

Crear los módulos que contendrá la carpeta Forms.

Se van a crear los módulos que permitirán realizar el mantenimiento de las diferentes tablas. Para realizar la construcción de los módulos se va a hacer uso de los asistentes de módulos de Cosmos. Se seguirán los mismos pasos que se efectuaron en el capítulo primero para la creación del primer proyecto. De esta forma se crean los siguientes módulos:

- ABM_dpto: mantenimiento de la tabla departamentos.
- ABM_empleados: mantenimiento de la tabla empleados.
- ABM_proyectos: mantenimiento de la tabla proyectos.
- ABM_empresas: mantenimiento de la tabla empresas.
- ABM_cargos: mantenimiento de la tabla cargos a proyectos.

El resultado de la definición de los diferentes módulos se refleja en la paleta del proyecto como se muestra en la figura 5.20.

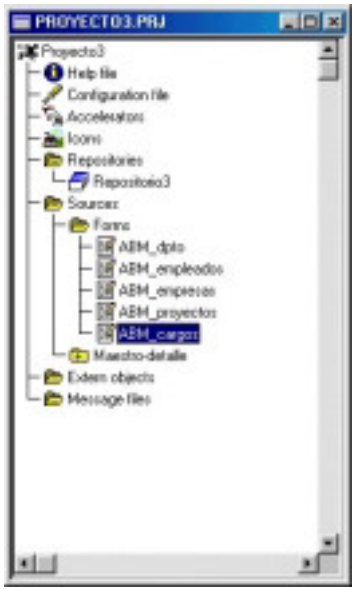


Figura 5.20. Paleta del proyecto después de añadir los módulos de mantenimiento.

Crear los módulos que contendrá la carpeta Maestro-detalle.

Para realizar estos módulos también se hará uso de los asistentes de módulos. Para seguir adecuadamente la definición de los módulos, se va ir paso a paso en la definición del módulo "md_dpto_empleados" que relaciona la tabla de departamentos y la tabla de empleados.

Sobre la carpeta de "Maestro-detalle" se pulsa en botón secundario del ratón y se selecciona la opción Add. Se muestra un cuadro de diálogo donde se escribe el nombre del nuevo módulo, a continuación se pulsa el icono  para invocar al Wizard de módulos. (Figura 5.21)

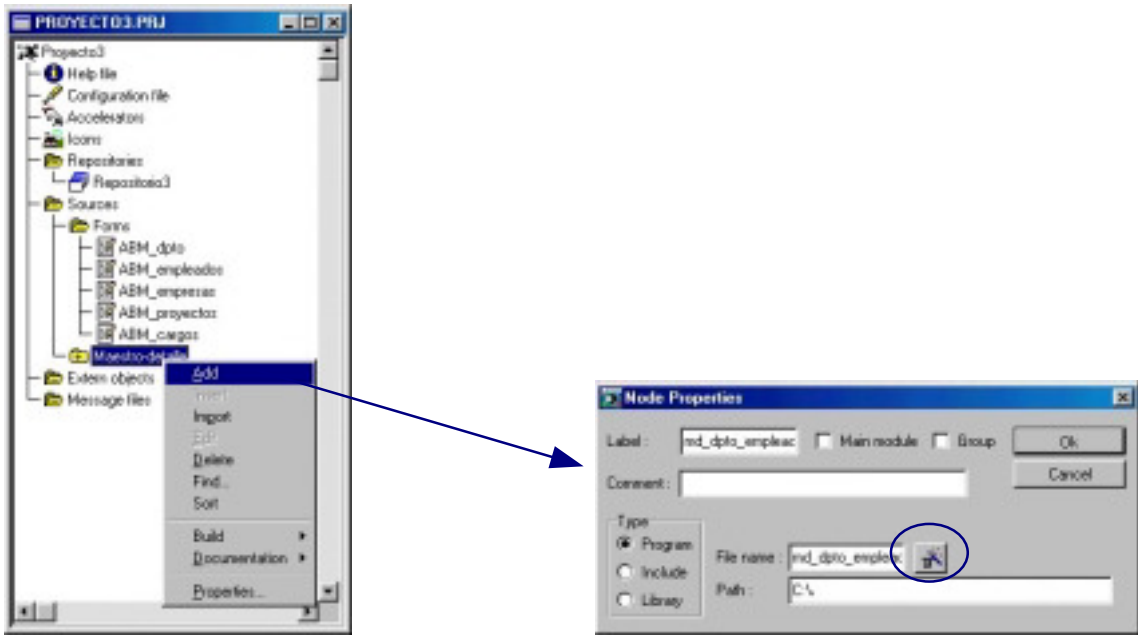


Figura 5. 21. Definición del nuevo módulo.

Al ejecutarse el asistente de módulos o Wizard, se muestran los cuadros de diálogo donde se debe introducir o seleccionar la información necesaria para la creación del módulo.

Los dos primeros cuadros nos permiten seleccionar el tipo de módulo que se desea realizar, en este caso se selecciona el tipo Head-Lines Form. El repositorio que esta asociado al módulo, se selecciona en el siguiente cuadro de diálogo. Además se puede introducir el nombre de la clase del módulo. (Figura 5.22)

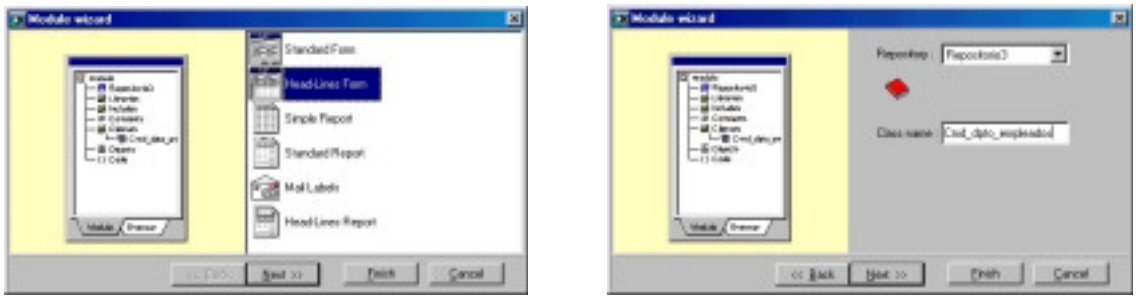


Figura 5.22. Selección del tipo de módulo y el repositorio asociado al mismo.

En la Figura 5.23. se muestra los siguientes pasos del Wizard de módulos. Se selecciona la tabla que representa el papel de maestro y las columnas de esta tabla que se mostrarán en la pantalla del módulo. En el siguiente paso se realiza lo mismo para la tabla que representa las líneas de la pantalla del módulo.

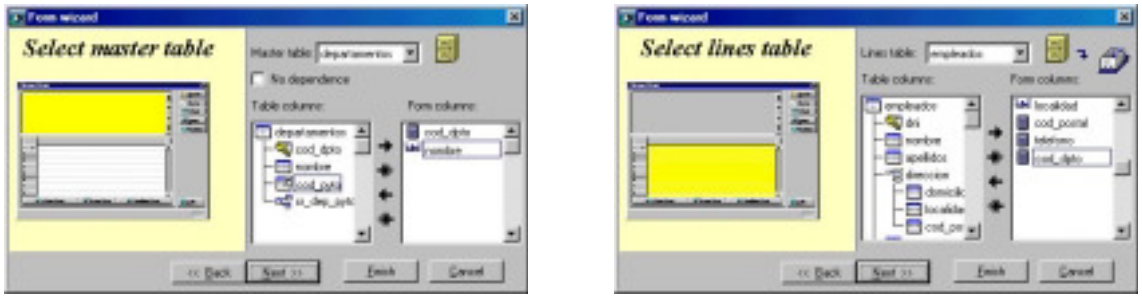


Figura 5.23. Seleccionar las tablas del módulo y sus columnas.

El último paso del Wizard del módulos es como establecer la conexión con la base de datos. (Figura 5.24)

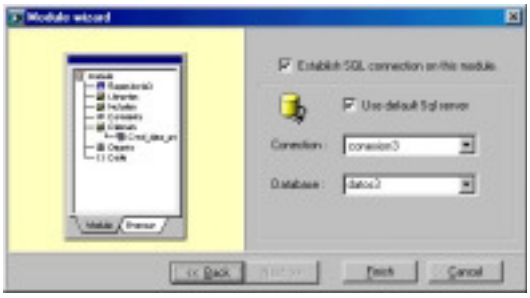


Figura 5.24. Establecer la conexión con la base de datos.

De forma similar a lo ejecutado en los pasos previos se realiza la definición de los demás módulos que componen la carpeta “Maestro-detalle”. Después de esta definición el resultado aparece en la paleta del proyecto como se muestra en la figura 5.25.

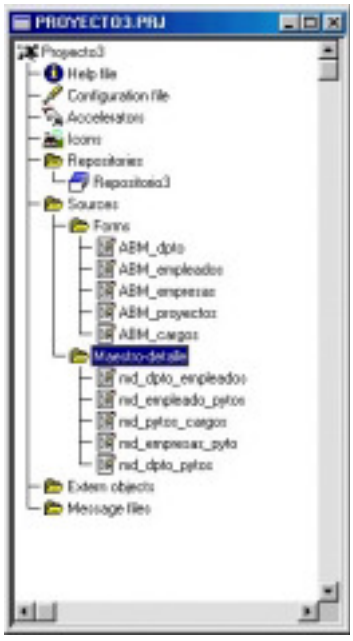


Figura 5.25. Paleta del proyecto con los nuevos módulos.

Crear el menú que enlace los módulos.

En este momento ya están definidos todos los módulos de la aplicación. Se va a definir un módulo **Main** que se ejecutará cuando se arranque la aplicación. Este módulo **main** contendrá un menú que lanzará la ejecución de los diferentes módulos de la aplicación.

Se selecciona la sección Sources y pulsando el botón secundario del ratón se elige la opción Add. Las propiedades del nuevo módulo se muestran en la figura 5.26.

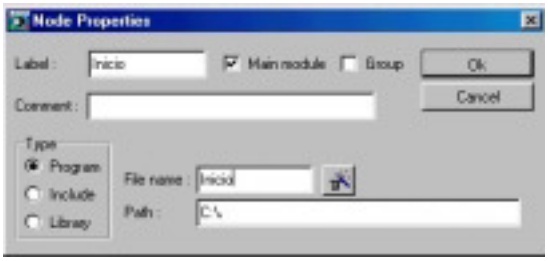


Figura 5.26. Propiedades del módulo Inicio de tipo main.

Dentro de este módulo se va a definir una clase derivada de Form. Para ello se abre la paleta del nuevo módulo haciendo doble click de ratón sobre el nombre del mismo en la paleta del proyecto. Seleccionando la sección **Classes** de la paleta y se pulsa el botón secundario del ratón, se selecciona la opción **Add**. Los pasos seguidos, las propiedades de la nueva clase y el resultado de la definición, se muestran en la figura 5.27.

Figura 5.27. Definición de la clase clnicio.

Sobre la sección **Menus** de la nueva clase se pulsa el botón secundario del ratón para añadir un objeto de la clase Menu. Recibe el nombre de “menuPpal”. El desarrollo de este punto se muestra en la figura 5.28.

Figura 5.28. Crear un nuevo menú.

En este momento se tiene definido el nuevo menú. Cada opción del menú lanzará la ejecución de uno de los módulos que se han definido anteriormente. Para que esto se realice hay que escribir los comandos asociados. Para definir estos comandos se selecciona la sección Code de la clase clnicio y se escribe el código. El código completo de la clase clnicio se muestra a continuación.

```
On Command mABMdpto
begin
    module.Run("ABM_dpto");
end
On Command mABMempleados
begin
    module.Run("ABM_empleados");
end
On Command mABMempresas
begin
    module.Run("ABM_empresas");
end
On Command mABMproyectos
begin
    module.Run("ABM_proyectos");
end
On Command mABMcargos
begin
    module.Run("ABM_cargos");
end
On Command mDptoEmpleados
begin
    module.Run("md_dpto_empleados");
end
On Command mEmpleadoPyto
begin
    module.Run("md_empleado_pytos");
end
On Command mPytosCargos
begin
    module.Run("md_pytos_cargos");
end
```

```
On Command mEmpresasPyto
begin
    module.Run("md_empresas_pyto");
end
On Command mDptoPytos
begin
    module.Run("md_dpto_pytos");
end
```

El significado de cada uno de estos comandos es el siguiente: si se lanza la ejecución del comando, por ejemplo, “mEmpleadoPyto”, se ejecutará el módulo llamado “md_empleado_pytos”. El resto de los comandos lanza la ejecución de los otros módulos.

Ahora se van a definir los diferentes componentes del menú para ello se hace doble click de ratón sobre el nombre del menú que se ha definido anteriormente o se pulsa el botón secundario del ratón, seleccionando la opción **Edit**. Se muestra un cuadro de diálogo que contiene una barra de menú vacia. Si se hace doble click de ratón sobre el cuadro resaltado, se muestra el cuadro de diálogo de las propiedades de ese campo del menú. Es en este cuadro donde se introduce el nombre de la opción de menú y el tipo, en este caso se trata de un menú desplegable. De la misma forma se definen los campos de “Mantenimiento Tablas” y “Maestro-Detalle”. (Figura 5.29)

Figura 5.29. Definir los campos principales del menú.

Se van a definir ahora las diferentes opciones que tienen los componentes de nivel 1 del menú. A la hora de definir cada una de las opciones se selecciona también el comando que ejecutará cuando se seleccione.

Para ilustrar el funcionamiento de definición se añade por ejemplo dentro de Archivo la opción Salir que lanzará el comando Close, este comando finalizará la ejecución de la aplicación. El contenido de las propiedades del componente Salir se muestra en la figura 5.30.

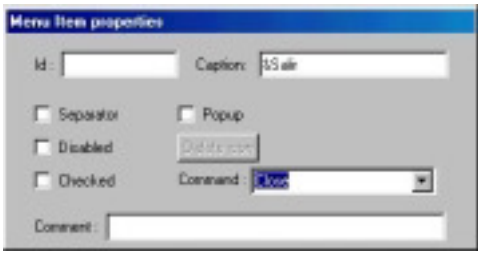


Figura 5.30. Propiedades de la opción Salir.

Los comandos que se escribieron en la sección de código de la clase clnicio se muestran en la lista desplegable, para que puedan ser asociados a alguna de las opciones del menú. (Figura 5.31)

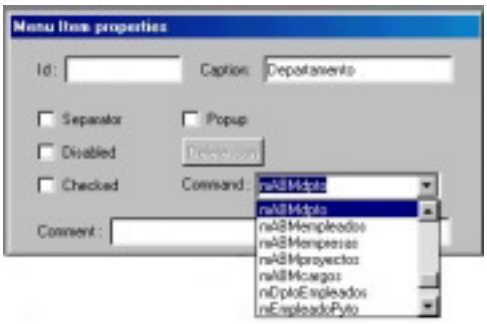


Figura 5.31. Comandos definidos.

El resto de las opciones de menú que se han creado y los comandos que lanzan cuando se seleccionan son las siguientes:

Nivel 1	Submenú	Comando
Archivo	Salir	Close
Mantenimiento Tablas	Departamento	mABMdpto
	Empleados	mABMempleados
	Proyectos	mABMproyectos
	Empresas	mABMempresas
	Cargos	mABMcargos
Maestro-Detalle	Departamentos-Empleados	mDptoEmpleados
	Empleados-Proyectos	mEmpleadoPyto
	Departamentos-Proyectos	mDptoPytos
	Empresas-Proyectos	mEmpresasProyecto
	Cargos-Proyectos	mPytosCargos

Una vez definido completamente el menú se debe seleccionar la screen de la clase cInicio. Dentro del menú principal del Editor Visual de Cosmos seleccionamos la opción Layout para definir la screen como contenedor del menú. En la figura 5.32. se muestra este paso y el aspecto actual de la Screen de la clase cInicio, en la cual se ha añadido el título “Menú principal” en sus propiedades.

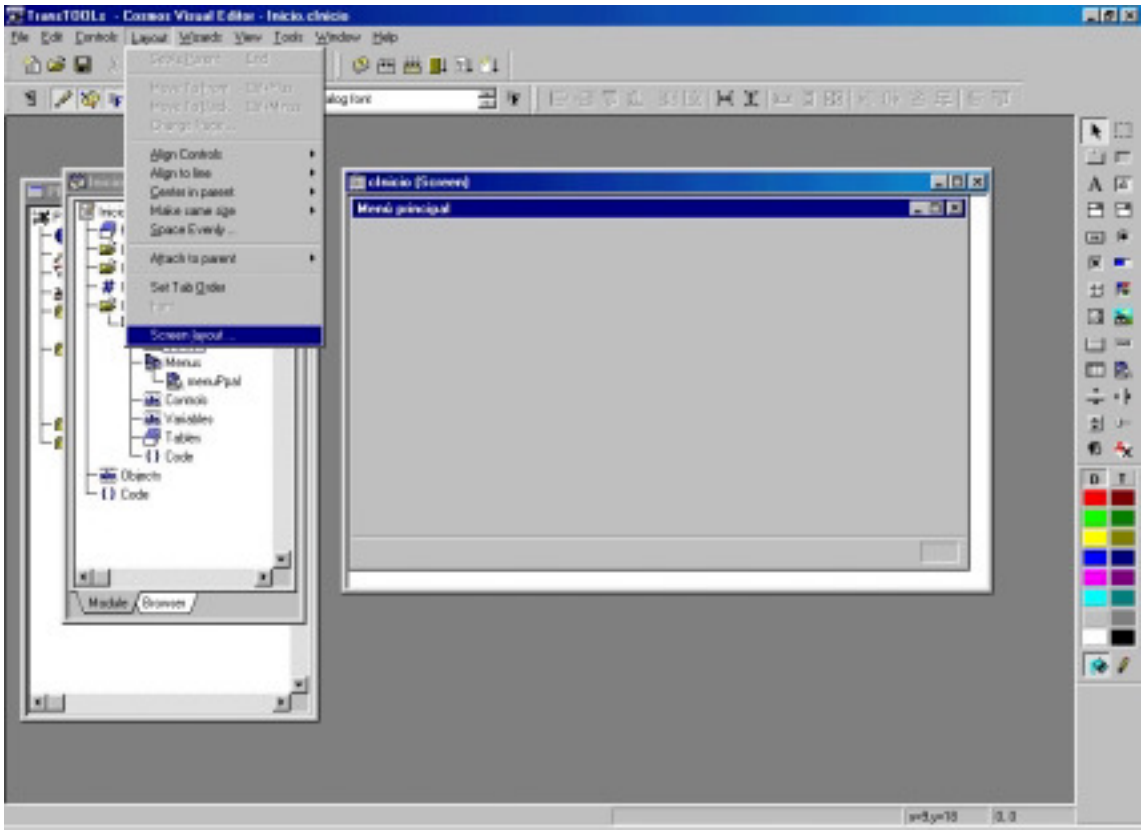


Figura 5.32. Seleccionar la opción Layout del menú principal.

Se muestra un cuadro de diálogo en el cual se selecciona la opción de opción de introducir un menú desplegable dentro de la screen. En la lista desplegable se selecciona el nombre del menú que hemos definido. Este cuadro de diálogo y el resultado se muestra en la figura 5.33.

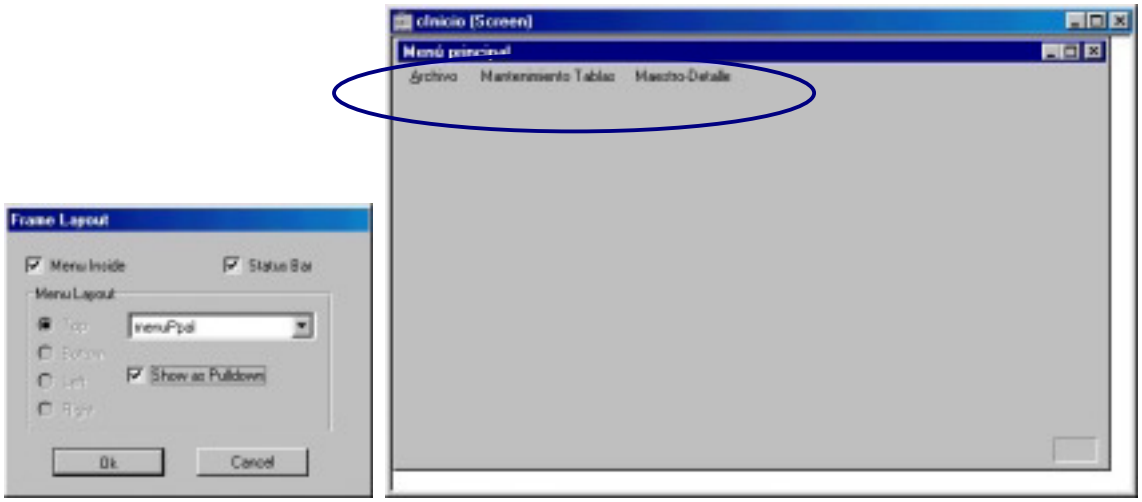


Figura 5.33. Seleccionar la definición del menú en la Screen.

Compilación y construcción de la aplicación completa.

Una vez definidos cada uno de los módulos y la screen con el menú que lanzará la ejecución de los mismos podemos seleccionar la opción **Build all** del menú **Tools** del Editor Visual para compilar todos los módulos. (Figura 5.34)

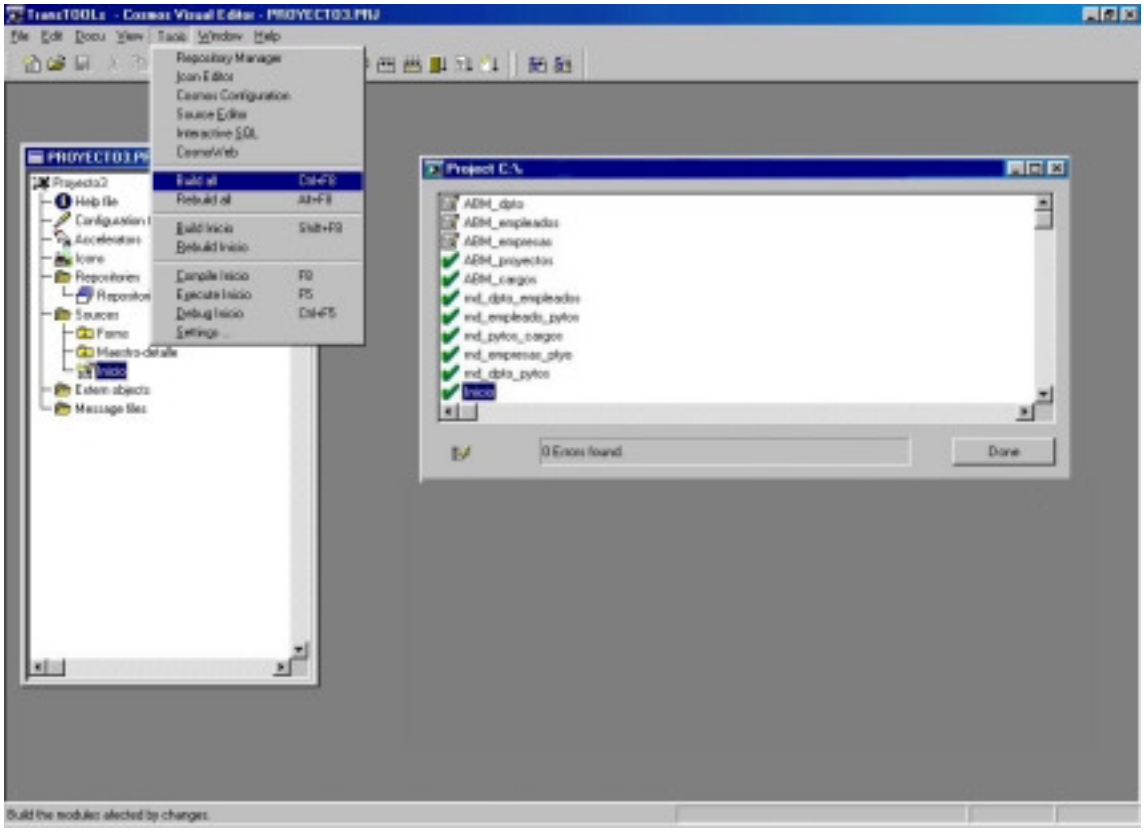


Figura 5.34. Compilación del proyecto.

11. Ejecución de la aplicación.

Cuando se ejecuta la aplicación la pantalla inicial que aparece se muestra en la figura 5.35. A partir de esta pantalla se pueden seleccionar las diferentes opciones que se han definido.

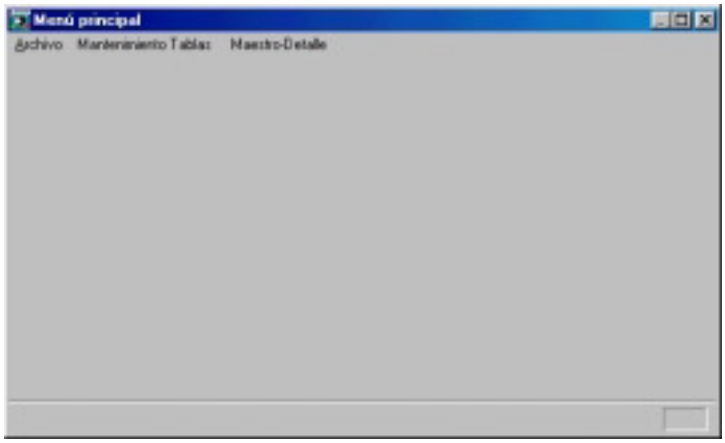


Figura 5.35. Pantalla principal de la aplicación.

Si se selecciona dentro del menú principal "Mantenimiento Tablas", se muestra el contenido de este menú. Seleccionamos la opción "Proyectos", se muestra la pantalla que nos permite realizar el mantenimiento de la tabla del mismo nombre. (Figura 5.36)

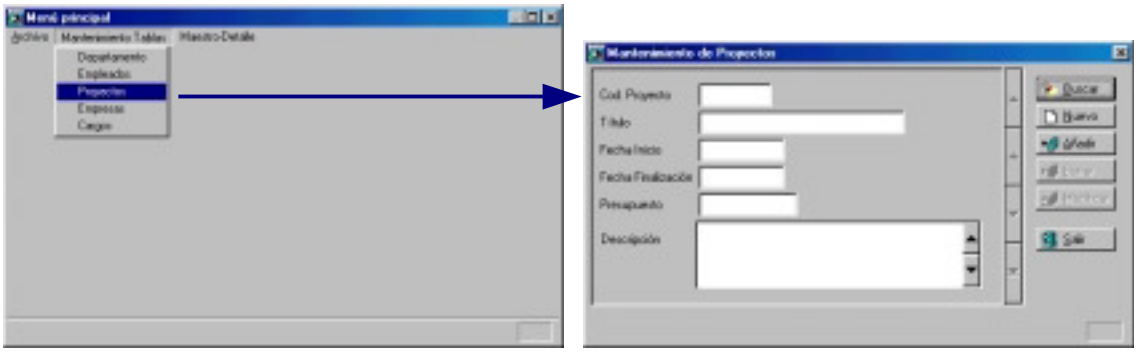


Figura 5.36. Ejecución del módulo de mantenimiento de la tabla Proyecto.

Si se selecciona el menú "Maestro-Detalle", se muestran las diferentes opciones. Como ejemplo en la figura 5.37. se muestra el contenido de este submenú y el resultado de la selección de "Cargos-Proyectos".

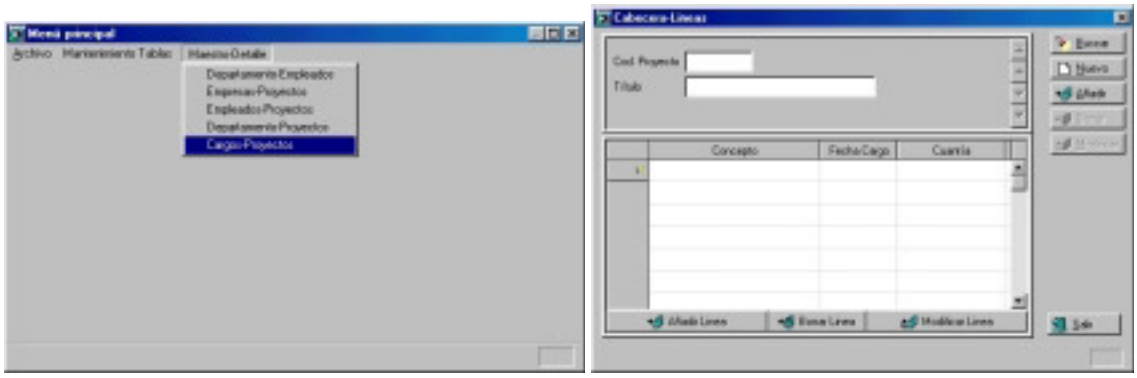


Figura 5.37. Ejecución del módulo de mantenimiento de la relación entre Proyecto y cargos.

12. Archivos generados en este capítulo.

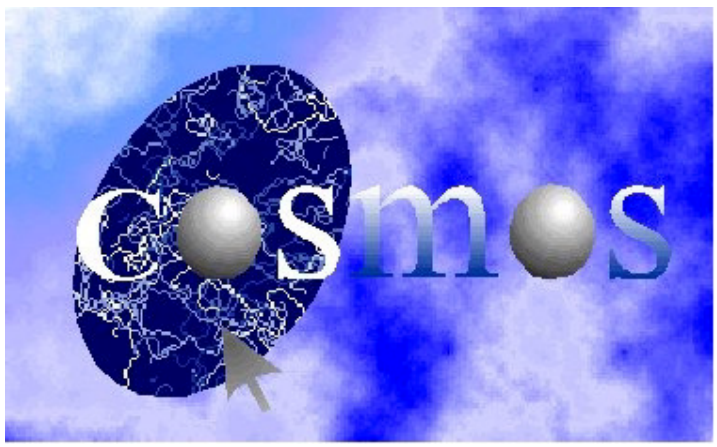
En este capítulo se han generado los siguientes archivos nuevos dentro de la carpeta "Proyecto3". Unicamente se reflejan los archivos de extensión .smd y por cada uno de ellos existirá un archivo del mismo nombre y extensión .omd que son los que contienen el fichero objeto.

Archivo	Contenido
Proyecto3.prj	Fichero que contiene el código global de la aplicación.
Inicio.smd	Fichero ASCII que contiene el código fuente del módulo denominado Inicio.
ABM_dpto.smd	Fichero ASCII que contiene el código fuente del módulo de mantenimiento de la tabla departamento.
ABM_empleados.smd	Fichero ASCII que contiene el código fuente del módulo de mantenimiento de la tabla empleados.
ABM_pytos.smd	Fichero ASCII que contiene el código fuente del módulo de mantenimiento de la tabla proyectos.
ABM_empresas.smd	Fichero ASCII que contiene el código fuente del módulo de mantenimiento de la tabla empresas.
ABM_cargos.smd	Fichero ASCII que contiene el código fuente del módulo de mantenimiento de la tabla cargos
Md_Dpto_empleados.smd	Fichero ASCII que contiene el código fuente del módulo de mantenimiento de la tabla de departamentos y empleados mediante el maestro-detalle..
Md_Empleados_Pyto.smd	Fichero ASCII que contiene el código fuente del módulo de mantenimiento de la tabla de Proyecto y empleados mediante el maestro-detalle..
Md_Empresas_pytos.smd	Fichero ASCII que contiene el código fuente del módulo de mantenimiento de la tabla de Empresas y proyectos mediante el maestro-detalle..
Md_Dpto_pytos.smd	Fichero ASCII que contiene el código fuente del módulo de mantenimiento de la tabla de departamentos y proyectos mediante el maestro-detalle..
Md_Pytos_cargos.smd	Fichero ASCII que contiene el código fuente del módulo de mantenimiento de la tabla de proyectos y cargos mediante el maestro-detalle..

Capítulo 6

Estableciendo relaciones n a n entre dos tablas

1. Introducción.
2. Descripción de la aplicación.
3. Crear las tablas en el Editor de Repositorios.
4. Crear la base de datos.
5. Creación de un fichero SQL.
6. Crear el proyecto.
7. Crear los módulos de mantenimiento.
8. Crear los módulos de maestro-detalle.
9. Crear los informes.
10. Generar la documentación de la aplicación.
11. Archivos generados en este capítulo.



1. Introducción.

A lo largo de los anteriores capítulos se han diseñado una serie de aplicaciones para el manejo de tablas y relaciones entre tablas.

Una de las relaciones más frecuentes que puede existir entre dos tablas es la llamada relación n a n. En esta relación una aparición de una de las tablas tiene asociada n apariciones de la otra tabla y viceversa.

A lo largo de esta capítulo se va diseñar un repositorio que permitirá la definición de esta relación entre dos tablas.

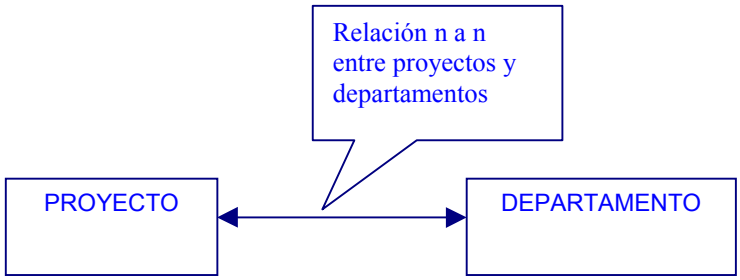
2. Descripción de la aplicación.

En un departamento de una empresa se gestionan n proyectos y además un proyecto puede ser realizado por varios departamentos de una misma empresa.

Las dos tablas que aparecen en esta aplicación junto con las columnas de las mismas son:

Tabla	Columnas	Contenido
Departamento	Cod_dpto	Código con el que se designa al departamento dentro de la empresa.
	Nombre	Nombre del departamento dentro de la empresa.
	Cod_pyto	Código del proyecto para poder crear una clave referencial entre esta columna y la tabla de proyecto.
Proyecto	Cod_pyto	Código con el que se designa al proyecto dentro de la empresa
	Nombre	Nombre del proyecto.
	Cod_dpto	Código del departamento para poder crear una clave referencial entre esta columna y la tabla de departamento.

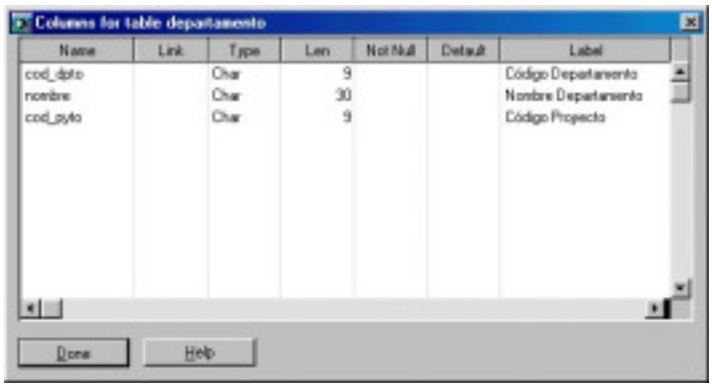
El sencillo diagrama entidad-relación de esta aplicación se muestra a continuación:



3. Crear las tablas en el Editor de Repositorio.

Se van a crear las dos tablas en el editor de repositorio. Para realizar esto lo primero es definir una nueva conexión en el Editor de Configuración de Cosmos. (Véase anexos) La nueva conexión tendrá como nombre “Conexion4” y el contenido de la variable de entorno DBNAME será “datos4”.

Definimos las tablas de forma similar a lo realizado a lo largo de los capítulos anteriores. En cada una de las columnas se ha introducido la documentación relativa a ella, de forma similar a lo realizado en el capítulo primero. Esta documentación la utilizaremos posteriormente. Las columnas de ambas tablas se muestran en la figura 6.1.



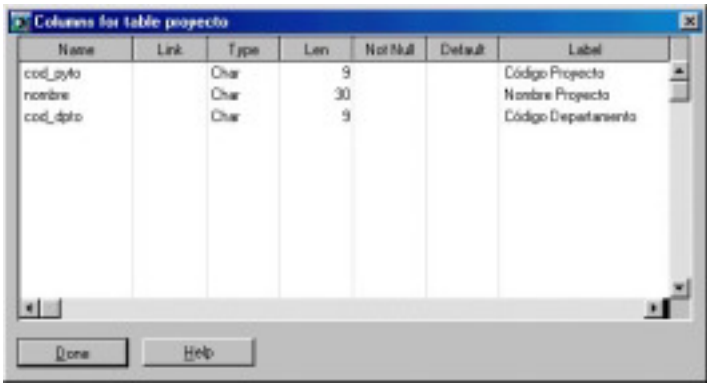


Figura 6.1. Columnas de las tablas Departamento y Proyecto

Las claves primarias de la tabla Departamento será “cod_dpto” y de la tabla Proyectos será “cod_pyto”.

Ahora se define la clave relacional de ambas tablas. Se va a mostrar la definición de la clave referencial para la tabla Proyecto. Los mismo pasos se realizarán para la definición de la clave referencial sobre la tabla Departamento.

Se pulsa el botón secundario del ratón sobre la tabla Proyecto y se selecciona del menú desplegable la opción Joins. Se muestra un cuadro de diálogo donde se pulsa el botón New para añadir una nueva clave referencial. (Figura 6.2)

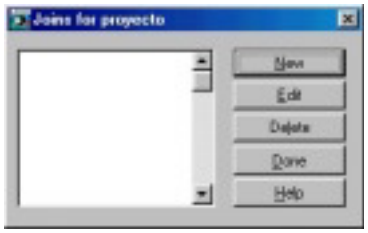


Figura 6.2. Seleccionar New.

El cuadro que se muestra a continuación es donde se realiza la definición completa del join. El nombre otorgado a esta clave referencial sobre la tabla Proyecto es “cr_dpto” y relaciona el campo de “cod_dpto” de la tabla Proyecto con el campo del mismo nombre de la

tabla Departamento. Después de pulsar el botón 

el resultado de esta definición se muestra en el cuadro de diálogo de los Joins de la tabla Proyecto. (Figura 6.3)

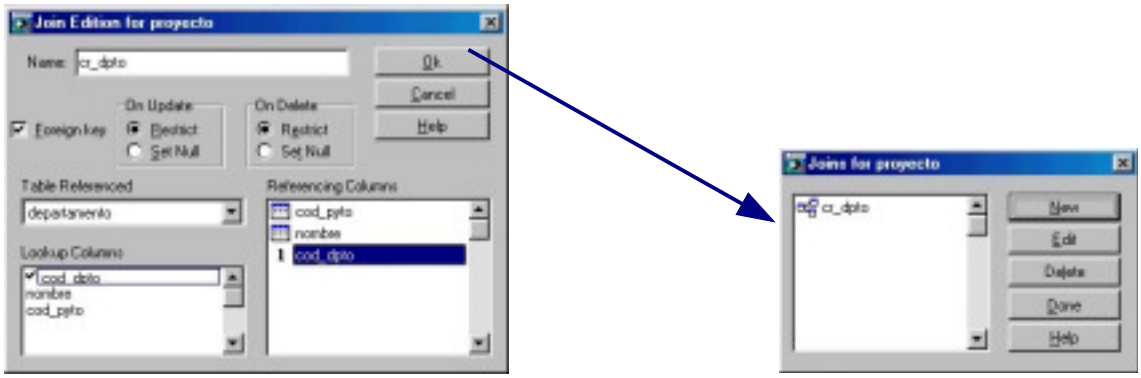


Figura 6.3. Definición de la clave referencial sobre la tabla proyecto.

Se siguen los mismos pasos sobre la tabla Departamento. En la figura 6.4. se muestra el contenido de las columnas de las dos tablas, donde se muestran mediante iconos, las claves primarias y las claves referenciales.

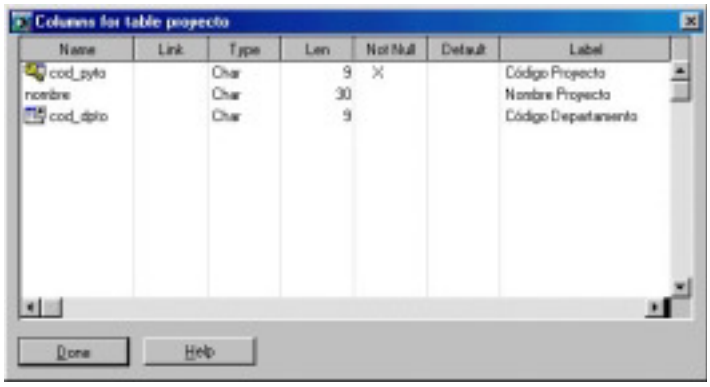
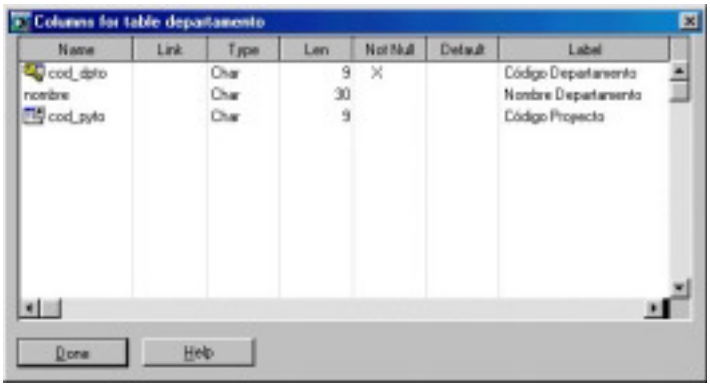


Figura 6.4. Columnas de las tablas Departamento y Proyecto



4. Crear la base de datos.

Se crea la base de datos a partir del repositorio siguiendo los pasos dados en los otros capítulos. Primero se selecciona dentro del menú del Editor de Repositorio la opción **SQL** y dentro de ella se selecciona **Database Options**. Se muestra un cuadro de diálogo donde se selecciona la conexión definida y la base de datos asociada a esta conexión. (Figura 6.5)

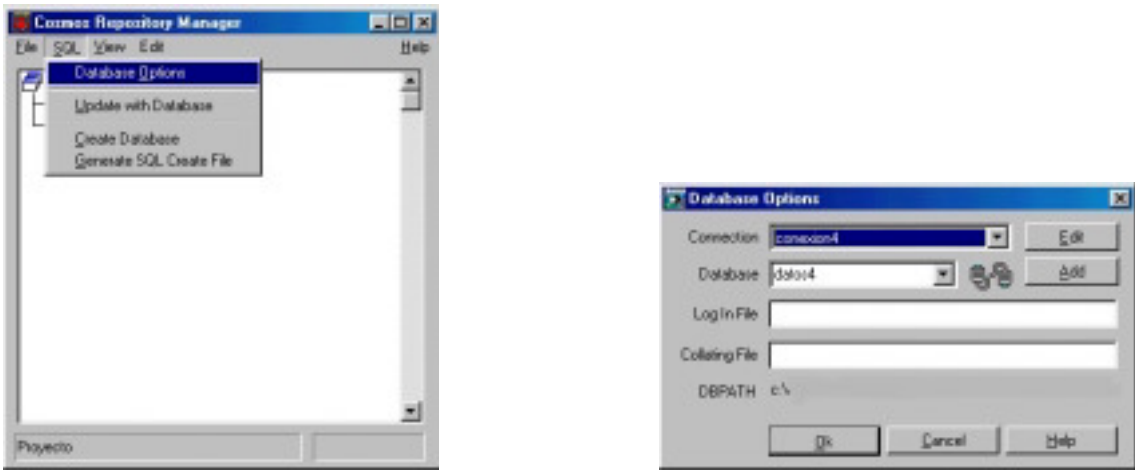


Figura 6.5. Seleccionar la conexión y la base de datos.

Una vez definidas las opciones de la base de datos se selecciona la **opción Create Database** dentro del mismo menú **SQL**. Una vez se crea correctamente la base de datos se muestra un cuadro de diálogo. (Figura 6.6)

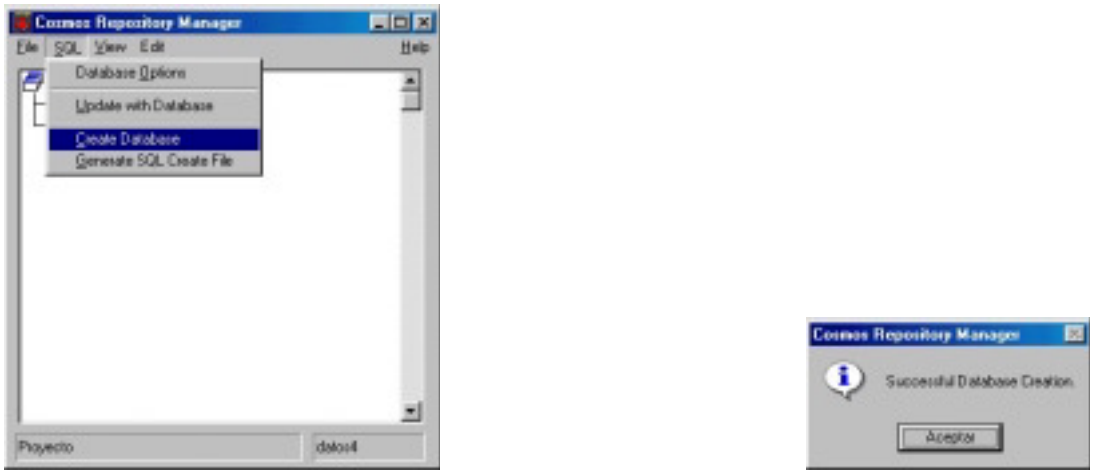


Figura 6.6. Creación de la base de datos.

5. Creación de un fichero SQL.

Una vez se tiene definido totalmente el repositorio, se puede hacer uso de otra de las utilidades del Editor de Repositorio de Cosmos. Se selecciona la opción **Generate SQL Create File** del menú **SQL** para generar un fichero SQL con la definición del repositorio. Se solicita el tipo de servidor con el que se utilizará el fichero generado. Se selecciona MultiBase. (Figura 6.7)

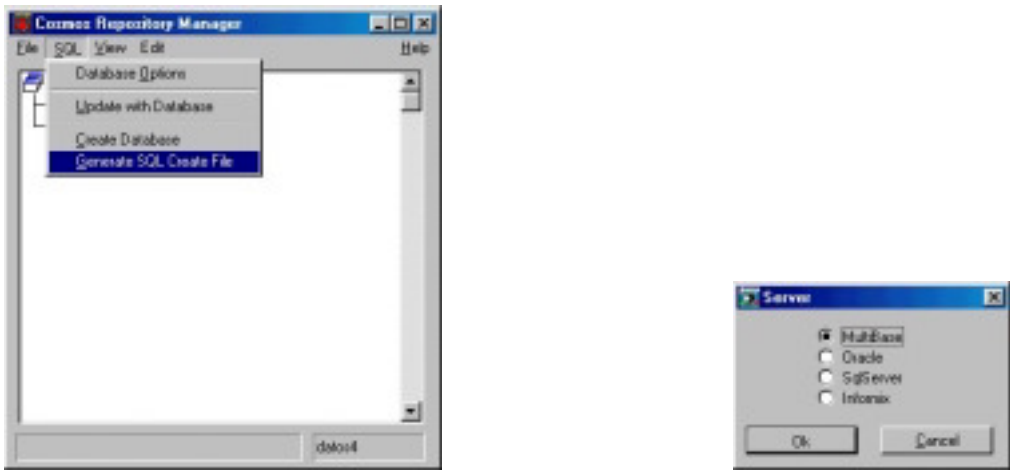


Figura 6.7. Solicitar la creación de un fichero SQL con el servidor MultiBase.

Se solicita a continuación la ruta y el nombre para el fichero que será generado. El resultado de la generación del fichero se muestra en un cuadro de diálogo, cuyo contenido se muestra a continuación.

```
{
    REPOSITORY: Repositorio4
    LOCATION: C:\Programa4\Repositorio4.crf
    DATE: 07/02/99   TIME: 21:04:52.
}
CREATE DATABASE datos4;
{
    TABLE: departamento
    Label: Departamento
}
CREATE TABLE departamento (
    cod_dpto  CHAR(9)      NOT NULL LABEL "Código Departamento",
    nombre    CHAR(30)     LABEL "Nombre Departamento",
    cod_pyto  CHAR(9)      LABEL "Código Proyecto"
    )
PRIMARY KEY (cod_dpto);
{
    TABLE: proyecto
    Label: Proyecto
}
CREATE TABLE proyecto (
    cod_pyto    CHAR(9)      NOT NULL LABEL "Código Proyecto",
    nombre      CHAR(30)     LABEL "Nombre Proyecto",
    cod_dpto    CHAR(9)      LABEL "Código Departamento"
    )
PRIMARY KEY (cod_pyto);
ALTER TABLE departamento
FOREIGN KEY cr_pyto (cod_pyto)
REFERENCES proyecto
    ON UPDATE RESTRICT
    ON DELETE RESTRICT;
ALTER TABLE proyecto
FOREIGN KEY cr_dpto (cod_dpto)
REFERENCES departamento
    ON UPDATE RESTRICT
    ON DELETE RESTRICT;
CLOSE DATABASE;
```

6. Crear el proyecto.

Se va a crear la aplicación mediante el Wizard de módulos de forma que se pueda seleccionar cada uno de los campos que se desean aparezcan en las diferentes pantallas e informes.

En el Editor Visual de Cosmos se selecciona la opcion **New** del menu **File** o se pulsa el icono  . Se elige la opción **Project Document** en el cuadro de diálogo, y se introduce a continuación el nombre del nuevo Proyecto y la ruta. (Figura 6.8)

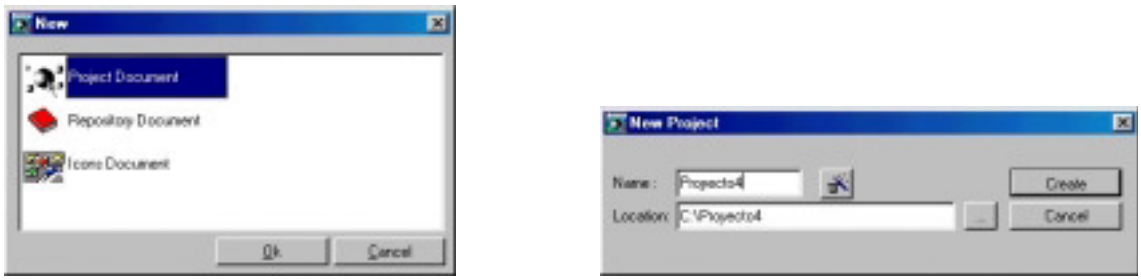


Figura 6.8. Crear un nuevo proyecto.

Después de pulsar el botón 

se muestra la paleta del nuevo proyecto.

El primer paso es asociar el repositorio que se ha creado al nuevo proyecto, para ello se selecciona la sección Repositories en la paleta del proyecto y se pulsa el botón secundario del ratón, se elige a continuación la opción Import.

En el cuadro de diálogo que se muestra a continuación se introduce la etiqueta para el repositorio y se selecciona el repositorio que se quiere asociar al proyecto. (Figura 6.9)

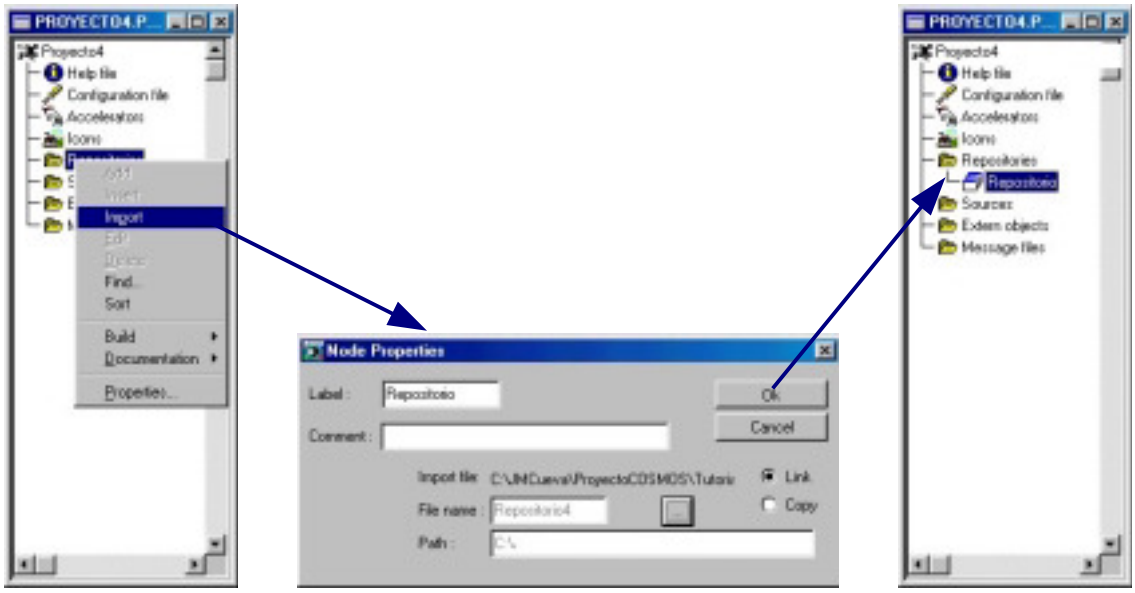


Figura 6.9. Asociar un repositorio al proyecto.

El siguiente paso es crear las carpetas de forma similar a las creadas en el capítulo 5. Se crearán tres carpetas: Mantenimiento, Maestro-Detalle e informes. El resultado se muestra en la figura 6.10.

Figura 6.10. Carpetas para los diferentes módulos.

7. Crear los módulos de mantenimiento.

En esta carpeta se crearán dos módulos para realizar el mantenimiento de las dos tablas del repositorio. El proceso para crear ambos módulos es similar al realizado en el capítulo primero mediante el Wizard de módulos. La paleta del módulo de mantenimiento de la tabla departamento, el aspecto de la Screen de la clase creada para ese módulo y la nueva paleta del proyecto se muestran en la figura 6.11.

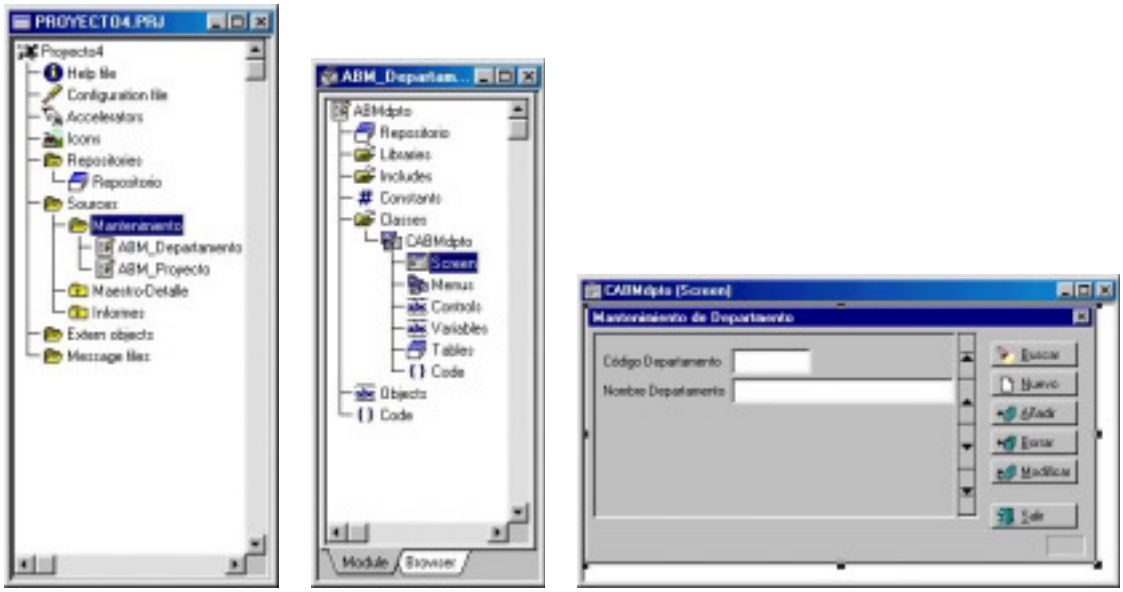


Figura 6.11. Paletas del proyecto y el módulo ABM_Departamento y aspecto de la Screen.

8. Crear los módulos de Maestro-detalle.

Para crear los módulos de maestro-detalle se utilizará el Wizard de módulos y se seguirán los pasos dados en el capítulo quinto en el punto 10.

En la figura 6.12. se muestra el aspecto de la paleta del proyecto con los nuevos módulos. También se muestra la paleta del módulo denominado “md_departamento_proyecto” así como la sección screen del mismo.

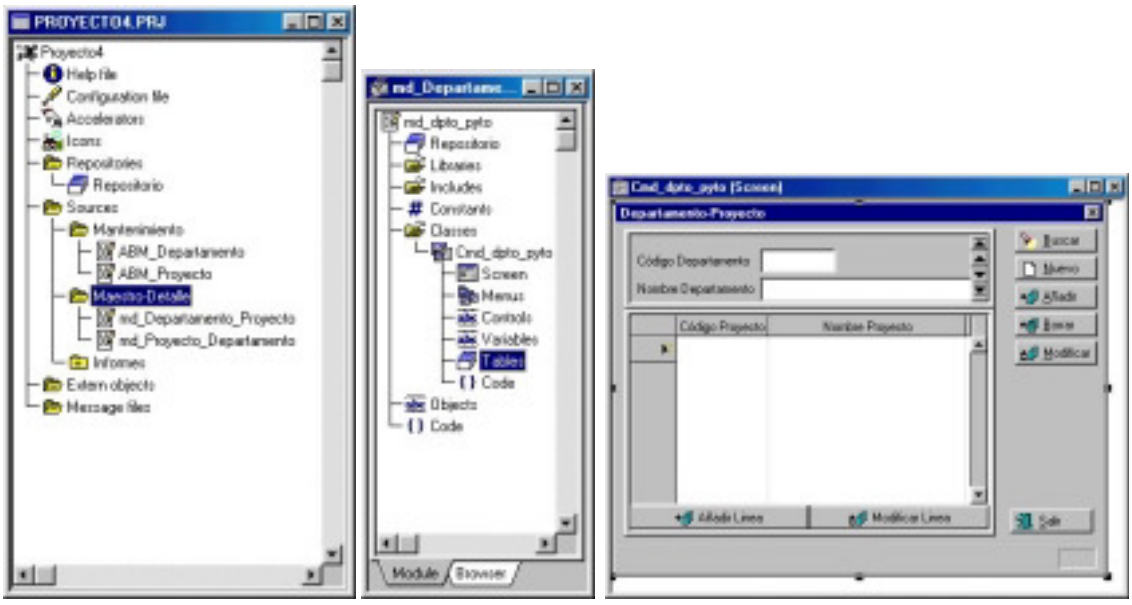


Figura 6.12. Paleta del proyecto y del módulo md_Departamento_Proyecto y la screen.

9. Crear los informes.

En esta aplicación se van a crear dos informes simples de las tablas. Se utilizará el Wizard de módulos para la creación de todos estos informes.

Crear el informe para la tabla Departamento y para la tabla Proyecto.

Los pasos que se siguen son similares en ambos casos por lo que únicamente se mostrará el proceso de creación del informe de la tabla Departamento.

Seleccionando la carpeta Informes se pulsa el botón secundario del ratón y se selecciona la opción Add. En el cuadro de diálogo que se muestra se debe introducir el nombre del nuevo módulo y la ubicación. Se pulsa a continuación el icono  para invocar al Wizard de módulos de Cosmos. Figura 6.13.

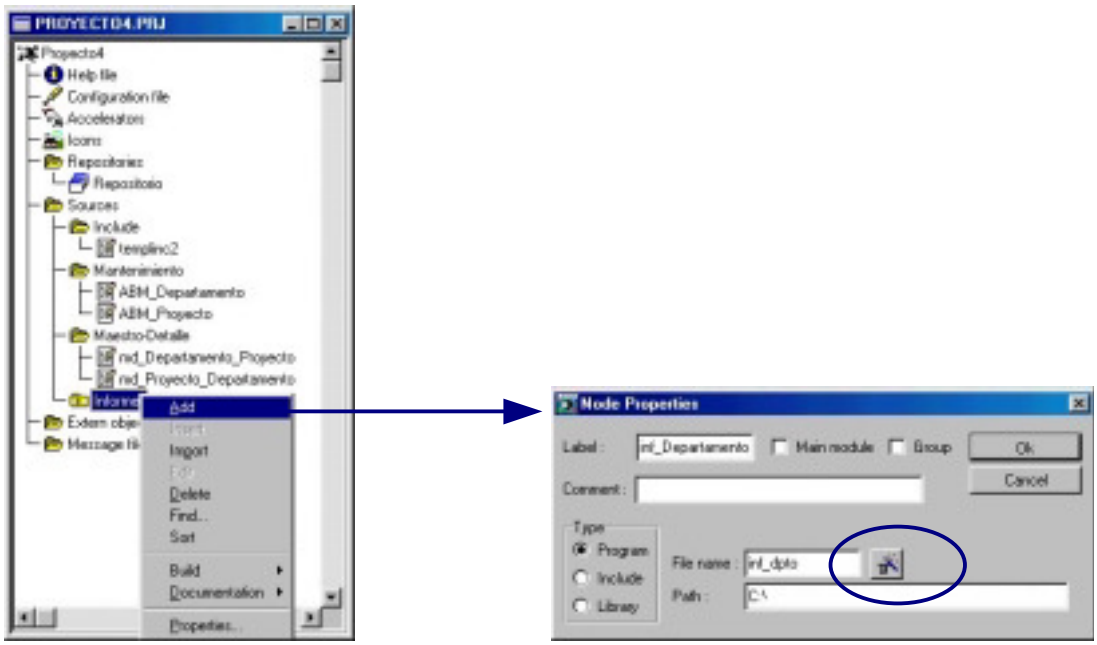
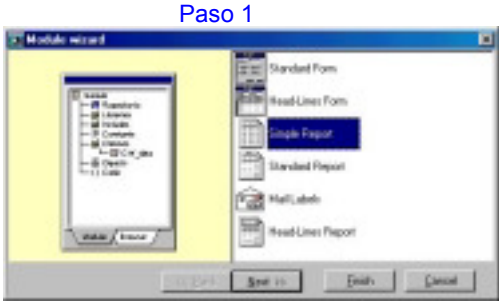


Figura 6.13. Añadir un nuevos módulo dentro de la carpeta informes.

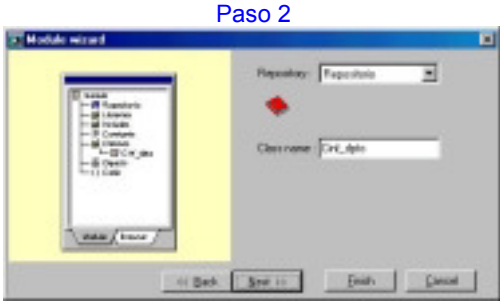
En la figura 6.14 se muestran los pasos que se siguen para la creación del informe mediante el Wizard.

En el paso 1 se selecciona el tipo de informe que se desea construir, en el paso 2 se selecciona el repositorio que se va a utilizar en el módulo y el nombre de la clase. En el paso 3 se selecciona el tipo de listado y en el paso 4 se elige la tabla del listado y las columnas de dicha tabla que se mostrarán en el informe. En el paso 5 se pide el título que se asignará al informe y en el paso 6 se establece la conexión con la base de datos.

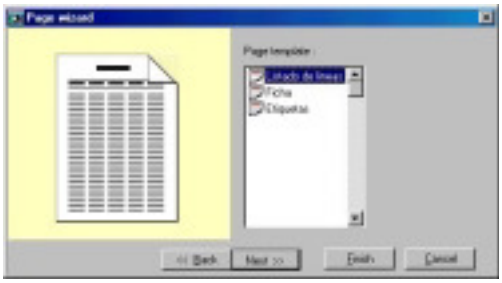
Paso 1



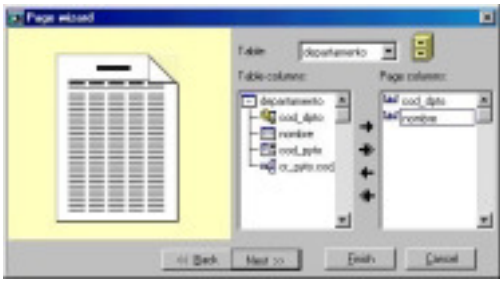
Paso 2



Paso 3



Paso 4



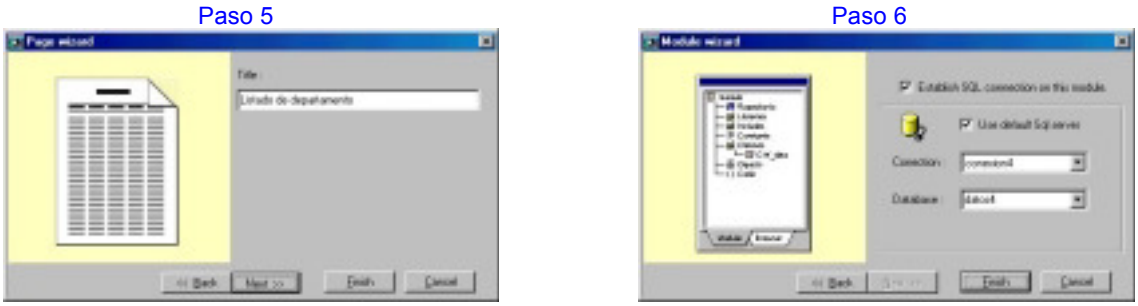


Figura 6.14. Pasos del Wizard para generar el informe de la tabla Departamento.

Después de terminar la definición del informe mediante el Wizard, la paleta del nuevo módulo y la sección **Templates** del mismo se muestra en la figura 6.15.

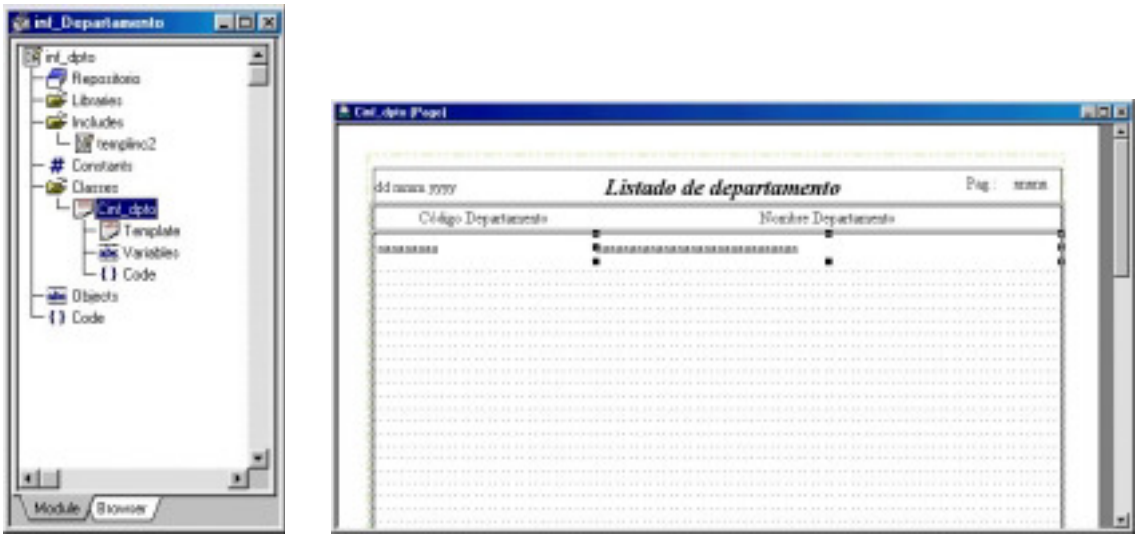


Figura 6.15. Módulo inf_Departamento y sección Templates.

De manera similar al capítulo 2, el Wizard ha añadido un include dentro del módulo llamado **templic2**. (Véase anexos)

Después de efectuar los mismos pasos para la creación del módulo del informe de la tabla Proyecto, la paleta del proyecto se muestra en la figura 6.16.

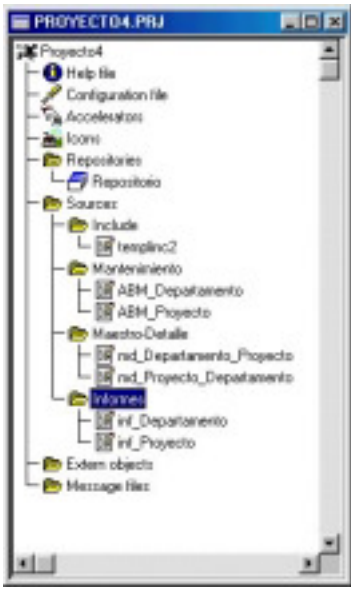


Figura 6.16. Paleta del proyecto actual.

10. Generar la documentación de la aplicación.

En el archivo “Repositorio4.DFR” se ha guardado la documentación que se escribió en la definición de las diferentes columnas de las tablas del repositorio. El contenido de este archivo se muestra a continuación.

```
[TABLE departamento]
Definición de la tabla departamento

[TABLE proyecto]
Definición de la tabla proyecto

[COLUMN departamento.cod_dpto]
Código del departamento
char(9)
[COLUMN departamento.nombre]
Nombre del departamento
char(30)

[COLUMN proyecto.cod_pyto]
Código del proyecto
char(9)
[COLUMN proyecto.nombre]
Nombre del proyecto
char(30)
```

En el Editor Visual de Cosmos existe una opción en el menú denominada **Docu** que permite generar la documentación del código y tablas de la aplicación en formato HTML.

Dentro de esta opción lo primero que se hace es establecer las diferentes opciones que se desea cumpla la documentación del proyecto. (Figura 6.17)

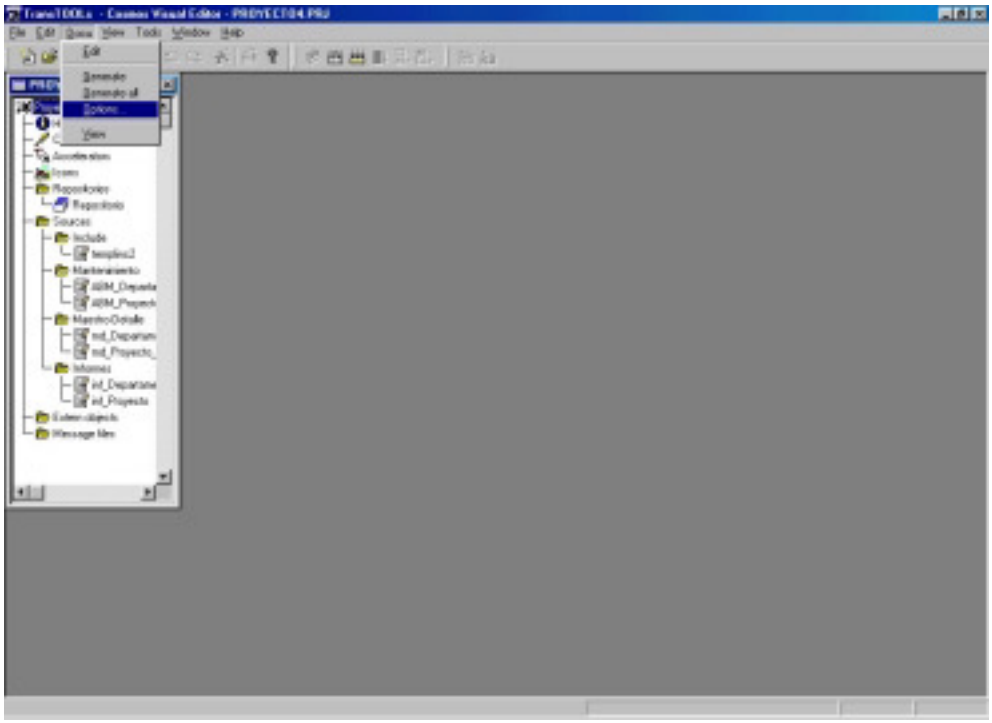


Figura 6.17. Seleccionar Options del menú Docu del Editor Visual.

Se muestra un cuadro de diálogo que permite establecer estas opciones, se han mantenido las opciones establecidas por defecto. (Figura 6.18)

Figura 6.18. Opciones de la documentación.

Después de pulsar el botón 

, se selecciona del mismo menú Docu la opción Generate All que generará la documentación completa del proyecto. Se genera en formato HTML y se puede visualizar mediante un navegador.

En la figura 6.19 se muestra el aspecto de la documentación generada. A partir de esta pantalla se puede visualizar todos los componentes de la aplicación.

Figura 6.19. Aspecto de la documentación.

Si se hace doble click de ratón sobre Repositorio4 se muestran las tablas que componen el repositorio. Haciendo doble click sobre cada una de ellas se muestran las columnas que tiene la tabla, así como sus características. En la figura 6.20 se ve ambas pantallas, una sobre otra.

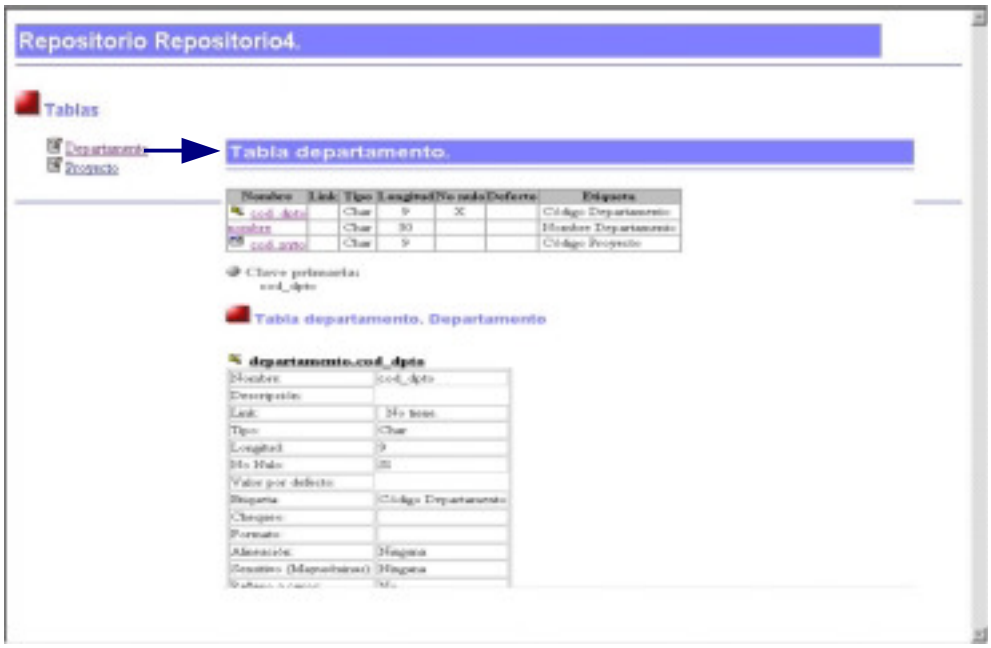


Figura 6.20. Aspecto de la documentación generada sobre la tabla departamento.

Si se selecciona ABMdpto dentro de la sección Módulos de la figura 6.19. se muestra la información sobre este módulo. (Figura 6.21)

Figura 6.21. Información del módulo ABMdpto.

11. Archivos generados en este capítulo.

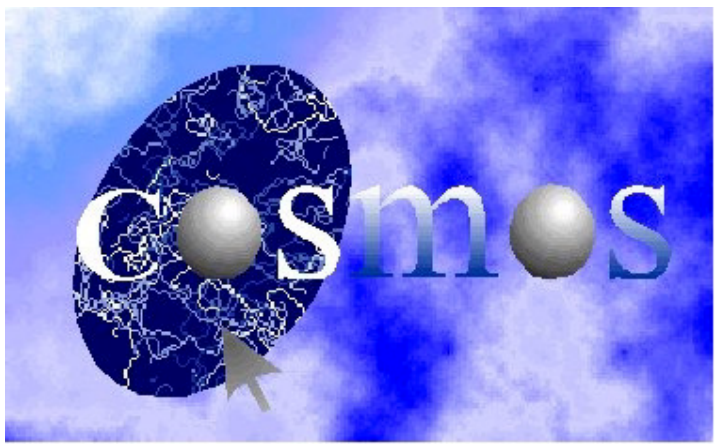
En este capítulo se han generado los siguientes archivos nuevos dentro de la carpeta "Proyecto4". Unicamente se reflejan los archivos de extensión .smd y por cada uno de ellos existirá un archivo del mismo nombre y extensión .omd que son los que contienen el fichero objeto.

Archivo	Contenido
Proyecto4.prj	Fichero que contiene el código global de la aplicación.
ABM_dpto.smd	Fichero ASCII que contiene el código fuente del módulo de mantenimiento de la tabla departamento.
ABM_pytos.smd	Fichero ASCII que contiene el código fuente del módulo de mantenimiento de la tabla proyectos.
Md_Dpto_pytos.smd	Fichero ASCII que contiene el código fuente del módulo de mantenimiento de la tabla de departamentos y proyectos mediante el maestro-detalle..
Md_Pytos_dpto.smd	Fichero ASCII que contiene el código fuente del módulo de mantenimiento de la tabla de proyectos y departamento mediante el maestro-detalle..
Inf_dpto.smd	Fichero ASCII que contiene el código fuente del módulo del informe de la tabla departamento.
Inf_pyto.smd	Fichero ASCII que contiene el código fuente del módulo del informe de la tabla proyecto.

Capítulo 7

Uso de la Clase SqlCursor de MultiBase COSMOS para la realización de Consultas

1. Introducción.
2. La clase SqlCursor del COOL.
3. Creación del módulo mCargos.
4. Creación de la clase cCargos.
5. Asociar las tablas al módulo mCargos.
6. Diseño de la interfaz de la consulta.
7. Código de la clase cCargos.
8. Ejecución de la nueva consulta.
9. Archivos generados en este capítulo.



1. Introducción.

En este capítulo se realizará una nueva versión de la aplicación creada en el capítulo 5 para la realización de una nueva consulta a los datos de la base de datos. Se utilizará para mostrar el uso de la clase SqlCursor del lenguaje COOL.

2. La clase SqlCursor del COOL.

Esta clase derivada de la clase Complex es derivable e instanciable permite declarar y ejecutar cursores del SQL sobre una base de datos determinada. Un cursor es una tabla virtual asociada al programa que lo define, cuyo contenido será el resultado de una instrucción SELECT utilizada en su definición.

Cuando en un programa se emplea una instrucción SELECT para realizar una consulta («query»), pueden ocurrir dos cosas: que el resultado (es decir, la «tabla derivada») incluya una única fila de la base de datos, o bien que incluya varias filas. En este último caso es donde se hace necesaria la utilización de un cursor, en el que el SQL incluirá todas las filas resultado de la instrucción SELECT. Evidentemente, también se dispone de los métodos necesarios para poder moverse a través de las distintas filas del cursor.

Los métodos de la clase SqlCursor son:

Método	Utilidad
AddGroup	Este método se utiliza para definir de uno en uno los agrupamientos en una instrucción SELECT previamente preparada.
AddTotal	Este método se utiliza para definir de una en una las columnas a totalizar en los grupos.
AttachServer	Asocia un servidor a la instrucción SQL, cuando se ejecute el cursor se hará contra el servidor asociado.
BreakLevel	Este método se utiliza para conocer el estado de la ruptura de grupos en un momento determinado. Se utiliza después de la ejecución del método Fetch.
Close	Este método se utiliza para cerrar un cursor abierto previamente por medio del método Open.
Error	Indica el código del último error producido en el servidor.
ErrMsg	Muestra el string asociado al último error producido en el servidor.
Execute	Ejecuta el cursor previamente preparado con el método Prepare.
Fetch	Permite hacer desplazamientos secuenciales dentro de un cursor.
FetchFirst	Este método lee la primera fila de la tabla derivada devuelta por un cursor abierto previamente por medio de una instrucción Open.
FetchLast	Este método lee la última fila de la tabla derivada devuelta por un cursor abierto previamente por medio de una instrucción Open.
FetchNext	Este método lee la fila siguiente a la fila en curso de la tabla derivada devuelta por un cursor abierto previamente por medio de una instrucción Open.
FetchPrev	Este método lee la fila anterior a la fila en curso de la tabla derivada devuelta por un cursor abierto previamente por medio de una instrucción Open.
FetchTuple	Permite hacer desplazamientos secuenciales dentro de un cursor. Este método lee una tupla de la tabla derivada devuelta por un cursor abierto previamente por medio del método Open. Devuelve la tupla en un Char con las columnas separadas por un delimitador.
Found	Indica si el cursor a procesado alguna fila o ninguna.
Free	Libera la memoria asignada por el cursor para la sentencia en curso.
GroupAverage	Este método se utiliza para obtener la media aritmética de todos los elementos que tiene un grupo hasta el momento.
GroupBy	Este método se utiliza para definir una lista de agrupamientos en una instrucción SELECT previamente preparada.

Método	Utilidad
GroupCount	Este método se utiliza para obtener el número de filas que tiene un grupo hasta el momento.
GroupMaximun	Este método se utiliza para obtener el máximo de todos los elementos que tiene un grupo hasta el momento.
GroupMinimun	Este método se utiliza para obtener el mínimo de todos los elementos que tiene un grupo hasta el momento.
GroupSum	Este método se utiliza para obtener la suma de todos los elementos que tiene un grupo hasta el momento.
Into	Permite indicar la lista de variables globales del programa que se utilizarán para recoger los valores de las columnas del cursor.
Locked	Este método devuelve TRUE si la fila en curso de un cursor esta bloqueada por otro usuario. En caso contrario su valor será FALSE.
Open	Este método abre un cursor previamente declarado con el método prepare, es decir, ejecuta la instrucción SELECT asociada al cursor.
Prepare	Este método permite construir una instrucción de tipo SELECT del SQL de forma dinámica.
Putname	Este método permite dar un nombre al curso para que el CTSQL pueda identificar el cursor y ejecute la instrucción update sin errores.
ScrollFetch	Permite hacer desplazamientos relativos dentro de un cursor. Este método lee una fila de la tabla derivada devuelta por un cursor abierto previamente por medio de una instrucción Open.
Totalize	Este método se utiliza para definir la lista de columnas que se desean totalizar en los grupos.

3. Creación del módulo mCargos.

Vamos a realizar una nueva opción dentro del menú existente en la aplicación del capítulo 5. Para ello se selecciona dentro de la paleta del proyecto el módulo Inicio y dentro de la paleta de este módulo se elige "menuPpal" dentro de la sección Menus. (Figura 7.1)

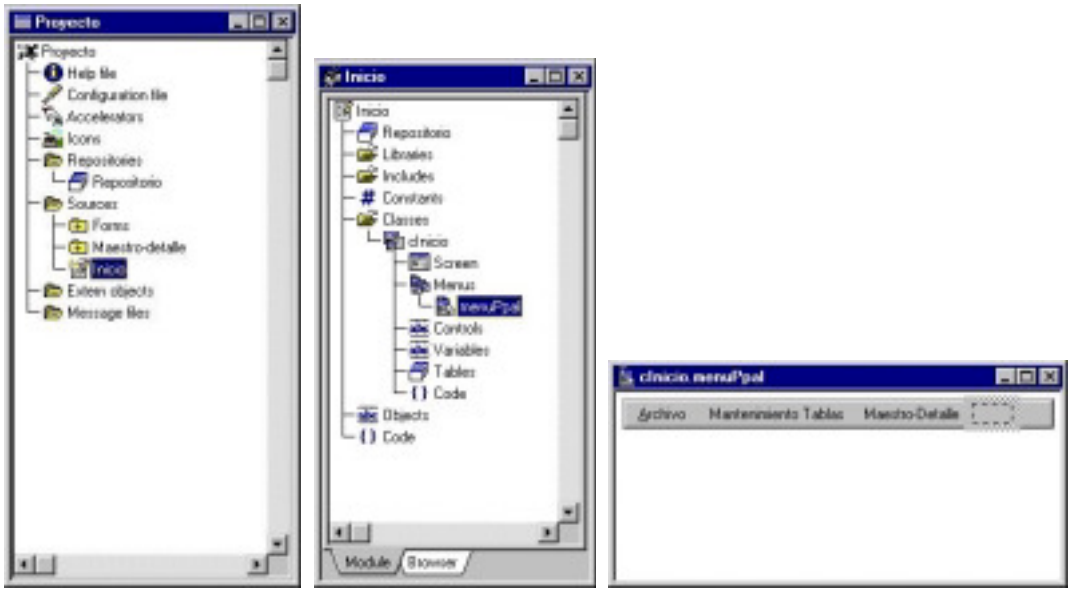


Figura 7.1.Selección del menuPpal dentro del módulo Inicio.

Se añade una nueva opción de menú desplegable que englobe todas las consultas que se van realizar. Dentro de esta opción se crea una nueva consulta llamada "Cargos de un proyecto". (Figura 7.2)

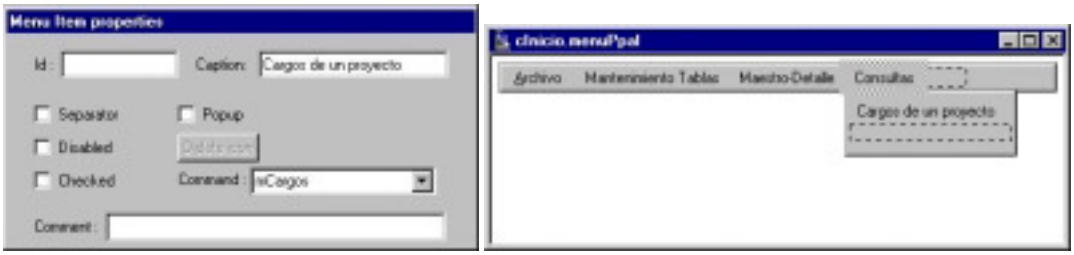


Figura 7.2. Nueva opción de menú.

Cuando se seleccione esta nueva opción se ejecutará un nuevo módulo que se va a llamar “mCargos”. Este nuevo módulo se va a crear en un nuevo grupo denominado “Consultas” dentro de la paleta del proyecto. (Figura 7.3)

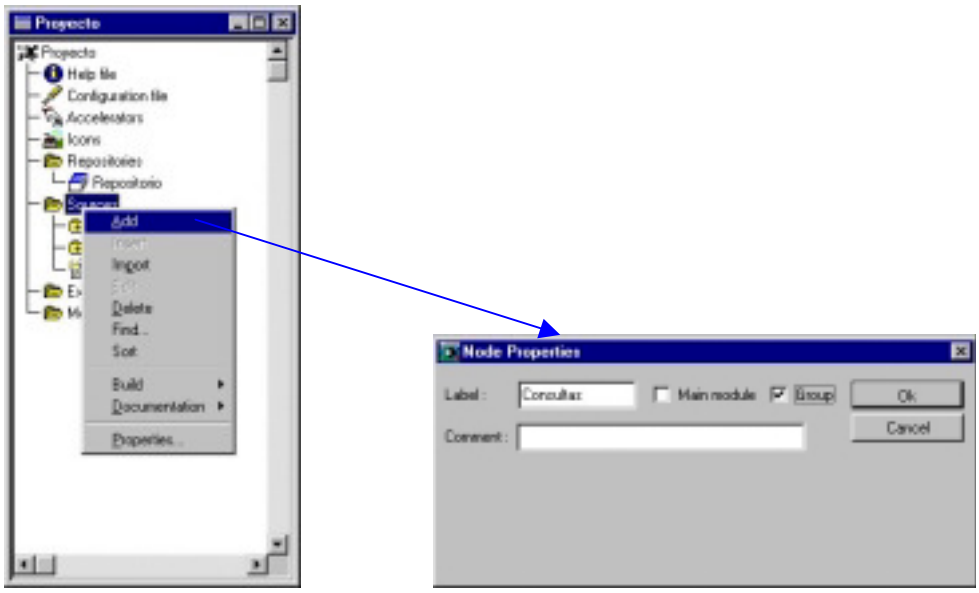


Figura 7.3. Creación del grupo de Consultas.

Dentro de este nuevo grupo se introduce un nuevo módulo que donde se va a definir la consulta de los cargos de un determinado proyecto. (Figura 7.4)

Figura 7.4. Creación del nuevo módulo dentro del grupo Consultas.

Al hacer doble click de ratón sobre este nuevo módulo se muestra la paleta del mismo.

4. Creación de la clase cCargos.

Se añade una nueva clase dentro de este nuevo módulo que se llamará “cCargos”. Esta nueva clase deriva de la clase predefinida de Cosmos Form. (Figura 7.5)

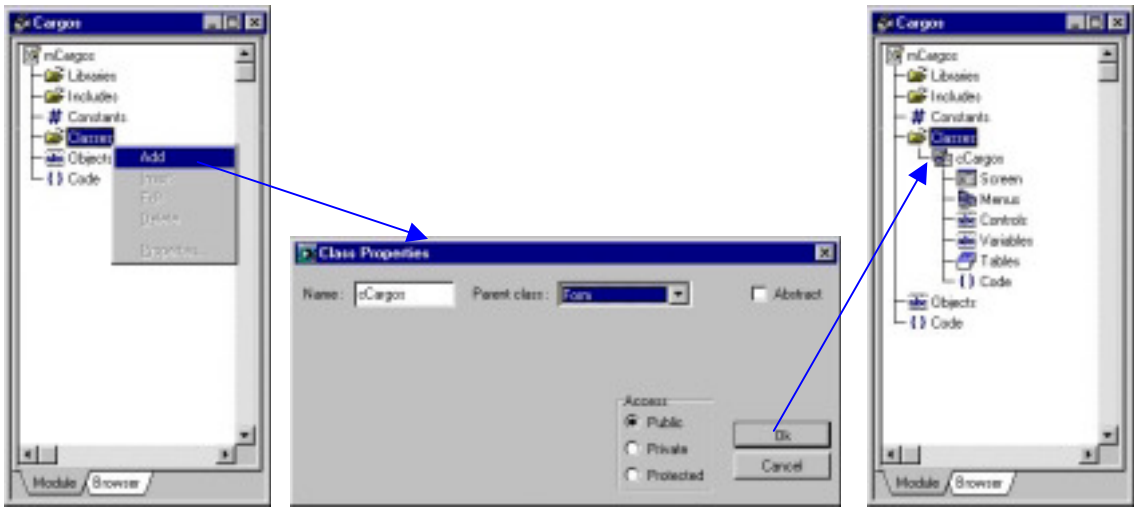


Figura 7.5. Creación de la nueva clase cCargos.

Dentro de la sección Objects del módulo Cargos se va a crear un objeto de esta nueva clase. (Figura 7.6)

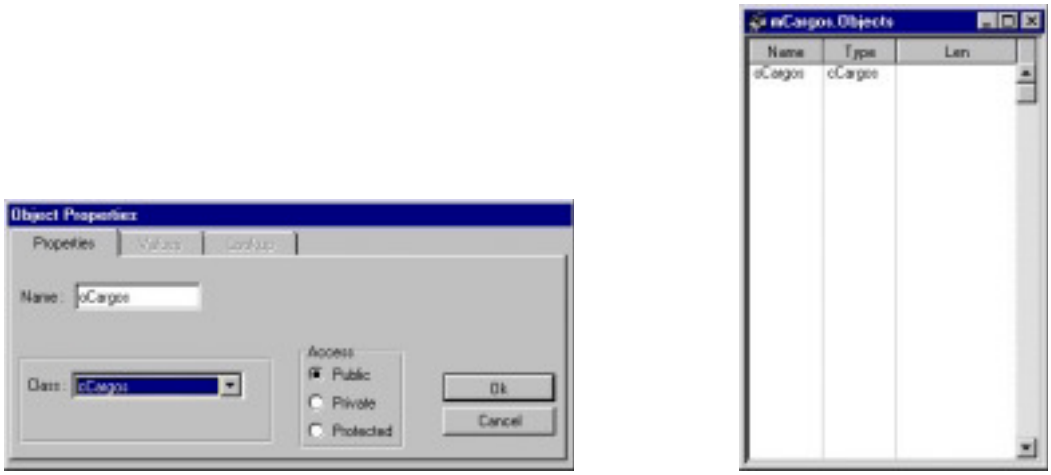


Figura 7.6. Definición del nuevo objeto.

Se añade en la sección Code de este módulo el código necesario para establecer la conexión con la base de datos y la invocación del método Run de la clase Form sobre el objeto creado anteriormente. (Figura 7.7)

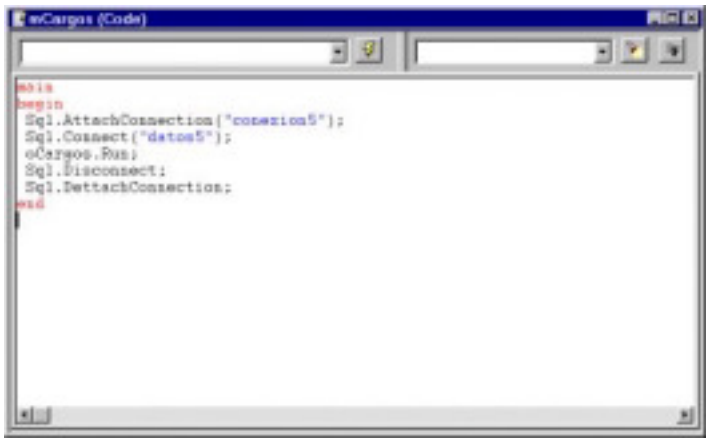


Figura 7.7. Sección código del módulo Cargos.

5. Asociar las tablas al módulo mCargos.

El siguiente paso es asociar el repositorio correspondiente al módulo. En la paleta del proyecto se hace doble click sobre el Repositorio, de esta forma se muestra las tablas que forman parte del mismo.

Se selecciona la tabla “proyectos” de la paleta del Repositorio y se arrastra sobre la sección Tables de la paleta del módulo mCargos. Se procede de forma similar con la tabla “cargos”. Efectuando la asociación de las tablas del Repositorio al módulo de esta forma el repositorio queda asociado al módulo automáticamente. (Figura 7.8)

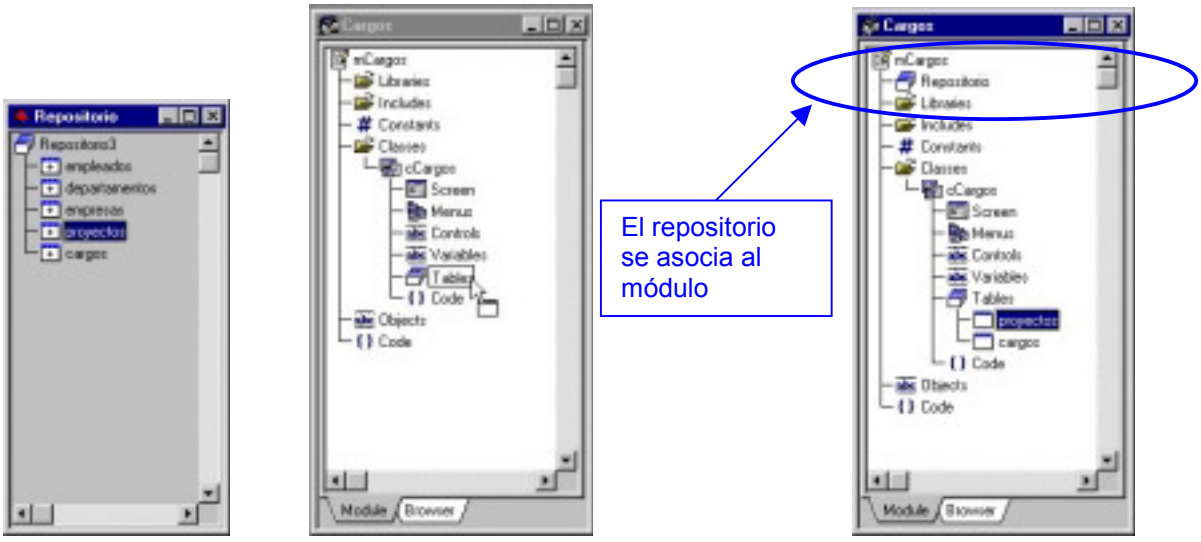


Figura 7.8. Asociar un repositorio a un módulo.

6. Diseño de la interfaz de la consulta.

Se hace doble click sobre la sección Screen de la clase cCargos que se creó en el apartado 4 de este capítulo para poder comenzar el diseño de la interfaz de la consulta. Se muestra el estado actual de la pantalla asociada a la clase. (Figura 7.9)

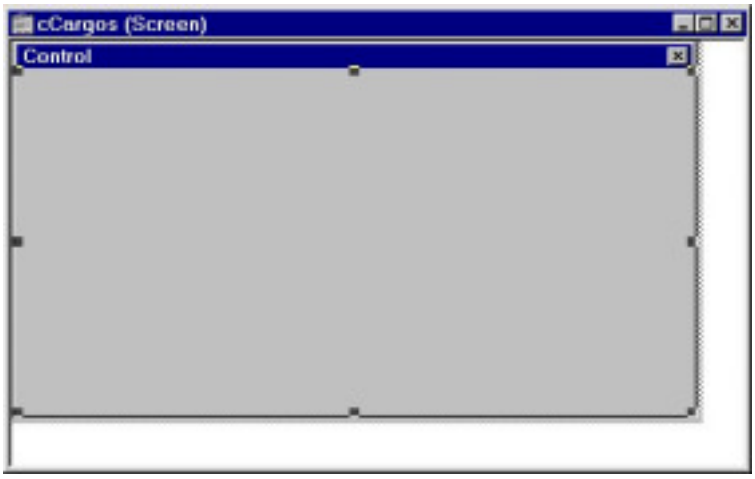


Figura 7.9. Aspecto de la interfaz de la clase cCargos.

Vamos a modificar el título de esta Screen. Para ello se hace doble click sobre el componente Frame de la Screen o se pulsa el botón secundario del ratón sobre la misma seleccionando a continuación la opción Properties del menú que se muestra. Dentro de las propiedades modificamos el campo Label. (Figura 7.10)

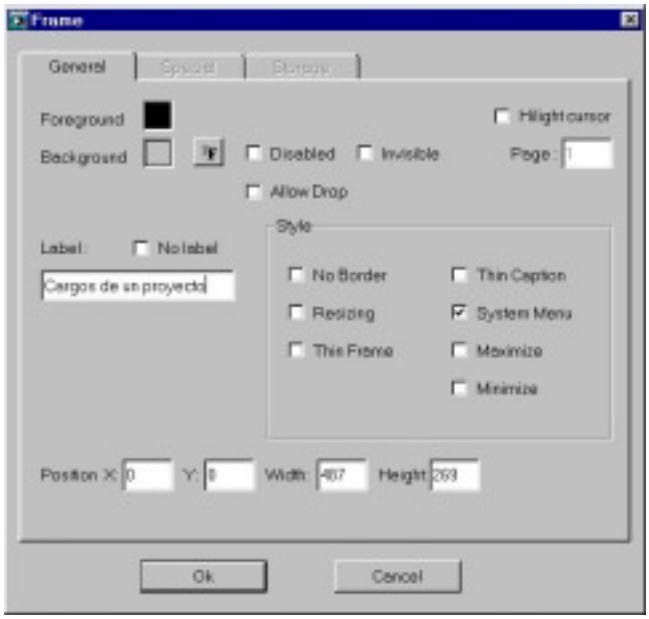


Figura 7.10. Modificación del título de la Screen.

Se va a añadir un control de tipo Box que se comportará como contenedor de los datos de la tabla “Proyectos”. Se selecciona dentro de la paleta de controles, que se muestra en la parte derecha del Editor Visual de COSMOS, el control Box. Se arrastra este control sobre la Screen y se estira para que ocupe el tamaño requerido. (Figura 7.11)

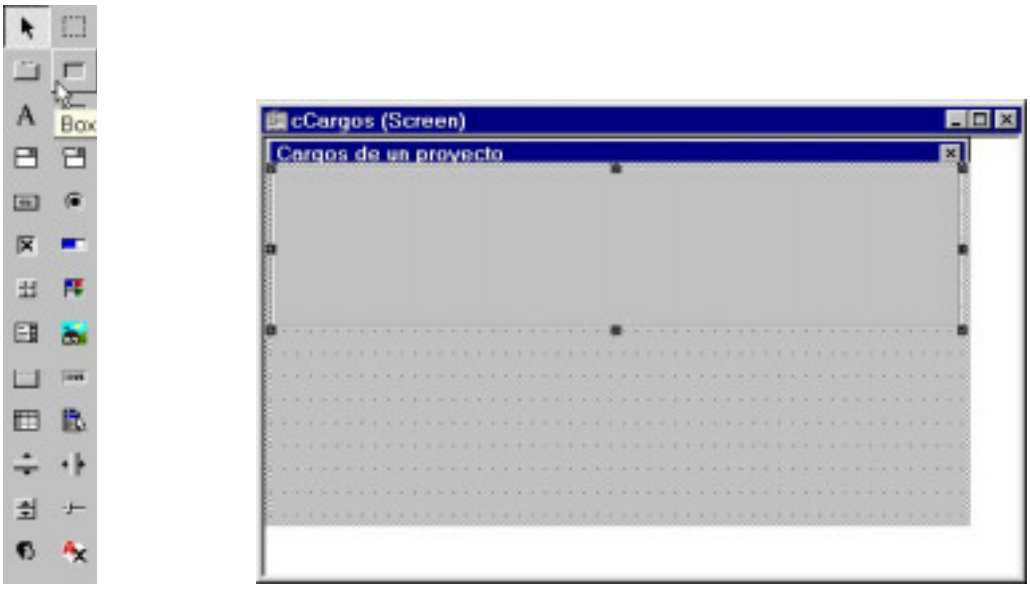


Figura 7.11. Arrastrar el control de tipo Box sobre la Screen.

Se hace doble click sobre el nuevo elemento añadido para mostrar sus propiedades, o también pulsando el botón secundario del ratón y seleccionando la opción Properties del menú que se muestra. Dentro de estas propiedades se va a realizar modificaciones en la pestaña Storage de este control. Seleccionamos la opción “Form Table” para vincular el control con la tabla Proyectos que se selecciona de la lista desplegable. (Figura 7.12)

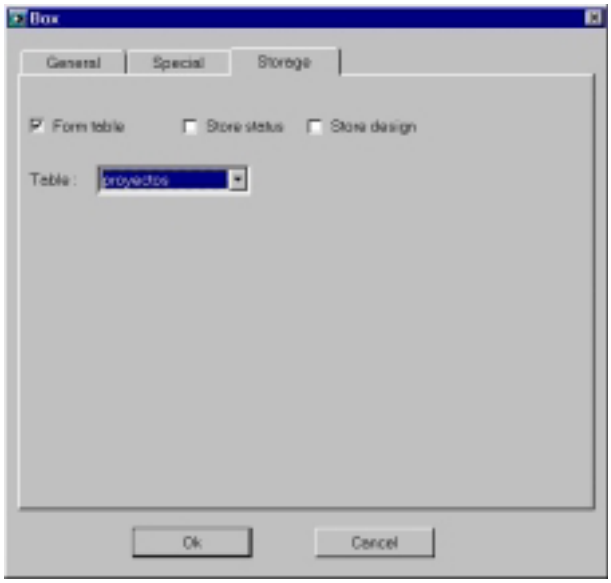


Figura 7.12. Vincular la tabla Proyectos al control de tipo Box.

Después de realizar este vínculo podemos definir dentro de este Box las columnas de la tabla que se desean mostrar. Seleccionamos la tabla “Proyectos” de la sección Tables de la clase cCargos dentro de la paleta del módulo mCargos. Haciendo doble click sobre esta tabla se muestran las columnas que contiene. Seleccionamos las columnas de “cod_pyto”, “titulo” y “presupuesto” y las arrastramos sobre el control Box. Se pide confirmación para añadir las etiquetas que se asignaron en la definición del repositorio a cada una de las columnas. Se responde que si, aunque se modifican para que se correspondan mas adecuadamente a la consulta en curso. (Figura 7.13)

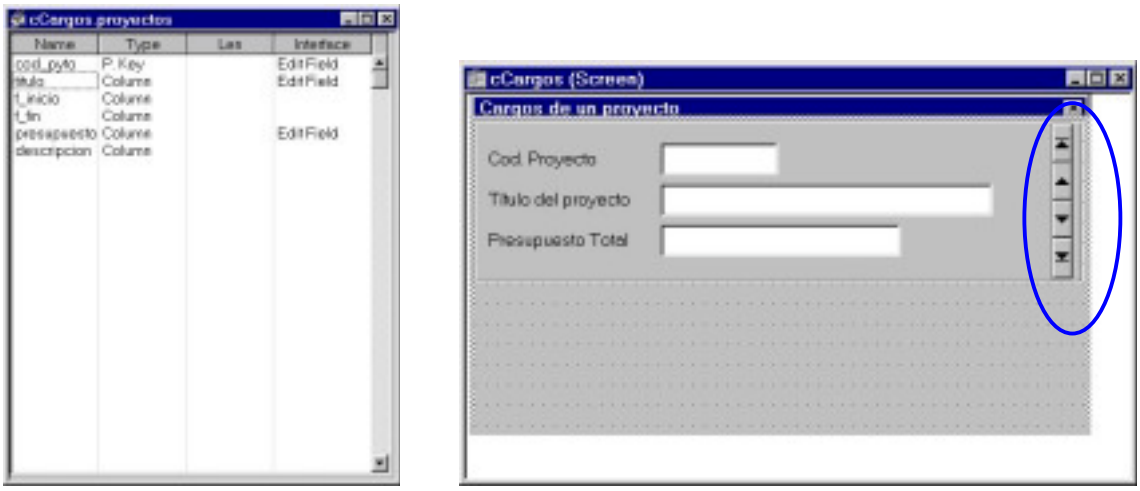


Figura 7.13. Añadir las columnas a la Screen.

En la parte derecha del control Box se ha añadido una barra de navegación. Esta barra de navegación permitirá recorrer los datos de las diferentes tuplas que componen la tabla “Proyectos”. Este control se ha seleccionado de la paleta de controles y se ha arrastrado sobre el Box.

Se editan sus propiedades haciendo doble click de ratón sobre él y se modifica de la pestaña “General” el identificador del control, añadiendo el nombre “navegador”.

Dentro de la pestaña “Special” se mantiene la definición vertical, se selecciona el número de botones que va a tener y se debe mantener sin marcar la opción “Send command”. Este último dato es muy importante, porque se va a definir la funcionalidad de este control desde el código de la clase cCargos y no de forma automática por Cosmos. (Figura 7.14)

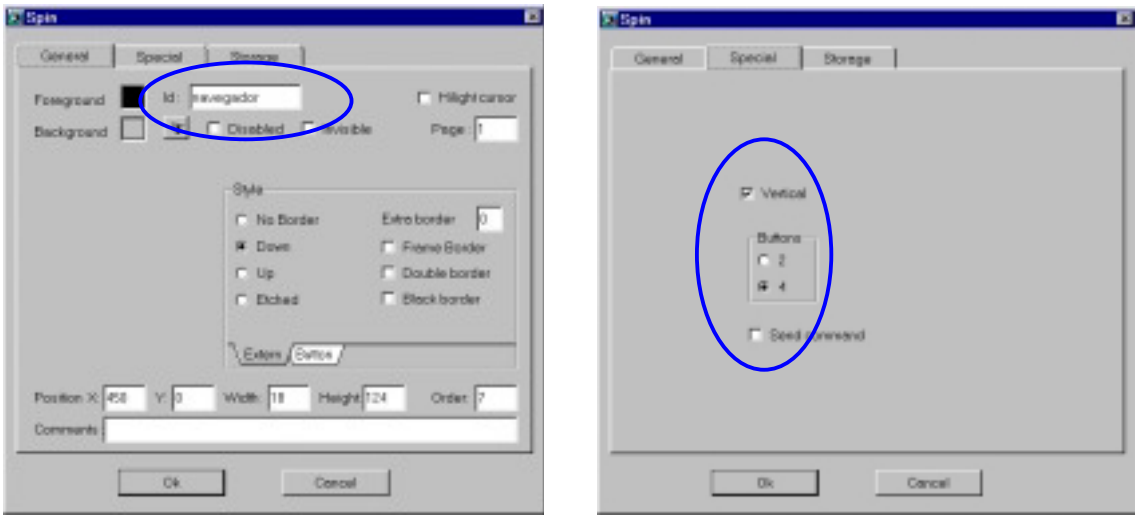


Figura 7.14. Modificación de las propiedades de la barra de navegación.

La siguiente acción a realizar es añadir las etiquetas y los campos de edición necesarios para mostrar el resultado de la consulta realizada mediante la clase SqlCursor respecto al proyecto que se muestre en cada momento en la parte superior de la Screen. Son necesarias tres campos de edición que contendrán la cuantía de los cargos efectuados hasta el momento al proyecto, el número total de cargos y el dinero que se encuentra pendiente de asignación. El aspecto definitivo de la Screen de la clase cCargos se muestra en la figura 7.15.

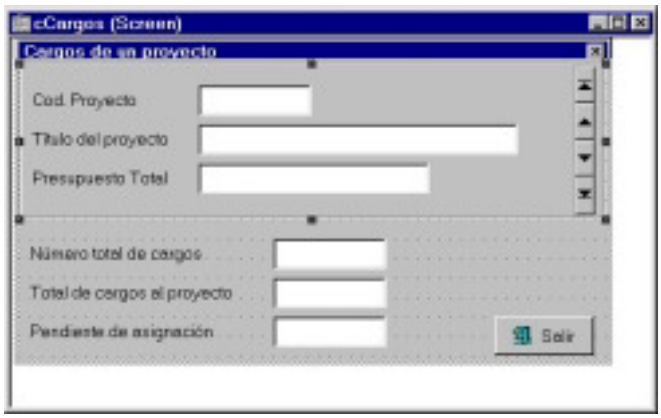


Figura 7.15. Aspecto definitivo de la Screen de la clase cCargos.

El botón “Salir” se ha añadido seleccionando el control de tipo Push Button de la paleta de controles y arrastrando el mismo sobre la Screen. Este botón enviará un comando de cierre de la Screen. Por esto en las propiedades de la pestaña “Special” se ha seleccionado el comando “Close” de la lista desplegable. (Figura 7.16)

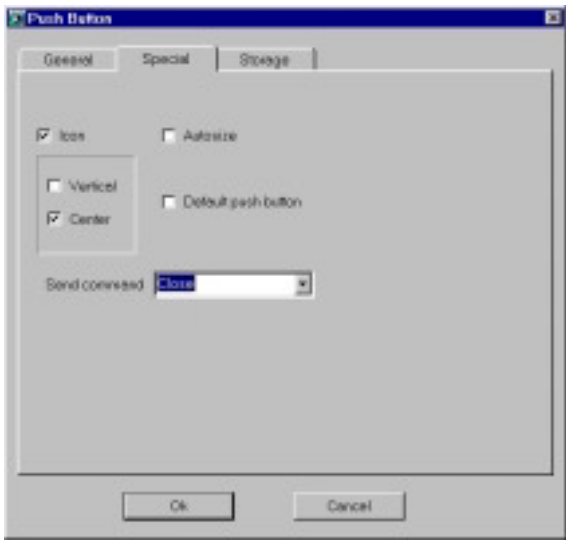


Figura 7.16. Propiedades del control de tipo Push Button.

A los seis campos de edición se les ha añadido un identificador para poder hacer referencia a cada uno de ellos en el código de la clase cCargos. Estos nombres son secuenciales comenzando desde el primero de la siguiente forma: edit0, edit1,..., edit5. En dos de ellos se ha modificado el color de las fuentes para destacar los resultados de la consulta. Esto se realiza seleccionando dentro de las propiedades del control, en la pestaña “General”, la propiedad “Foreground”, si se hace doble click de ratón sobre el cuadro de color negro se muestra una paleta de colores de la cual se selecciona la que deseemos. (Figura 7.17)

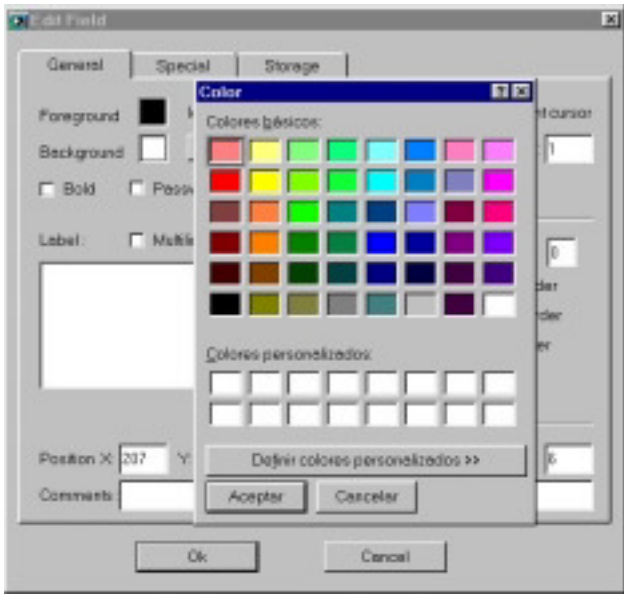


Figura 7.17. Cambio del color de las letras del campo de edición.

7. Código de la clase cCargos.

En este momento se puede escribir el código correspondiente a la clase que se acaba de definir. Para ello se hace doble click de ratón sobre la sección “Code” dentro de la paleta de la clase cCargos. Se muestra entonces una ventana donde se introduce el código. (Figura 7.18)

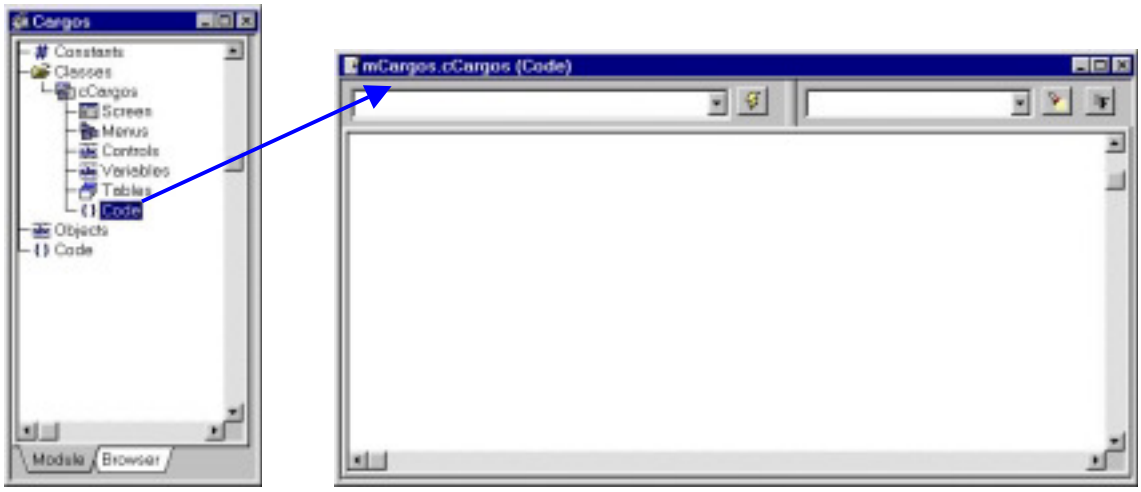


Figura 7.18. Apertura de la ventana del código de la clase cCargos.

La función donde se va a definir la consulta sobre la tabla cargos se ha denominado “bCalcular”. El código completo de esta función que se ha definido de acceso Privado, se muestra a continuación:

```
private function bCalcular
objects
begin
 //Definición de los objetos
 codigo as char
 scursor as SqlCursor
 presu cuantia total dif as money(15,2)
 contador as smallint
end
begin
 // Almacenar sobre el objeto codigo el contenido del campo de
 // edición edit0 que muestra el código del proyecto.
 codigo=edit0.Text;
 // Almacenar sobre el objeto presu el contenido del campo de
 // edición edit2
 presu=edit2.Text;
 // Inicializar a cero los objetos total y contador.
 // total contendrá la cuantía total de los cargos del proyecto.
 // contador contiene el número de cargos de un proyecto.
 total=0;
 contador=0;
 // Se invoca al método Prepare de la clase SqlCursor para
 // definir la consulta en SQL que se va ha realizar sobre la
 // tabla "Cargos". Esta consulta se puede leer como
 // "Seleccionar cuantia en la tabla Cargos donde la columna
 // cod_pyto tenga el valor ?"
 scursor.Prepare("Select cuantia from cargos where
                  cargos.cod_pyto=?");
 // Se invoca el método Open de la clase SqlCursor pasando como
 // parámetro el código del proyecto activo actualmente. El
 // signo de interrogación de la instrucción Select anterior se
 // sustituirá por este parámetro.
 scursor.Open(codigo);
 // Se comienza en recorrido secuencial del cursor generado por
 // la dos acciones anteriores.
 scursor.Fetch(cuantia);
 // Se realiza un bucle que se ejecutará mientras existan tuplas
 // en el cursor. En este bucle se actualiza el contador de
 // cargos, se suma el nuevo cargo al total y se avanza hasta la
 // siguiente tupla.
 while scursor.Found do begin
  contador+=1;
  total+=cuantia;
  scursor.Fetch(cuantia);
 end
 // Se invoca a los métodos Close y Free de la clase SqlCursor
 // para cerrar el cursor y liberar la memoria asignada al
 // mismo.
 scursor.Close;
 scursor.Free;
 // Se muestran los resultados de la consulta en los campos de
 // edición definidos para ello.
 edit3.Text=contador;
 edit4.Text=total;
 edit5.Text=presu-total;
end
```

Otro de los métodos que se van a escribir es el correspondiente a las acciones que se desean ejecutar al abrirse la ventana correspondiente a esta consulta. Estas acciones se engloban dentro de un evento de la clase Form denominado Open que se ejecuta en el momento que se ejecuta una instancia de una clase derivada de Form. El código escrito es el siguiente:

```
On Open
Begin
 //deshabilitar los campos de edición correspondientes a las
 //columnas de la tabla "Proyectos"
 edit0.Disabled=true;
 edit1.Disabled=true;
 edit2.Disabled=true;

 //Realizar una consulta general sobre la tabla proyectos
 Query;

 //Invocar la ejecución de la funcion bCalcular
 bCalcular;
end
```

El resto de los métodos son los encargados de capturar los eventos de la barra de navegación que se ha definido dentro de la Screen. Cada vez que se avanza o retrocede sobre los datos de la tabla Proyectos se realiza una llamada a la función bCalcular.

```
On SpinDown navegador
begin
 proyectos.Next;
 bCalcular;
end

On SpinUp navegador
begin
 proyectos.Prev;
 bCalcular;
end

On SpinTop navegador
begin
 proyectos.First;
 bCalcular;
end

On SpinBottom navegador
begin
 proyectos.Last;
 bCalcular;
end
```

8. Ejecución de la nueva Consulta.

Dentro de la paleta del proyecto se selecciona el módulo “Inicio” y se pulsa el icono  , o también podemos pulsar el icono  que ejecuta el proyecto sin tener que seleccionar el modulo Main. (Figura 7.19)

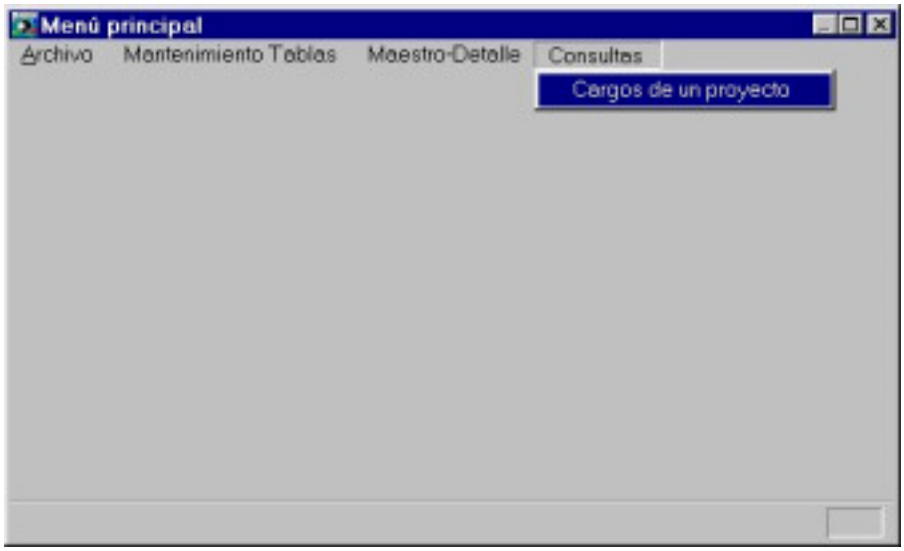


Figura 7.19. Ejecución del proyecto y selección de la consulta.

Seleccionamos dentro del menú principal la opción Consultas y hacemos click de ratón sobre la única consulta existente. La ejecución de esta consulta se muestra en la figura 7.20.

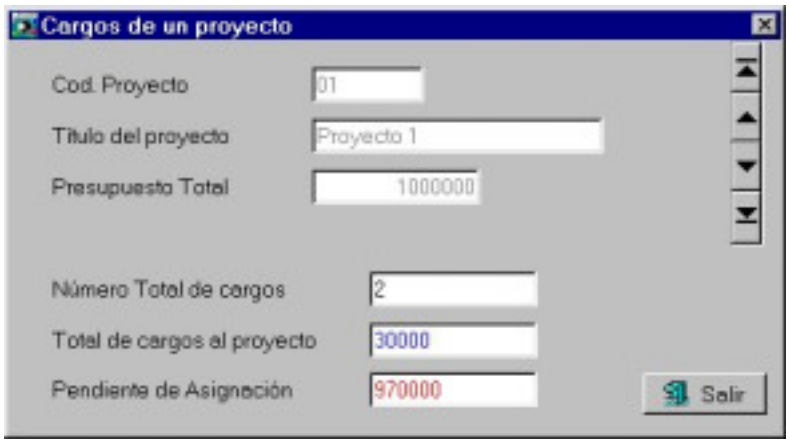


Figura 7.20. Consulta de los cargos del “Proyecto 1”

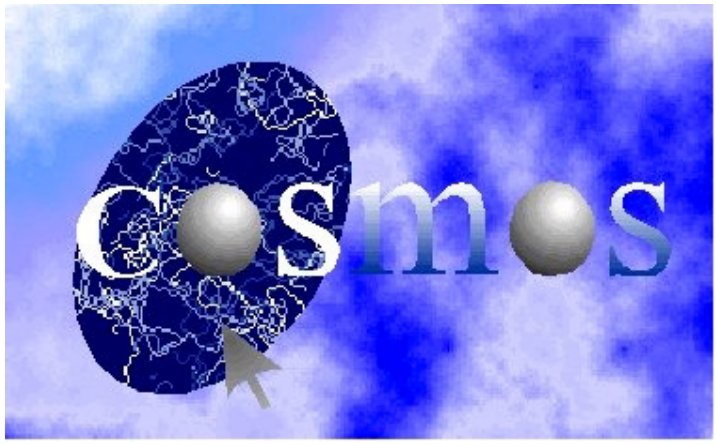
9. Archivos generados en este capítulo.

Los archivos generados en este capítulo son:

Archivo	Contenido
MCargos.smd	Fichero que contiene el código del módulo mCargos.
MCargos.omd	Fichero que contiene el código objeto del módulo mCargos.

Capítulo 8

Uso de la Clase SqlCursor de MultiBase COSMOS para la realización de informes



1. Introducción.

En este capítulo se va a realizar un informe con el listado de los Empleados de un determinado Departamento. Se utilizarán las clases predefinidas SqlCursor (definida en el capítulo 7), Page y PrnDocument.

2. La clase Page del COOL.

Esta clase permite diseñar en su definición, un formato de impresión mediante un editor gráfico. Los métodos de esta clase son:

Método	Utilidad
AddBand	Esta función permite añadir una banda a un control de tipo Band group de la página.
Calculate	Recalcula las dimensiones de la página y de sus controles, así como el número de elementos visibles que admite cada control de impresión de tipo grupo. Este método debe usarse si el tamaño de la página al ser editada varía respecto al de la impresora seleccionada en ejecución.
Clear	Inicializa los datos de la página a sus valores por defecto.
Control	Este método retorna por referencia el control de la página que tiene de identificador el indicado como parámetro. Permite acceder a un control de una clase Page abstracta.
LoadVars	Este método actualiza las variables de la página de impresión con el valor que tengan sus controles asociados.
PrintVars	Este método actualiza los controles de la página de impresión con los valores de sus variables asociadas.

3. La clase PrnDocument del COOL.

Esta clase nos proporciona los servicios necesarios para crear informes conectando con una impresora del sistema y enviándole las páginas que correspondan. Los métodos de esta clase son:

Método	Utilidad
AbortPrinter	Cancela la impresión del documento si este se encuentra en el administrador de impresión. Limpia el buffer de páginas y cierra la ventana de preview si esta se encuentra activa.
AttachPrinter	Permite asociar el dispositivo de impresión (impresora) que se desea utilizar.
Clear	Elimina las páginas de buffer del documento sin enviarlas al administrador de impresión e inicializa el contador de páginas.
ClientSize	Este método retorna las dimensiones del área de trabajo (altura y anchura) del papel seleccionado en la impresora asociada al documento.
ClosePrinter	Cierra el documento abierto previamente con OpenPrinter.
Flush	Vuelca sobre el documento de impresión el buffer de páginas de impresión, en el cual se han ido almacenando las páginas mediante el método SendPage.
GetNumCopies	Este método permite consultar el número de copias que se han pedido de cada página del documento.
GetPageNum	Indica el número de página que se esta procesando del documento de impresión.

Método	Utilidad
HideView	Esconde la ventana de Preview si esta se encuentra activa.
NumCopies	Permite indicar el número de copias que se desean de cada página del documento. Por defecto es 1.
OpenPrinter	Abre el documento de impresión para su salida por impresora.
Orientation	Este método permite consultar el tipo de orientación de la página que se tiene seleccionado para la impresora.
PaperSize	Este método retorna las dimensiones (altura y anchura) del papel seleccionado en la impresora asociada al documento.
Preview	Muestra la presentación preliminar del documento de impresión. Se muestra la ventana de preview de forma “modal”, esto es, el programa se detiene hasta que el usuario la cierra.
SendPage	Envía una página al buffer del documento de impresión donde se va almacenando antes de enviarlo a la impresora o antes de hacer la presentación preliminar. Refresca la ventana de Preview si esta se encuentra visible.
SetBufferPages	Indica el número máximo de páginas que puede alojar el buffer del documento de impresión.
SetDocName	Asigna un nombre al documento de impresión para su identificación en el administrador de impresión de Windows. Este nombre se utiliza también como título de la ventana de la presentación preliminar.
SetOrientation	Este método permite modificar el tipo de orientación de la página que se tiene seleccionado para la impresora.
SetPageNum	Permite iniciar la cuenta del número de página a un número determinado.
SetPaper	Este método permite modificar el tipo de papel que utilizará la impresora.
SetPaperName	Este método permite modificar el tipo de papel que utilizará la impresora. Recibe como parámetro el nombre del tipo de papel.
SetPaperSize	Este método permite modificar el tamaño de papel.
Setup	Este método muestra el dialogo de selección de impresoras y asocia al documento el dispositivo de impresión (impresora) que se seleccione.
ShowView	Muestra la ventana de presentación preliminar del documento de impresión de forma “no modal”, esto es, no se detiene la ejecución del programa.
WritePage	Equivale a SendPage y Flush. Si el documento ha sido abierto, no almacena la página en el buffer de almacenamiento de páginas sino que la envía directamente al dispositivo de impresión asociado al PrnDocument, esto es, la envía directamente al administrador de impresión de Windows.

4. Creación del módulo mEmpDpto.

Se va a añadir una nueva opción dentro del menú principal llamada “Listados” que será de tipo Popup. (Figura 8.1)

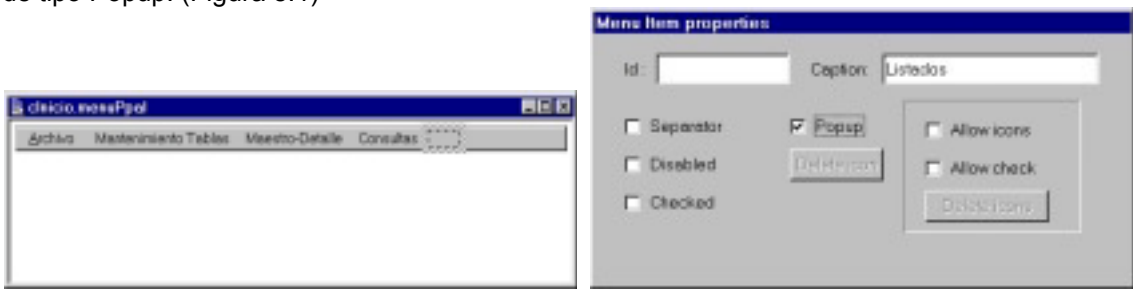


Figura 8.1. Añadir la opción Listados.

Dentro de esta opción se crea “Empleados de un Departamento” que lanzará la ejecución del nuevo módulo. (Figura 8.2)

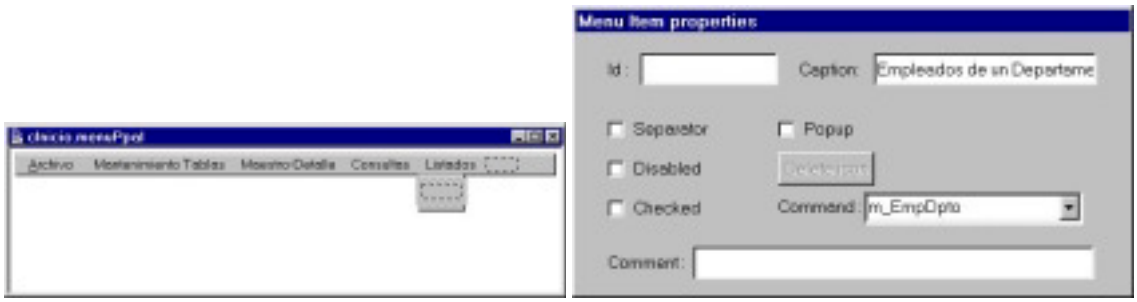


Figura 8.2. Añadir el nuevo tipo de Listados.

Dentro del código del módulo principal “Inicio” se añade el código necesario para lanzar la ejecución del módulo mEmpDpto.

```
On Command m_EmpDpto
begin
    module.Run("mEmpDpto");
end
```

En siguiente paso es crear un nuevo grupo dentro de la sección “Sources” de la paleta del Proyecto. (Figura 8.3)

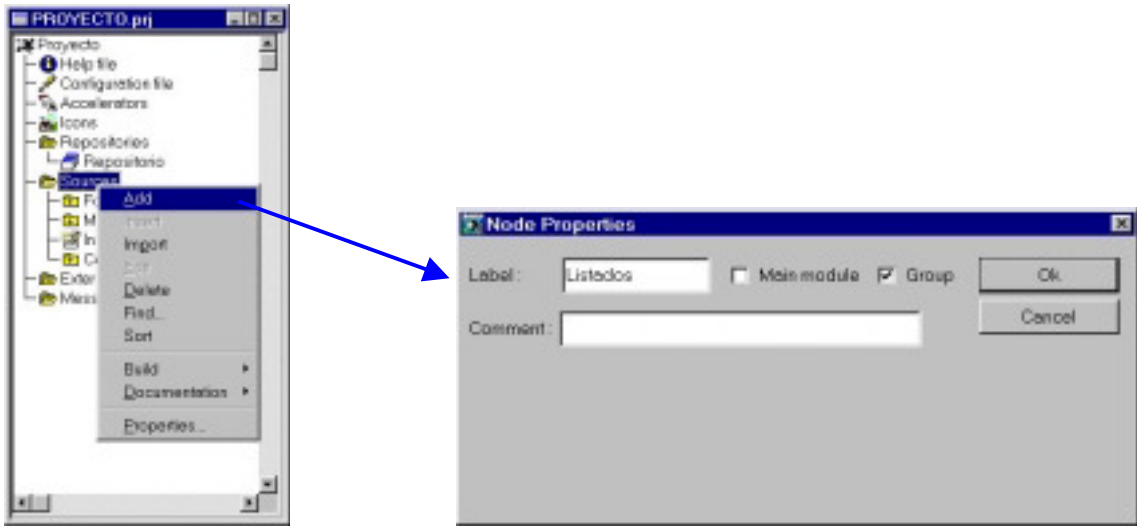


Figura 8.3. Añadir el grupo Listados.

Dentro de este nuevo grupo se añade el módulo mEmpDpto que englobará toda la funcionalidad necesaria para la realización del nuevo Listado. (Figura 8.4)

Figura 8.4. Añadir el nuevo módulo.

5. Creación de la clase cEmp.

Esta clase va a derivar de la clase predefinida Page que como se indica en el apartado segundo de este capítulo permite definir toda la funcionalidad necesaria para realizar listados. En la paleta del módulo mEmpDpto se añade esta nueva clase. (Figura 8.5)

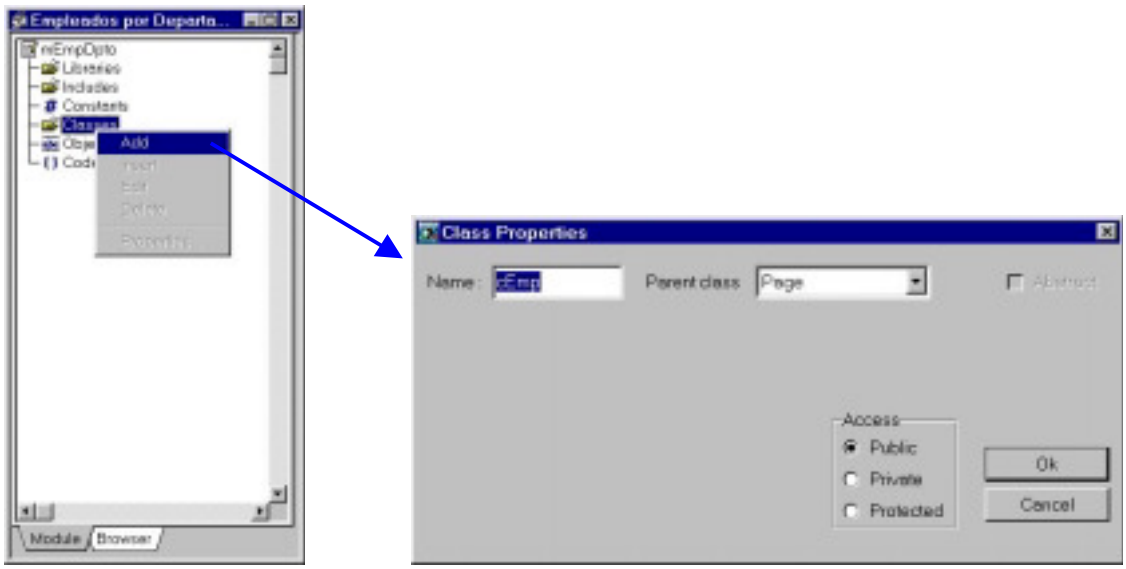


Figura 8.5. Añadir la clase cEmp al módulo cEmpDpto.

Las secciones de las clases derivadas de la clase Page son tres, como se muestra en la figura 8.6.

- Template: Sección en la que se define el diseño de la plantilla de la página para hacer formularios, impresos, listados, etc
- Variables: Sección opcional en la que se definen las variables que se utilizan en la página de impresión
- Code: Sección en la que se definen los métodos asociados a la clase Page.

Figura 8.6. Secciones de la clase cEmp.

Si se hace doble click de ratón sobre la sección “Templates” de esta clase se muestra una plantilla donde se realizará la definición de la funcionalidad del listado. (Figura 8.7)

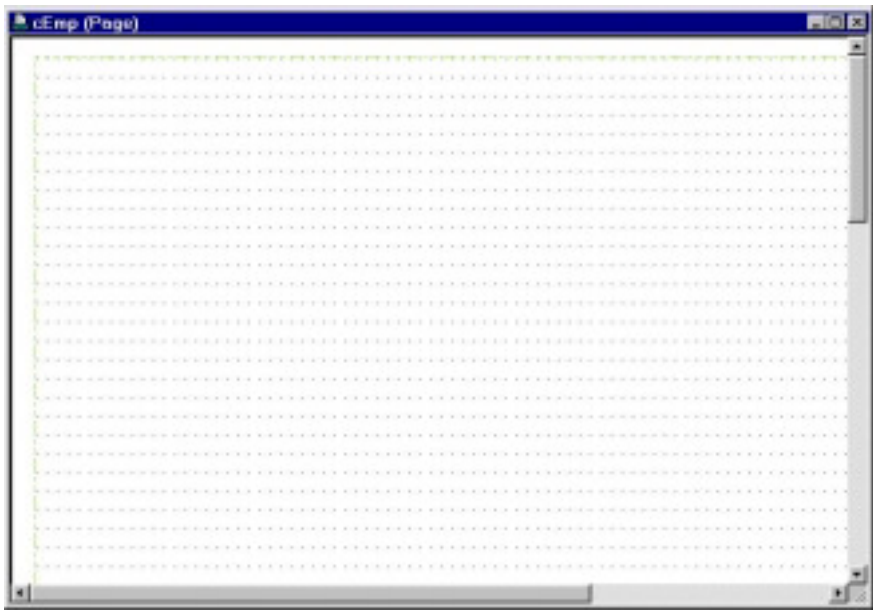


Figura 8.7. Plantilla para definir los componentes del listado.

6. Añadir variables a la clase cEmp.

Como se va a realizar un listado de los Empleados de un determinado Departamento, debemos añadir en la sección “Variables” de la clase cEmp, los columnas de la tabla que se quieren mostrar. Unicamente se mostrarán tres columnas: dni, nombre y apellidos de cada empleado.

Se añade el Repositorio al módulo mEmpDpto y se hace doble click para que se muestre su contenido.

Haciendo doble click de ratón sobre la sección “Variables” se muestra el contenido de la misma, de momento vacia. Se seleccionan las tres columnas requeridas de la paleta del Repositorio y se arrastran sobre la sección “Variables”. (Figura 8.8)

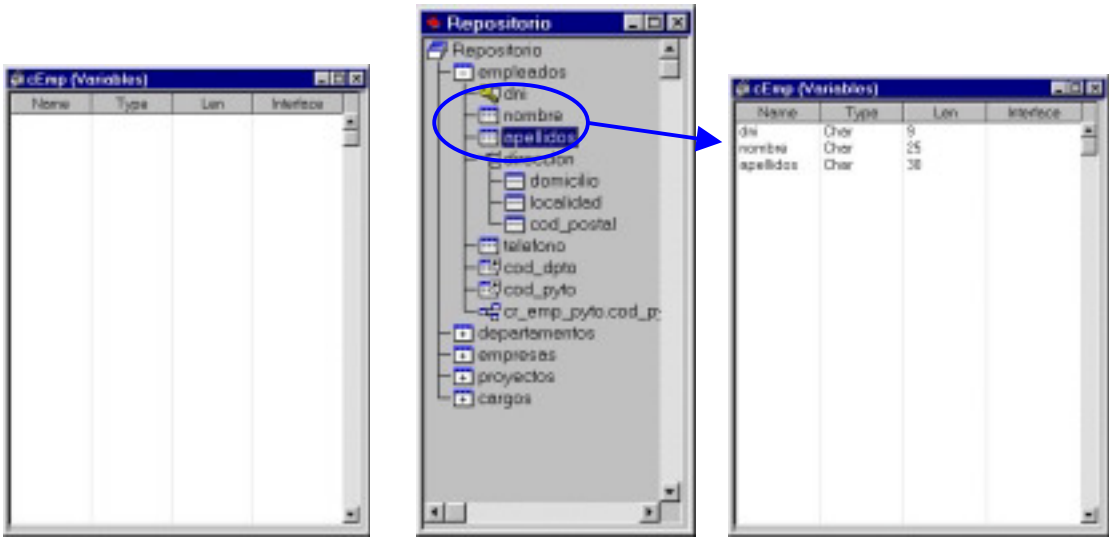


Figura 8.8. Añadir variables a la clase cEmp.

7. Diseño de la plantilla del listado.

Una vez se tienen las variables definidas en la nueva clase se procede a realizar el formato del listado.

En la parte derecha del Editor Visual de COSMOS, cuando se tiene abierta la plantilla del listado, se muestran una paleta con los diferentes controles que se pueden introducir en la plantilla. (Figura 8.9)

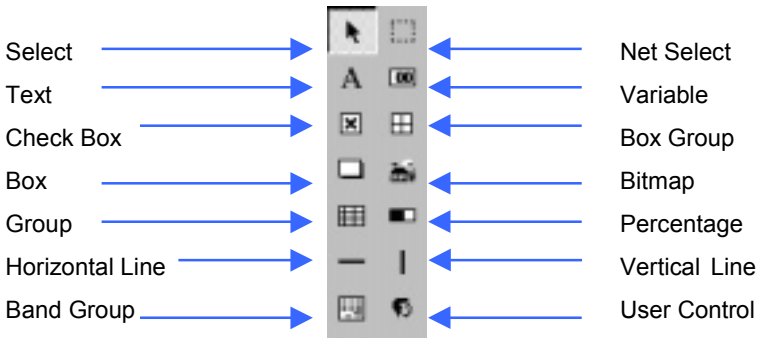


Figura 8.9. Paletas para el diseño de la plantilla del Listado.

Se introduce un control de tipo Variable en la parte superior de la plantilla que mostrará el nombre del Departamento del que se muestran los Empleados. Como este nombre se conocerá en tiempo de ejecución, dentro de las propiedades del control se indica un identificador para poder enviarle la información a mostrar. También se indica el tipo de variable que mostrará y el aspecto. (Figura 8.10)

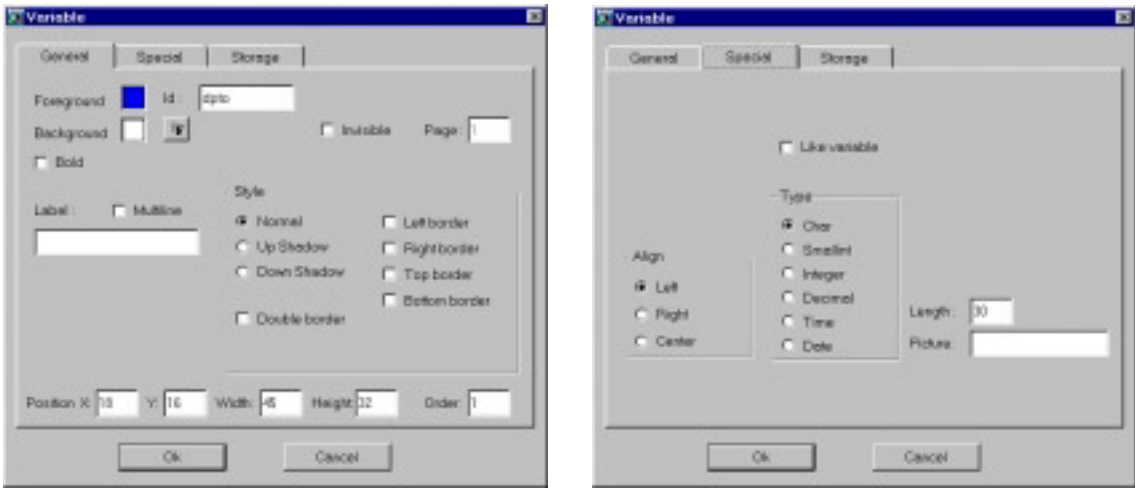


Figura 8.10. Propiedades de la variable del Departamento.

En la parte superior derecha del listado se va a mostrar el número de página del listado. Se introduce un control de tipo Text para escribir “Página:” y un control de tipo Variable que será donde se muestre el número de página actual. (Figura 8.11)

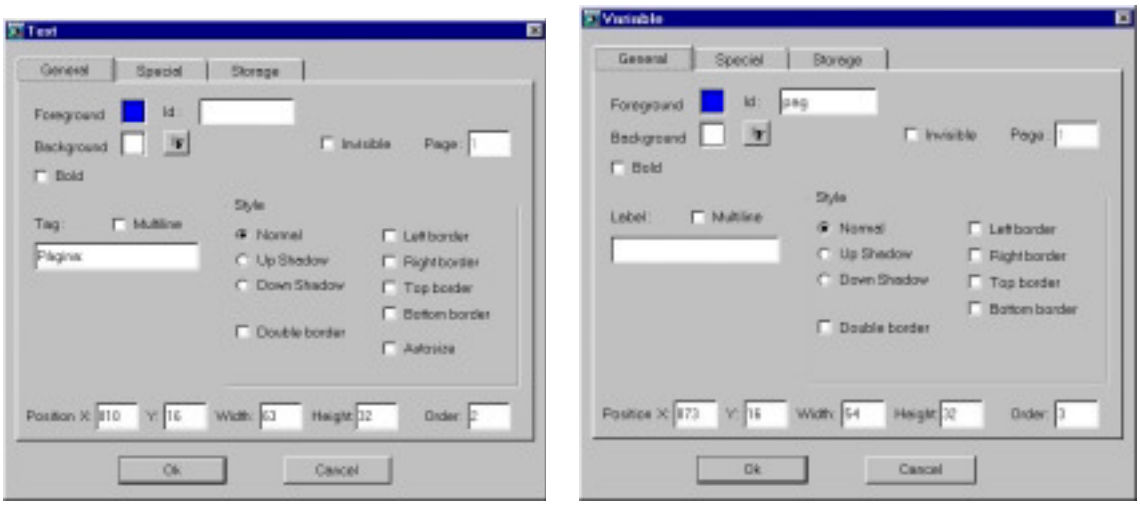


Figura 8.11. Propiedades de los controles para mostrar el número de página.

El último título que se introducirá será un sencillo control de tipo Text que muestra el literal “Listado de Empleados”.

La plantilla con los pasos realizados se muestra en la figura 8.12. A todos los controles se les ha modificado el tipo de letra y el color de la misma.

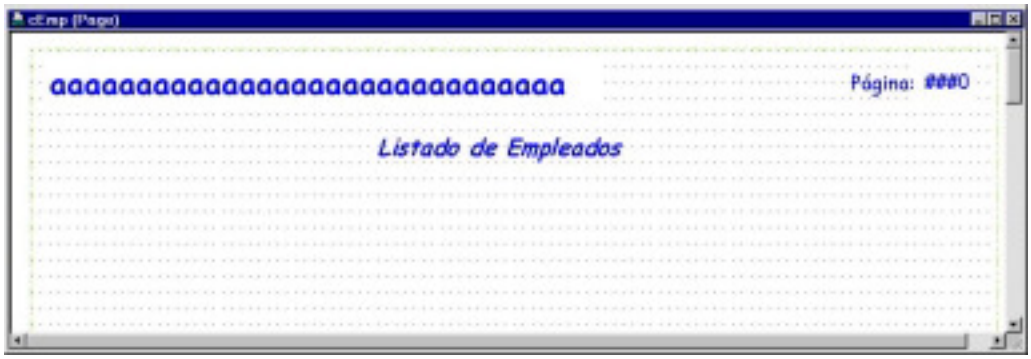


Figura 8.12. Aspecto del tapiz con los títulos.

Se añade un nuevo control de tipo Group. Un Grupo es un control de impresión que sirve para agrupar un conjunto de controles que se van a repetir por filas y/o columnas. La repetición de los controles del grupo se define en las propiedades del control. Se estira el control hasta que ocupe toda la dimensión de la plantilla. (Figura 8.13)

Figura 8.13. Inserción del control de tipo Group.

Las propiedades de este control se muestran en la figura 8.14. Se ha añadido un identificador al control para poder hacer referencia a él dentro del código de la clase cEmp. En la pestaña Special se ha definido el tipo de repetición de los campos, por filas, y se ha seleccionado las opciones Vertical y Horizontal Grid para que se muestre las líneas separadoras de filas y columnas.

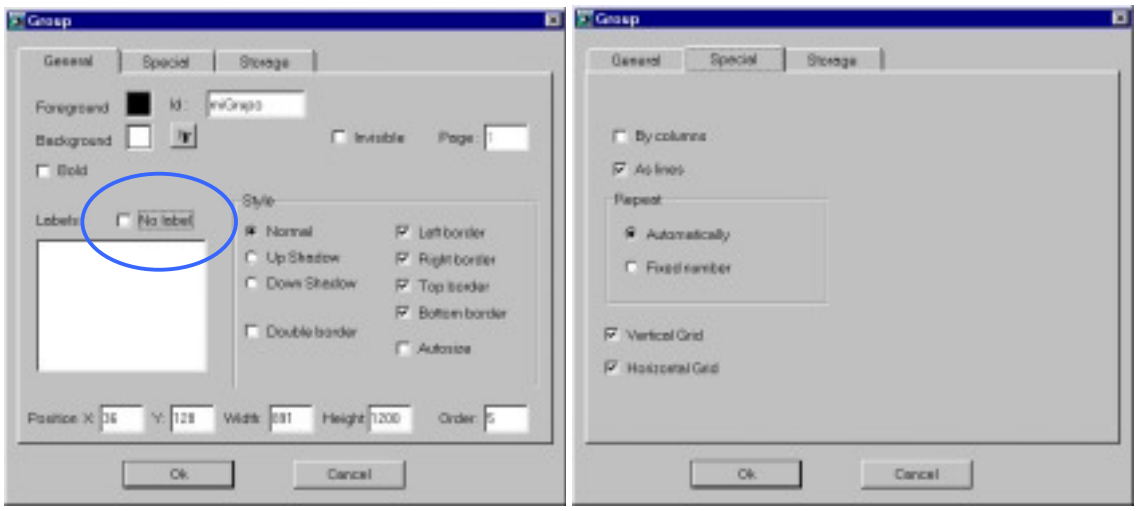


Figura 8.14. Propiedades del control de tipo Group.

Sobre este control se añaden tres controles de tipo Variable que contendrán respectivamente el dni, apellidos y nombre de los empleados. Se selecciona dentro de la pestaña “Storage” de cada uno la variable correspodiente. (Figura 8.15)

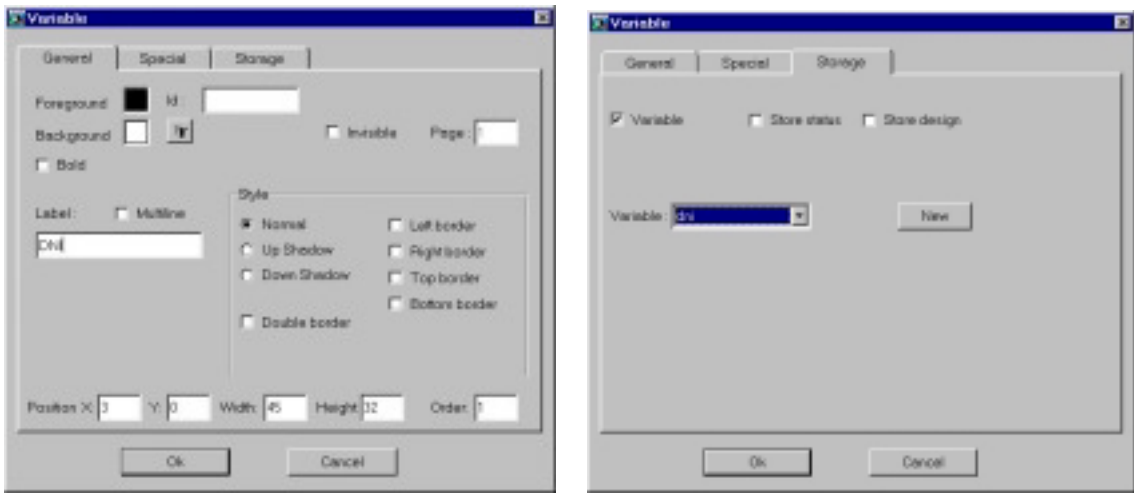


Figura 8.15. Propiedades del control Variable que contiene la columna DNI.

Realizados los pasos anteriores el aspecto que presenta la plantilla del listado se muestra en la figura 8.16. Las etiquetas de las columnas se han definido en la pestaña General de cada uno de los controles de tipo Variable y dentro de la misma pestaña del control Group quitando la selección de la casilla “No labels”.

Figura 8.16. Aspecto definitivo de la plantilla del listado.

Pulsando el botón secundario del ratón sobre la plantilla y seleccionando la opción Test, se muestra el aspecto que tendrá el listado una vez se rellenen todos sus campos. (Figura 8.17)

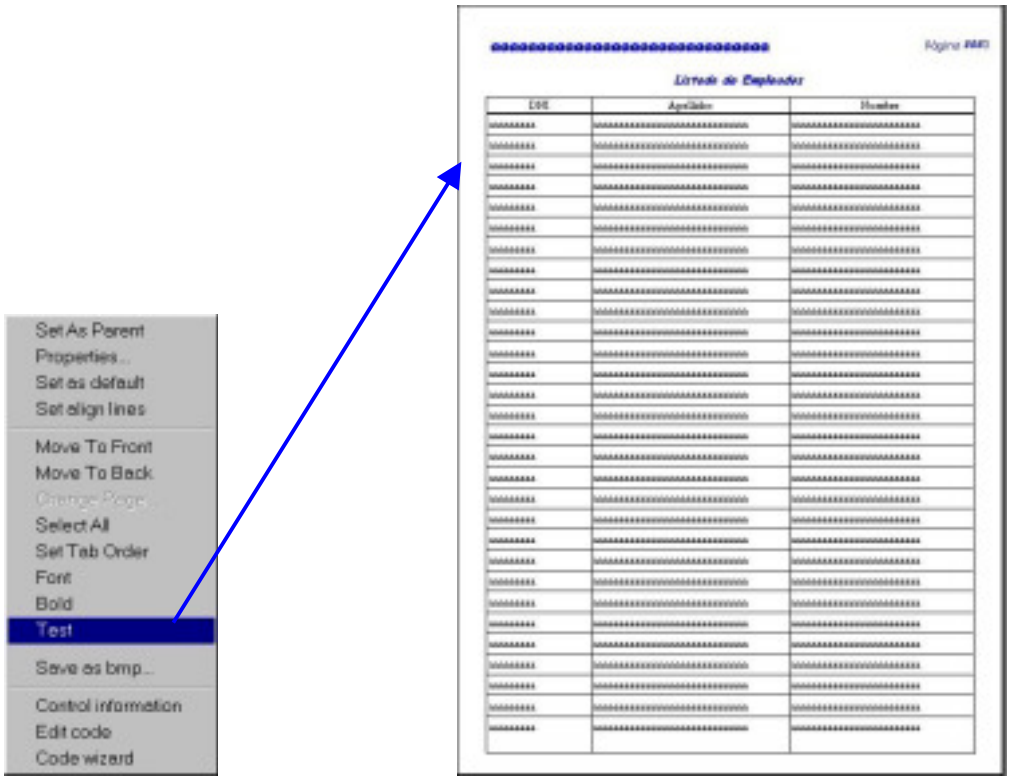


Figura 8.17. Opción Test de la plantilla del Listado.

8. Código de la clase cEmp.

A continuación se realiza la definición del código de la clase. Unicamente se define un método de acceso público llamado Listar.

Este método recibe dos parámetros que son el código y el nombre del Departamento del que se va a realizar el listado de Empleados.

```
public Listar (cod as smallint, nomdpto as char)
objects
begin
 nlin npag as smallint
 scursor as SqlCursor
 //definición de un objeto de la clase PrnDocument
 prn as PrnDocument
end
begin
 //se escribe el nombre del Departamento pasado en el segundo
 //parametro en el control "dpto" definido en la plantilla
 dpto.Text=nomdpto;
 //Instrucción select sobre la tabla Empleados para obtener
 //los empleados del Departamento pasado como parametro
 scursor.Prepare( "select dni, nombre, apellidos from empleados"
                + "where cod_dpto=?"
                + "order by nombre " );
 scursor.Into(dni,nombre,apellidos);
 scursor.Open(cod);
 //Inicialización del número de líneas y páginas
 nlin=npag=1;
 scursor.Fetch;
```

```
 //Se escribe el número de página en el control "pag" definido
 //en la plantilla
 pag.Text=npag;
 //Recorrido secuencial del cursor realizando una llamada al
 //método PrintVars del control de tipo Group con cada tupla
 while scursor.Found do begin
  if nlin>miGrupo.Count then begin
     //Página completa
     pag.Text=npag;
     ++npag;
     prn.SendPage(Self);
     Self.Clear;
     nlin=1;
  end
  miGrupo.CurrentRow=nlin;
  miGrupo.PrintVars;
  scursor.Fetch;
  ++nlin;
 end
 //Cerrar el cursor y liberar la memoria asignada
 scursor.Close;
 scursor.Free;
 //Enviar a la impresora la última página
 prn.SendPage(Self);
 //Mostrar la Vista Preliminar del listado
 prn.Preview;
end
```

9. Creación de la clase cDpto.

Cuando se ejecute el nuevo módulo se mostrará una pantalla donde se seleccionará el Departamento del que se desea obtener el listado de Empleados. Para realizar esta selección se crea la clase cDpto que es una derivación de la clase predefinida Form. (Figura 8.18)

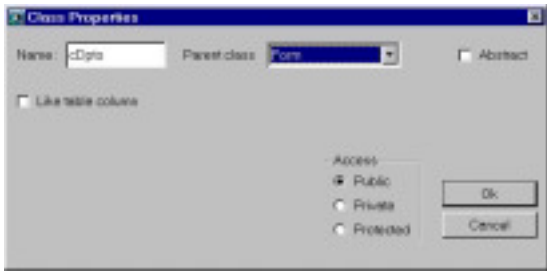


Figura 8.18. Clase cDpto.

Dentro de la sección "Tables" de la nueva clase se añade la tabla de Departamentos del Repositorio. El diseño de la pantalla se muestra en la figura 8.19.

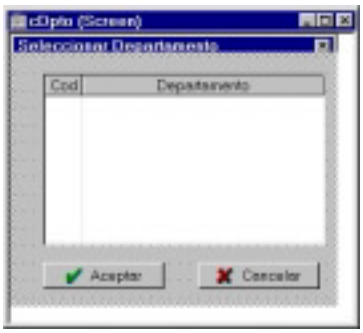


Figura 8.19. Screen de la clase cDpto.

Esta Screen tiene únicamente tres elementos, un control de tipo List Box y dos Push Buttons.

El botón Cancelar ejecuta la instrucción Close de la Form y el botón Aceptar ejecuta el comando “Aceptado”. (Figura 8.20)

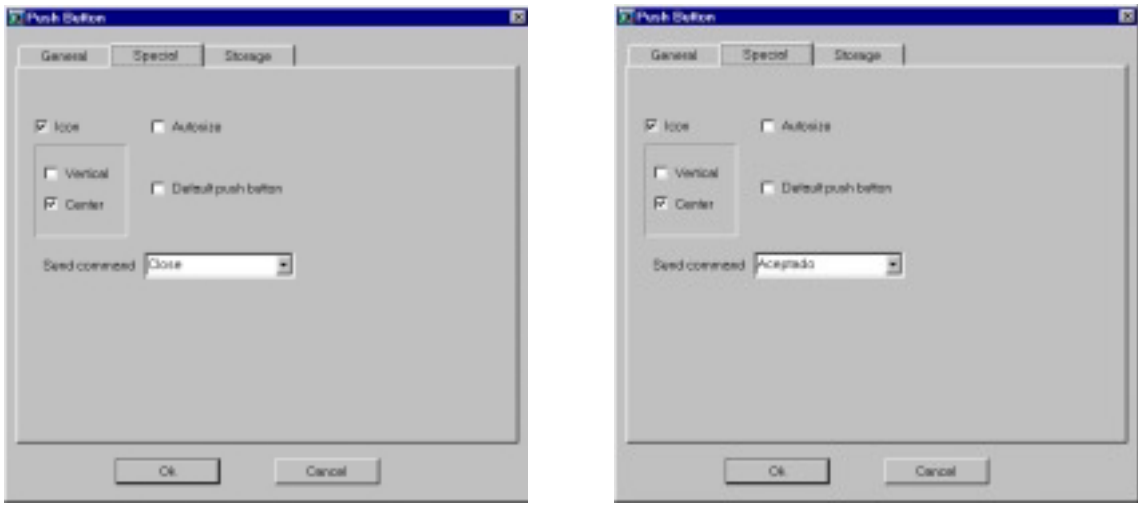


Figura 8.20. Propiedades de los botones Aceptar y Cancelar.

En control de tipo List Box se ha definido de forma que sea una lista de selección simple, que sólo permite tener seleccionado un ítem a la vez. Además es de tipo SQL, es decir, muestra el resultado de una instrucción SELECT del SQL. Tendrá además dos columnas que mostrarán el código y el nombre del Departamento. Las propiedades de este control se muestran en la figura 8.21.

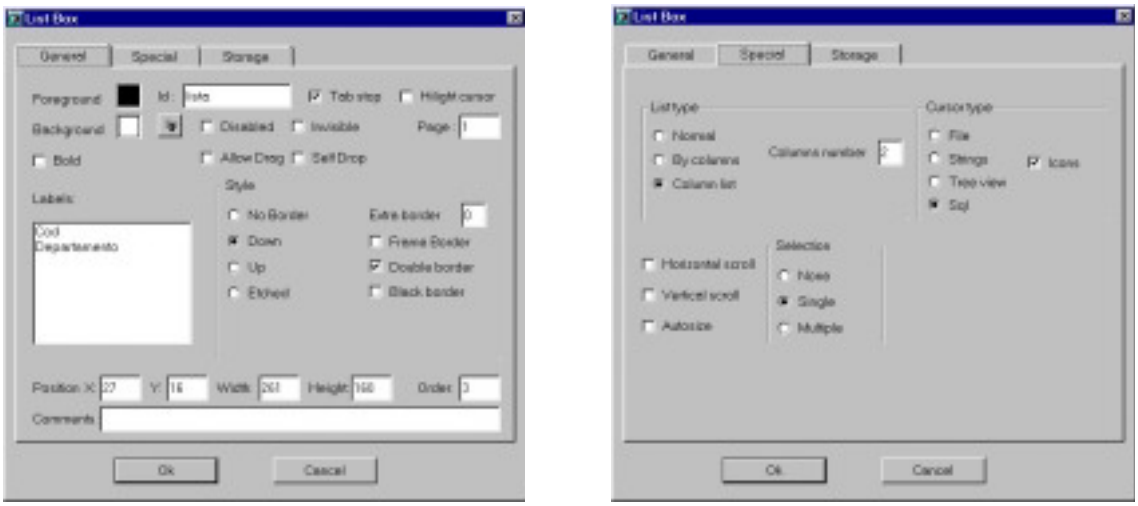


Figura 8.21. Propiedades del control Lista de tipo List Box.

El código que se ha introducido en la sección Code de la clase cDpto son dos métodos:

- El método On Open que realiza la consulta sobre los Departamentos existentes en la base de Datos y muestra los resultados en el control Lista

```
On Open
begin
 lista.LoadSelect("select cod_dpto,nombre from departamentos”
                  +” order by cod_dpto",2);
end
```

- El comando Aceptado que lanzará la ejecución del informe con el listado de los Empleados del Departamento selecionado. El código de este comando se muestra a continuación.

```
On Command Aceptado
objects
begin
 //definición de un objeto de la clase cEmp
 oEmp as cEmp
 posicion codigo as smallint
 nombre as char(50)
end
begin
 //obtener el elemento seleccionado de la lista.
 posicion=lista.Selected;
 //obtener el código y el nombre del departamento seleccionado
 codigo=lista.GetListColumnText(posicion,1);
 nombre=lista.GetListColumnText(posicion,2);
 //llamada al método Listar de la clase cEmp
 oEmp.Listar(codigo,nombre);
 //cerrar la pantalla
 Close;
End
```

10. Código del módulo mEmpDpto.

Una vez definidas las clases que componen el módulo, se selecciona la sección Code del mismo y se escribe el código necesario para la conexión a la base de datos. Se crea una instancia de la clase cDpto y se invoca sobre este objeto la ejecución del método Run de la clase Form.

```
main
objects
begin
 oDpto as cDpto
end
begin
 Sql.AttachConnection("conexion");
 Sql.Connect("datos");
 oDpto.Run;
 Sql.Disconnect;
 Sql.DettachConnection;
End
```

11. Ejecución del módulo mEmpDpto.

Después de realizar la compilación del nuevo módulo, se procede a la ejecución de la aplicación. Dentro del menú se selecciona la opción añadida en este capítulo. (Figura 8.22)

Figura 8.22. Ejecución del nuevo módulo.

Se muestra a continuación la Screen de la clase cDpto donde aparecen los Departamentos que existen en la empresa. Se selecciona el primero de ellos y se pulsa el botón Aceptar. (Figura 8.23)

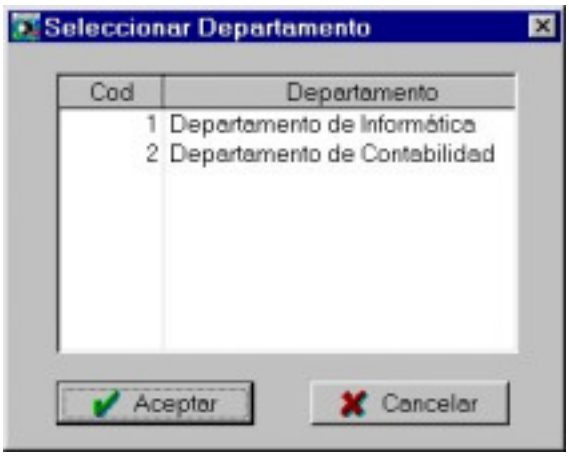


Figura 8.23. Selección del Departamento.

Se muestra a continuación la vista preliminar del listado (Figura 8.24). En la parte inferior de esta vista previa existen los siguientes elementos:

 Barra de Navegación: Para recorrer hacia delante y hacia atrás las diferentes páginas del listado, si existe mas de una

Indicador del número de páginas: Indica el número de la página actual y el número total de páginas.

Zoom del documento: para ampliar o disminuir el tamaño de la vista preliminar del listado.

Botón Print: envía a la impresora el listado y cierra la vista preliminar.

Botón Close: cierra la vista preliminar.

Figura 8.24. Vista preliminar del listado de los empleados del Departamento de Informática.

12. Archivos generados en este capítulo.

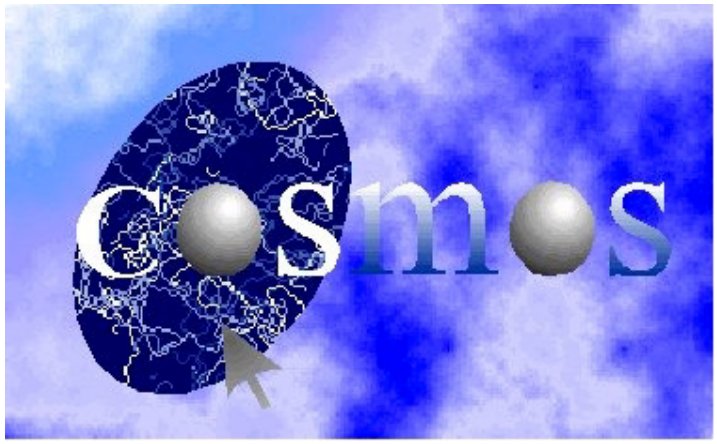
Los archivos generados en este capítulo son:

Archivo	Contenido
mEmpDpto.smd	Fichero que contiene el código del módulo.
mEmpDpto.omd	Fichero que contiene el código objeto del módulo.

Capítulo 9

Uso de la Clase SqlStatement de MultiBase COSMOS

1. Introducción.
2. La clase SqlStatement del COOL.
3. Creación del módulo mInicialización.
4. Código del módulo mInicialización.
5. Ejecución del módulo mInicialización.
6. Archivos generados en este capítulo.



1. Introducción.

En el presente capítulo se va a construir un nuevo módulo en el proyecto de los capítulos anteriores, cuya finalidad será el borrado del contenido de todas las tablas. Para poder realizar esta acción se va a utilizar la clase SqlStatement del COOL.

2. La clase SqlStatement del COOL.

Esta clase derivada de la clase Complex del COOL de COSMOS, es derivable e instanciable (el usuario puede crear objetos de dicha clase). Permite la comunicación con el SQL. En esta clase se almacenan los siguientes objetos:

- Instrucciones SQL de consulta: SELECT.
- Instrucciones para el tratamiento dinámico del SQL. Son instrucciones de tipo SQL construidas mediante literales y expresiones para posteriormente ejecutarlas.

Los métodos de esta clase son:

Método	Utilidad
AttachServer	Asocia un servidor a la instrucción SQL, cuando se ejecute la instrucción se hará contra el servidor asociado.
Error	Indica el código del último error producido.
ErrMsg	Muestra el string asociado al último error producido en el servidor.
Execute	Ejecuta una instrucción del SQL previamente preparada con el método Prepare. Es un método para el tratamiento de instrucciones SQL de forma dinámica.
Free	Libera la memoria alocada para la sentencia en curso. Es conveniente liberar la memoria cuando no se va a ejecutar más esta sentencia.
Prepare	Este método permite construir una instrucción SQL de forma dinámica (posibilidad de concatenar expresiones) para la comprobación de su sintaxis.
Rows	Retorna el número de filas procesadas en la última operación de INSERT, DELETE, UPDATE, SELECT, LOAD realizada.
SqlExec	Prepara y ejecuta una instrucción SQL. Sirve para el tratamiento de instrucciones SQL de forma dinámica.

3. Creación del módulo mInicialización.

Dentro del menú principal se va a añadir una nueva opción llamada “Inicialización de Tablas”. Se selecciona dentro de la paleta del proyecto el módulo Inicio y se hace doble click de ratón sobre él.

Dentro de la paleta del módulo Inicio se selecciona menuPpal y se hace doble click sobre este elemento. (Figura 9.1)

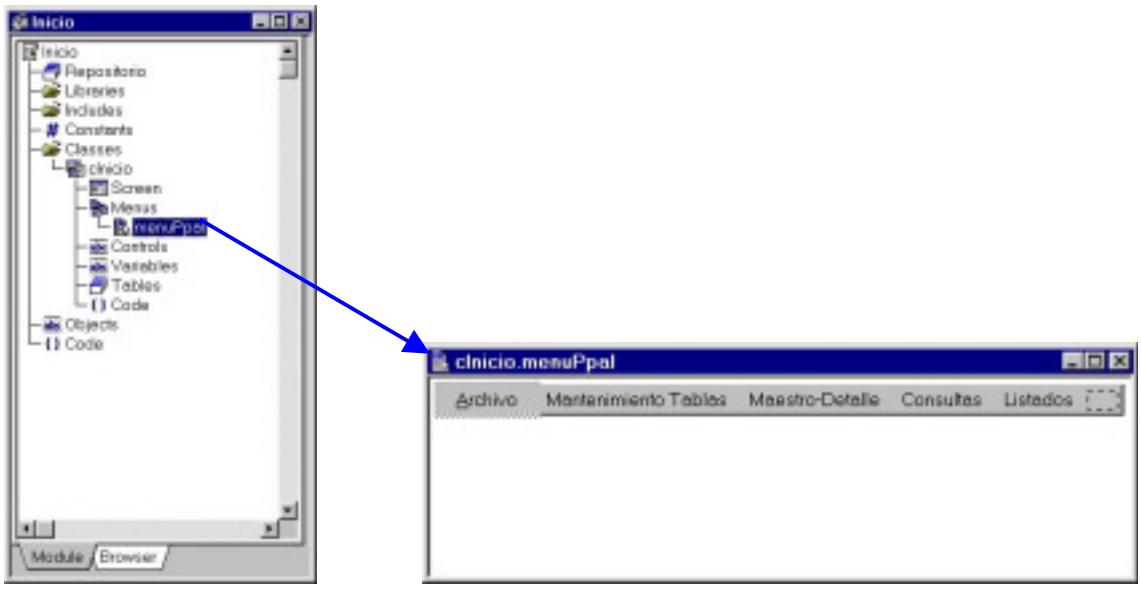


Figura 9.1. Selección del menú principal.

En este menú se añade la nueva opción dentro del menú Archivo. Cuando se pulse la nueva opción se ejecutará el método “m_Inicializacion” de la clase clnicio que es quien invocará la ejecución del nuevo módulo que se creará. (Figura 9.2)

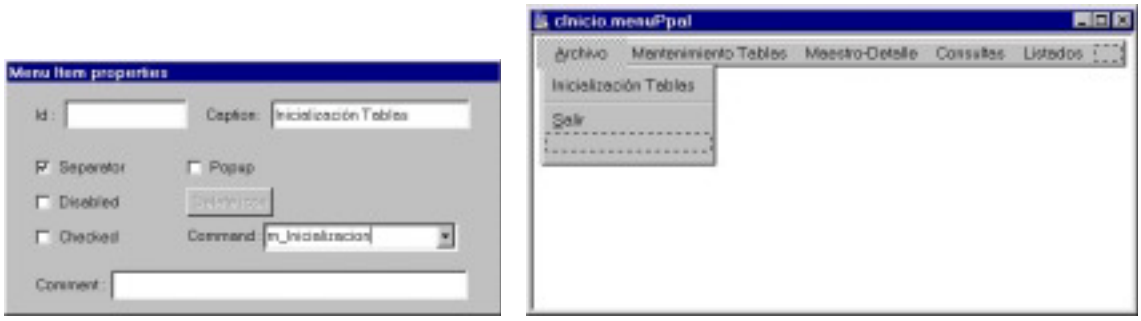


Figura 9.2. Creación de la nueva opción de menú.

Se crea un nuevo grupo de programas llamado Varios, dentro de la sección Sources de la paleta del proyecto. (Figura 9.3)

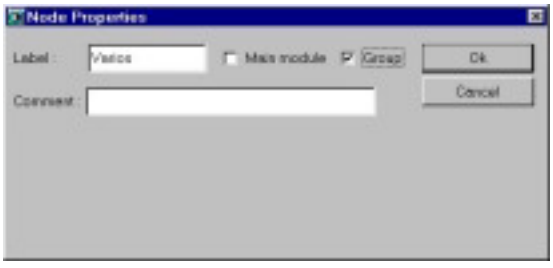


Figura 9.3. Creación de un nuevo grupo.

Dentro de este grupo se añade un nuevo módulo llamado “mInicialización”, este nuevo módulo no va a tener ninguna clase definida en él. (Figura 9.4)

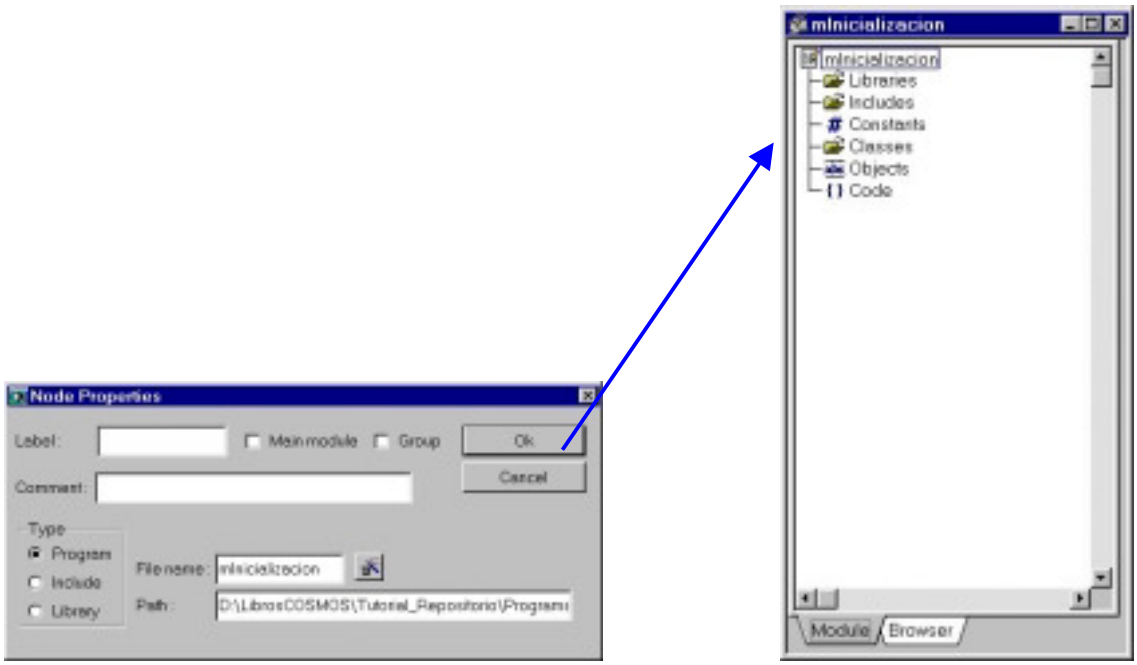


Figura 9.4. Creación del nuevo módulo.

4. Código del módulo mInicializacion

El método main de este módulo realizará la conexión a la base de datos y mostrará un mensaje al usuario pidiendo la confirmación del borrado de los datos de las tablas.

Si el usuario confirma el borrado de los datos se invoca la ejecución de un método privado llamado borrarTablas.

```
main
objects
begin
 msg as boolean
end
begin
 //Conexión de la base de datos
 Sql.AttachConnection( "conexion" );
 Sql.Connect( "datos" );

 //Solicitud de confirmación de borrado
 msg=Ok( "Desea vaciar el contenido de las tablas" ,
         "PRECAUCION",false);
 if msg then begin
   borrarTablas;
  end
 else begin
  Exit;
 end

 //Desconexión de la base de datos
 Sql.Disconnect;
 Sql.DettachConnection;
end
```

El método borrar líneas utiliza la clase SqlStatement para realizar el borrado del contenido de las tablas empleados, departamentos, empresas, proyectos y cargos.

```
private borrarTablas
objects
begin
 //Definición de un objeto de la clase SqlStatement
 st as SqlStatement
end
begin
 //Borrar el contenido de la tabla Empleados
 //Se construye la instrucción SQL
 st.Prepare( "delete from empleados" );
 //Se realiza la ejecución de la instrucción SQL
 st.Execute;
if st.Error<0 then
  st.ErrMsg.Trace;
 //Se muestra por pantalla el número de tuplas borradas
 ("Tabla Empleados: "+st.Rows+" tuplas borradas").Trace;
 //Libera la memoria asignada
 st.Free;

 //Borrar el contenido de la tabla Departamentos
 st.Prepare( "delete from departamentos" );
 st.Execute;
 if st.Error<0 then
  st.ErrMsg.Trace;
 ("Tabla Departamentos: "+st.Rows+" tuplas borradas").Trace;
 st.Free;

 //Borrar el contenido de la tabla Empresas
 st.Prepare( "delete from empresas" );
 st.Execute;
 if st.Error<0 then
  st.ErrMsg.Trace;
 ("Tabla Empresas: "+st.Rows+" tuplas borradas").Trace;
 st.Free;

 //Borrar el contenido de la tabla Proyectos
 st.Prepare( "delete from proyectos" );
 st.Execute;
 if st.Error<0 then
  st.ErrMsg.Trace;
 ("Tabla Proyectos: "+st.Rows+" tuplas borradas").Trace;
 st.Free;

 //Borrar el contenido de la tabla Cargos
 //Se utiliza el métod SqlExec en lugar de Prepare y Execute
 st.SqlExec( "delete from cargos" );
 if st.Error<0 then
  st.ErrMsg.Trace;
 ("Tabla Cargos: "+st.Rows+" tuplas borradas").Trace;
st.Free;
end
```

5. Ejecución del módulo mInicialización.

Cuando se selecciona la opción creada en el apartado 3 se muestra un cuadro de diálogo donde se pide la confirmación del borrado de datos. (Figura 9.5)

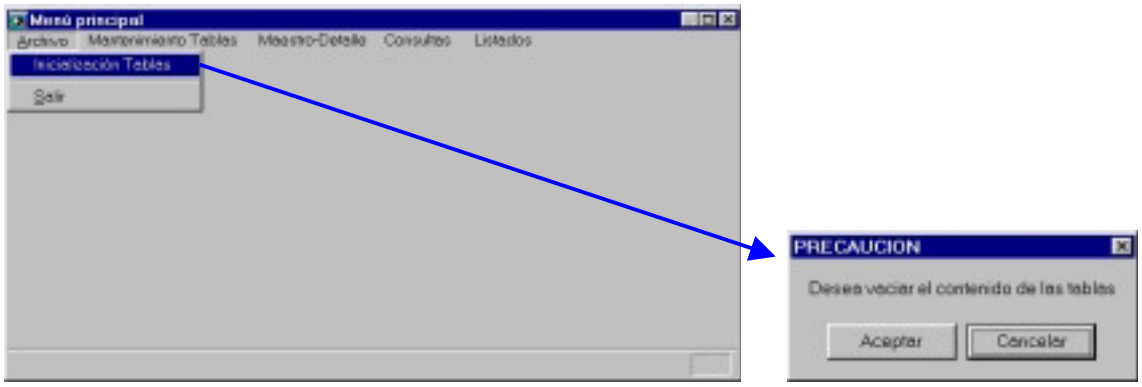


Figura 9.5. Solicitud de confirmación de borrado.

Si se pulsa el botón  se procede al borrado efectivo de los datos de las tablas que componen la base de datos. Cada vez que se borra una tabla se muestra un mensaje con el nombre de la tabla y el número de tuplas que se han borrado. (Figura 9.6)

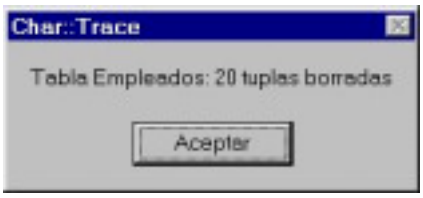


Figura 9.6. Mensaje de borrado de tablas.

Una vez borradas todas las tablas si se realiza la consulta de “Cargos de un Proyecto” que se desarrollo en el capítulo 7, la pantalla aparece vacía porque ya no existe ningun proyecto ni ningún cargo. (Figura 9.7)

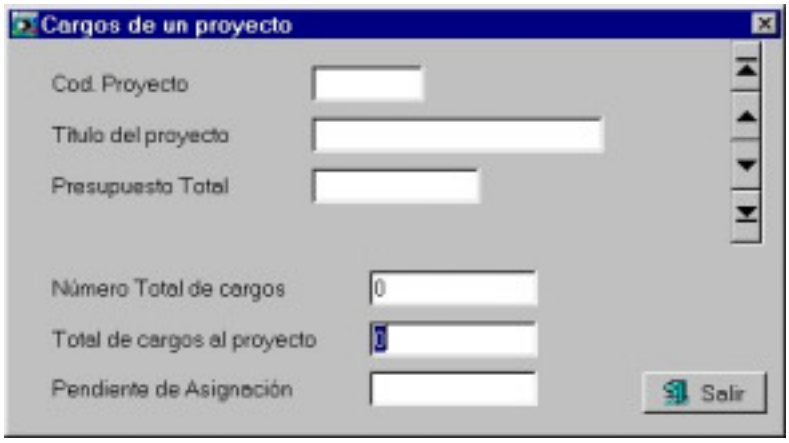


Figura 9.7. Las consultas están vacías.

6. Archivos generados en este capítulo.

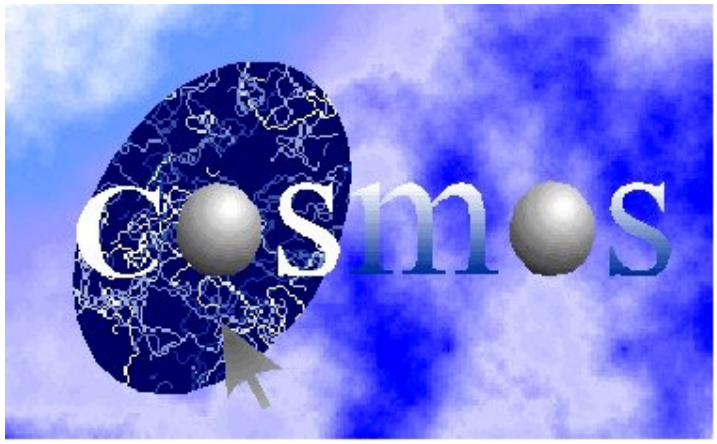
Los archivos generados en este capítulo son:

Archivo	Contenido
mInicializacion.smd	Fichero que contiene el código del módulo.
mInicializacion.omd	Fichero que contiene el código objeto del módulo.

Capítulo 10

Uso de varias bases de datos con un único Repositorio

1. Introducción.
2. Especificación de Requisitos.
3. Creación del entorno de trabajo.
4. La clase Module del COOL.
5. Creación del módulo mNuevo.
6. Creación del módulo mAbrir.
7. Creación del módulo mBorrar.
8. Modificación del módulo mInicialización.
9. Ejecución de la aplicación.
10. Archivos generados en este capítulo.



1. Introducción.

En este capítulo se va a mostrar la utilización de un repositorio para la definición y manejo de varias bases de datos.

En el capítulo 5 se realizó el análisis y diseño de la aplicación que realiza la gestión de proyectos de una empresa. Ahora se va a ampliar la aplicación para que se realice la gestión de proyectos de varias empresas diferentes.

2. Especificación de Requisitos.

Los puntos básicos de la aplicación son:

- Se desea realizar una gestión de proyectos multiempresa.
- El repositorio generará una base de datos que se utilizará como plantilla para la realización de varias bases de datos con la misma estructura.
- El repositorio es el mismo que se definió en el capítulo 5.
- Se seleccionará en cada momento la empresa que se desea gestionar.

3. Creación del entorno de trabajo.

Para organizar la gestión de varias empresas, se necesita definir una conexión en el Editor de Configuración de Cosmos (véase Anexo B) que sea genérica. Las variables de entorno de esta conexión se van a modificar por el programa para que en cada momento se seleccione la base de datos correspondiente a una empresa determinada.

Para mantener la estructura de las tablas de la base de datos del repositorio, estas se mantienen en un subdirectorio llamado “datos.dbs”. Cada una de las empresas tendrán su propia base de datos con la estructura del repositorio pero con sus datos particulares.

4. La clase Module del COOL.

Esta clase define la funcionalidad básica de un modulo Cosmos (programas, librerías, includes). Los métodos esta clase son:

Método	Utilidad
AsgVar	Este método permite modificar el valor del objeto global sobre el que se aplica.
Beep	Este método hace sonar una alarma correspondiente a un nivel de alerta dado. El sonido para cada uno de los niveles de alerta está identificado en el panel de control de Windows.
CloseDir	Este método permite cerrar un directorio abierto para su lectura con OpenDir.
Copy	Este método permite copiar (duplicar) un fichero con otro nombre.
ChDir	Este método cambia el directorio en curso.
CosmosDir	Este método indica el directorio donde se encuentra instalado Cosmos.
Delete	Este método permite borrar un fichero.
DesktopSize	Este método retorna las dimensiones del escritorio de Windows en pixeles.
Ending	Este método puede ser definido en cualquier módulo COOL y se ejecutará cuando se descargue el módulo de memoria.

Método	Utilidad
ErrorLevel	Este método permite modificar el nivel de error de la aplicación asociada al módulo.
EvalVar	Este método retorna por referencia el valor del objeto sobre la que se aplica.
Exec	Ejecuta un programa Windows o MS-DOS sin esperar a que termine.
Exit	Este método permite abortar la ejecución de la aplicación o del módulo de la aplicación que se esté ejecutando en ese momento.
FileSize	Este método devuelve el tamaño, en bytes, de un fichero.
GetCursorPos	Este método devuelve la posición del cursor en la pantalla.
GetCwd	Este método devuelve el directorio en curso.
GetEnv	Este método devuelve el valor de la variable de entorno que se le pasa como parámetro.
GetIni	Este método permite obtener el valor correspondiente a una entrada de una sección de un fichero de configuración.
GetOption	Este método permite consultar el comportamiento de un módulo en ejecución.
Hinstance	Este método retorna la instancia Windows del programa Cosmos que se está ejecutando.
HMS	Este método devuelve un objeto de tipo Time.
IsLoaded	Este método permite consultar si un módulo Cosmos esta cargado en memoria.
License	Este método retorna el número de licencia de Cosmos.
Load	Este método permite cargar en memoria un módulo Cosmos que pertenezca al proyecto en curso.
MDY	Este método devuelve un objeto de tipo Date.
MkDir	Este método permite crear un directorio.
Move	Este método permite renombrar y mover de directorio un fichero.
MsgText	Devuelve el texto del mensaje del fichero de mensajes que se le indica.
MsgTextDef	Devuelve el texto del mensaje del fichero de mensajes que se le indica. Si no lo encuentra puede devolver el mensaje que el usuario desee.
MessageBox	Este método muestra un cuadro de mensaje para pedir una respuesta al usuario.
OpenDir	Este método prepara un directorio para su lectura.
Ok	Este método muestra un cuadro de mensaje con los botones «Ok» y «Cancel» para pedir una respuesta al usuario.
ProjectDir	Este método indica el directorio donde se encuentra el proyecto en curso.
PutEnv	Este método permite añadir o modificar el valor de la variable de entorno que se le pasa como parámetro.
Random	Devuelve un número aleatorio dentro de un intervalo que se pasa como parámetro.
ReadDir	Este método lee el siguiente elemento del directorio cuyo manejador se pasa como parámetro.
Rgb	Este método calcula el valor entero correspondiente a un color dado en formato RGB.
RmDir	Este método permite borrar un directorio.
Run	Este método permite ejecutar otro programa Cosmos. Este método carga en memoria el programa y al finalizar su ejecución lo descarga.
ScreenSize	Este método retorna las dimensiones de la pantalla en pixeles.
SetCursorPos	Este método mueve el cursor del ratón a la posición indicada.
SetIni	Este método permite modificar el valor correspondiente a una entrada de una sección de un fichero de configuración.
SetOption	Este método permite definir el comportamiento de un módulo en ejecución
Sql	Cosmos tiene definido un servidor por defecto: Es un Objeto de la clase SqlServer al que se accede por medio de este método que puede ser utilizado en cualquier momento por el programador.

Método	Utilidad
Sleep	Este método detiene la ejecución del programa tantos segundos como se indique en el parámetro.
Start	Este método puede ser definido en cualquier módulo COOL y se ejecutará cuando se cargue el módulo en memoria.
System	Ejecuta un programa Windows o MS-DOS y espera a que termine.
TestFile	Este método comprueba si un fichero es de un tipo determinado o bien si tiene unos permisos concretos.
TreeWalk	Este método muestra un cuadro de diálogo que permite cambiar de directorio para elegir un fichero.
UnLoad	Este método permite descargar de memoria un módulo Cosmos.
WaitCursor	Este método permite cambiar el cursor del ratón.
WindowsDir	Este método indica el directorio donde se encuentra instalado Windows.
Yes	Este método muestra un cuadro de mensaje con los botones «Yes» y «No» para pedir una respuesta al usuario.
Yield	Este método cede el control a Windows para que procese los mensajes pendientes.

5. Creación del módulo mNuevo

Se añade una nueva opción en el menú principal del módulo Inicio que lanzará la ejecución de este nuevo módulo. (Figura 10.1)

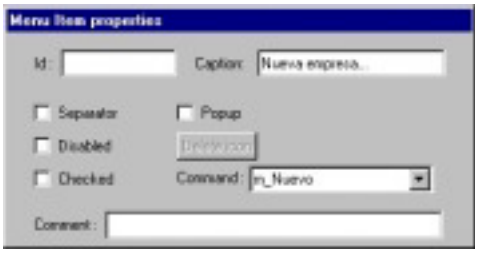


Figura 10.1. Nueva opción en el menú principal.

Se crea a continuación el nuevo módulo que creará una base de datos para una nueva empresa de la que se gestionarán los proyectos. Este módulo se creará dentro de la carpeta “Varios” que se realizó en el capítulo 9. En este módulo se inserta una clase llamada cNuevo que deriva de la clase Form de Cosmos. (Figura 10.2)

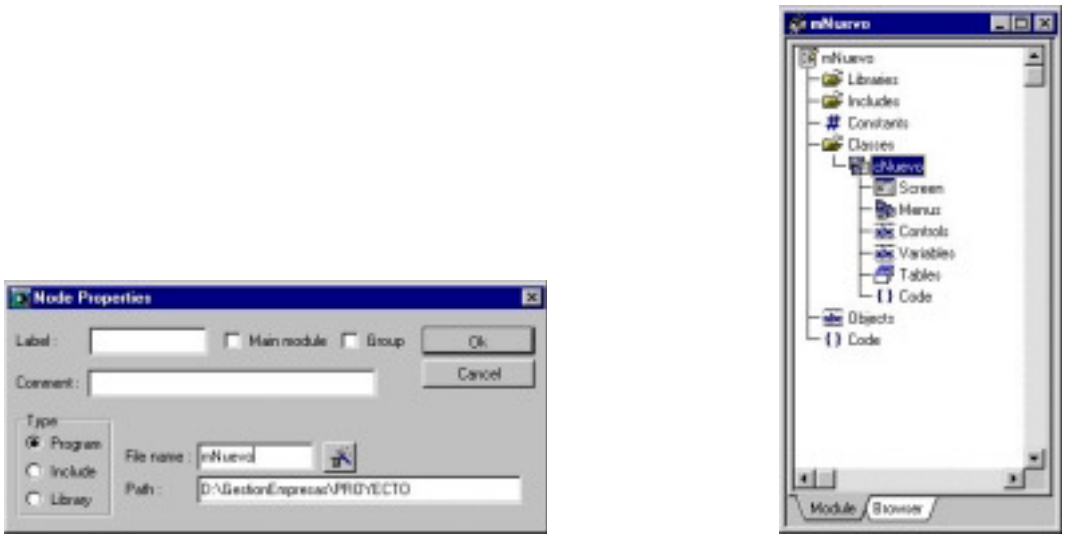


Figura 10.2. Nuevo módulo.

Se diseña la sección Screen de esta nueva clase para que solicite el nombre de la nueva empresa. (Figura 10.3)

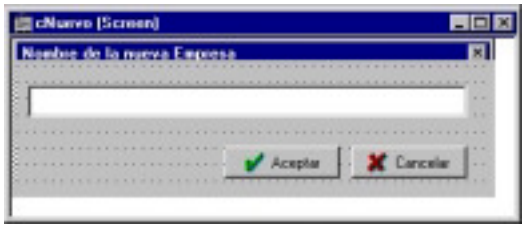


Figura 10.3. Sección Screen de la clase cNuevo.

El código de esta clase se compone de dos métodos, el que se ejecuta cuando se pulsa el botón Aceptar y un método que ejecuta la pantalla y que devuelve el nombre introducido sobre el objeto que lo invoca.

```
On command aceptar
begin
 // obtener el nombre introducido
 empresa=edit0.Text;
 if empresa<>""
  then Close;
  else Ok("Introduzca un nombre de empresa", "MENSAJE",true);
end

public nuevoNombre return char
begin
 // ejecutar la solicitud de datos
 Self.Run;
 // devolver el nombre introducido
 return empresa;
end
```

En el método “main” del módulo mNuevo se realiza la llamada al método “nuevoNombre” para obtener el nombre de la nueva empresa que se desea introducir. Después se ejecuta un método que crea la nueva base de datos, creando para ello el subdirectorio y copiando en él las tablas de la base de datos de plantilla que se guardan en el subdirectorio “datos.dbs”

```
main
objects
begin
 nombre as char
end
begin
// Se invoca la ejecución del método nuevoNombre
 nombre=oNuevo.nuevoNombre;
// Se llama al método crearNuevo
 crearNuevo(nombre);
// modificacion de las variables de entorno DBPATH y DBNAME
 SetIni("Environment ConexionGeneral","DBPATH",GetCwd,NULL);
 SetIni("Environment ConexionGeneral","DBNAME",nombre,NULL);
end


private crearNuevo(nombre as char)
objects
begin
 pathOrigen pathDestino as char
end
begin

 // situarse en el directorio del proyecto
 ChDir(ProjectDir);
 ChDir("..");
 pathOrigen=GetCwd+"\datos.dbs";
 pathDestino=GetCwd+"\"+nombre+".dbs";

 // crear el nuevo subdirectorio
 MkDir(pathDestino);

 // copiar la tablas base al nuevo subdirectorio
 if Copy(pathOrigen+"\*.*",pathDestino+"\")
  then ("Directorio " +pathDestino+ " creado con exito").Trace;
  else begin
      ("ERROR: el Directorio "+pathDestino+" no creado").Trace;
        RmDir(pathDestino);
       end
end
```

La ejecución de este módulo se muestra en la figura 10.4. Se debe introducir el nombre representativo de la empresa:

Figura 10.5. Ejecución del módulo mNuevo.

La ejecución de este módulo provoca la modificación de la entrada referida a la conexión denominada “Conexión general” en el fichero de configuración “Cosmos.ini” situado en el subdirectorio \etc\ del directorio donde se encuentra instalado Multibase Cosmos. Esta modificación se muestra a continuación.

```
[Environment ConexionGeneral]
DBPATH=D:\GestionEmpresas
DBNAME=Crisa
```

6. Creación del módulo mAbrir.

En este módulo se va a seleccionar una de las empresas que se gestionan para realizar su mantenimiento.

El primer paso es añadir una opción dentro del menú principal llamada “Abrir empresa...” cuya selección provoque la ejecución del módulo mAbrir. (Figura 10.5)

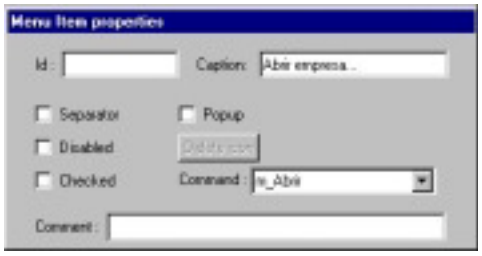


Figura 10.5. Añadir nueva opción en el menú principal.

Se añade a continuación el nuevo módulo llamado mAbrir de tipo Program. Dentro de este módulo se añade una clase derivada de la clase predefinida Form. Esta nueva clase tiene la función de solicitar al usuario la selección de la empresa que se desea abrir. El nombre de esta nueva clase es cSeleccion. (Figura 10.6)

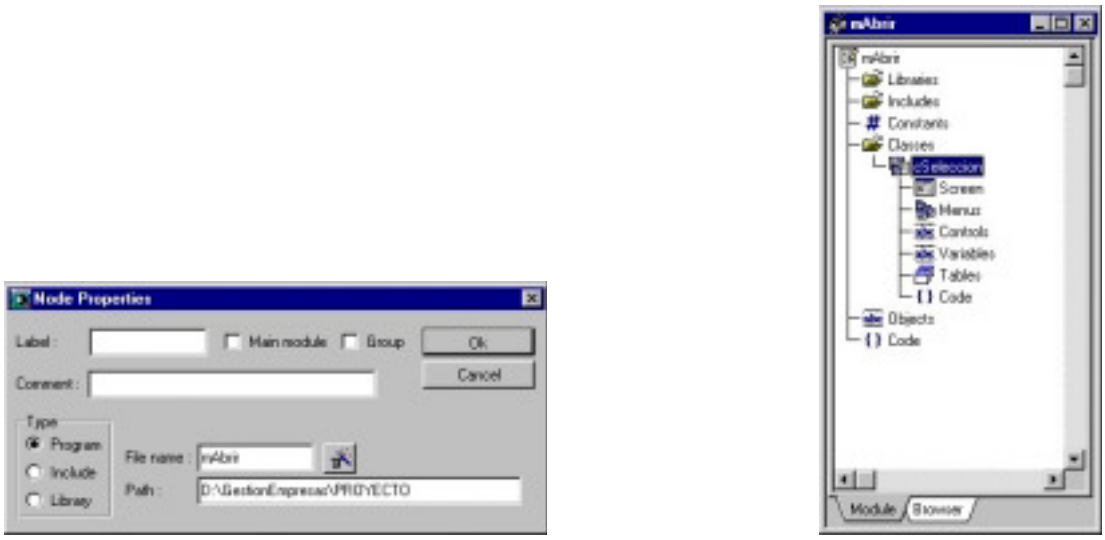


Figura 10.6. Añadir el nuevo módulo y su paleta.

La sección Screen de la clase cSeleccion se compone de un control de tipo List Box que mostrará el forma de árbol el contenido del directorio donde se almacenan las diferentes bases de datos de las empresas. Además se añaden dos Push buttons que generan la aceptación de la selección o la cancelación de la operación. (Figura 10.7)

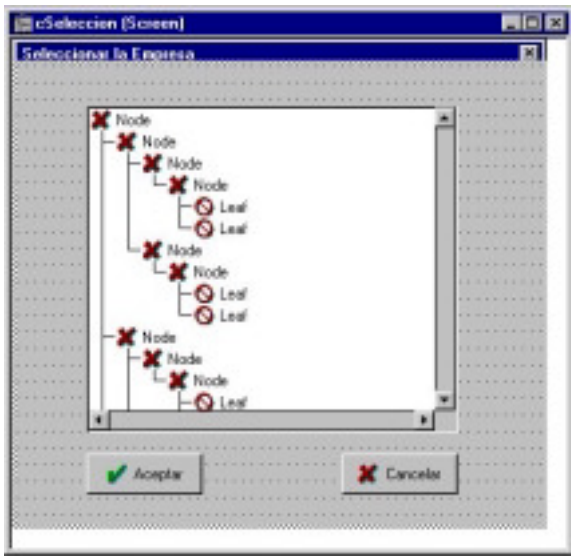


Figura 10.7. Aspecto de la sección Screen.

Las propiedades del control de tipo List Box se muestran en la figura 10.8. Se ha añadido un identificador al control para poder hacer referencia al mismo dentro del código de la clase. En la pestaña Special se ha seleccionado el tipo de lista que se desea mostrar.

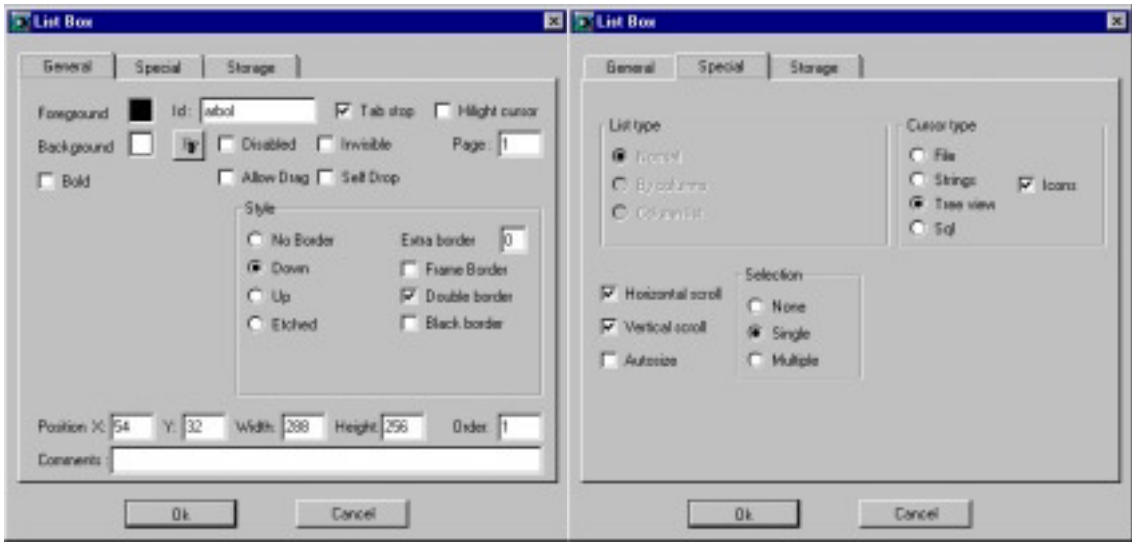


Figura 10.8. Propiedades del control de tipo List Box.

El código de esta clase se compone de tres métodos. El método On Open se ejecuta en el momento que se invoca la apertura de esta Form. En este método se han introducido las instrucciones necesarias para mostrar en el control de tipo List Box el contenido del directorio donde se encuentran almacenadas las bases de datos de las diferentes empresas. Este código se muestra a continuación

```
On Open
objects
begin
 fich as char
 err icono handle as integer
end
```

```
begin
 //se posiciona en el directorio padre del proyecto
 //donde se posicionan las diferentes bases de datos
 ChDir(ProjectDir);
 ChDir("..");
 //Añade al arbol el nodo del directorio padre
 arbol.AddTreeString(0,GetCwd,true,150,149);
 //abre el directorio almacenando el manejador asignado al mismo
 handle=OpenDir(GetCwd,fich);
 //realiza la lectura del contenido del directorio hasta el
 //que no existan ficheros o subdirectorios
 do
 begin
  if TestFile(fich,"f")
   then icono=164;
   else icono=149;
  //Añade al arbol el nodo del directorio
  arbol.AddTreeString(1,fich,false,150,icono);
  //lee el siguiente elemento del directorio
  err=ReadDir(handle,fich);
 end
 while err<>0;
end
```

Cuando se pulsa el botón “Cancelar” se cierra la ejecución de la pantalla de selección. Cuando se pulsa el botón “Aceptar” se ejecuta el método denominado con el mismo nombre.

```
On Command aceptar
objects
begin
 pos as smallint
end
begin
 //obtener el elemento seleccionado
 pos=arbol.Selected;
 //comprobar que se ha seleccionado un elemento
 if pos<>0
  then begin
        //obtener el nombre del elemento seleccionado
        nombre=arbol.GetListText(pos);
        //comprobar si es una base de datos
        if (nombre.Count(".dbs"))==0
         then Ok( "Debe seleccionar una base de datos",
                  "MENSAJE",true);
         else begin
               nombre.Replace(".dbs","",1);
               Close;
              end
       end
  else Ok("Debe seleccionar una base de datos o pulsar el botón"
          +" Nuevo", "MENSAJE",true);
end
```

Por último el método “cargarDir” que ejecuta la propia Form y devuelve el nombre de la base de datos seleccionada.

```
public function cargarDir return char
begin
 Self.Run;
 return nombre;
end
```

El código del módulo mAbrir se compone únicamente del método “Main”. En primer lugar se invoca la ejecución del método “cargarDir” de la clase cSeleccion. Una vez se tiene el nombre de la base de datos que se desea abrir, se modifica la variable de entorno correspondiente a “ConexionGeneral” para que se asigne el path y el nombre de la base de datos de la empresa seleccionada.

```
main
objects
begin
 dir as char
end
begin
 // Se invoca la ejecución del método cargarDir de la clase
 // cSeleccion
 dir=oSeleccion.cargarDir;
 // Modificación de las variables de entorno.
 SetIni("Environment ConexionGeneral","DBPATH",GetCwd,NULL);
 SetIni("Environment ConexionGeneral","DBNAME",dir,NULL);
end
```

La ejecución del nuevo módulo se muestra en la figura 10.9. Se muestra dentro del control “arbol” el contenido del directorio donde se encuentran las bases de datos de las diferentes empresas además del repositorio y el directorio donde se encuentra almacenado el proyecto en curso.

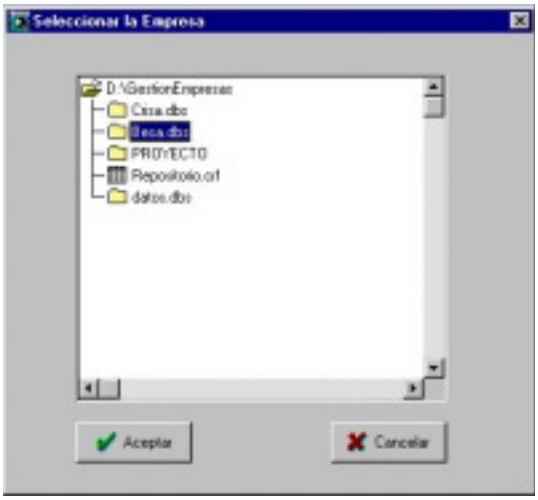


Figura 10.9. Ejecución del módulo mAbrir.

Al seleccionar la empresa “Besa” se modifica las variables de entorno DBPATH y DBNAME de la conexión denominada “ConexionGeneral” del fichero de configuración “Cosmos.ini”.

```
[Environment ConexionGeneral]
DBPATH=D:\GestionEmpresas
DBNAME=Besa
```

7. Creación del módulo mBorrar.

Este módulo borrará la base de datos correspondiente a una empresa gestionada por la aplicación.

El primer paso es añadir una nueva opción dentro del menú principal, denominada “Eliminar empresa...” que lanzará la ejecución del nuevo módulo. (Figura 10.10)

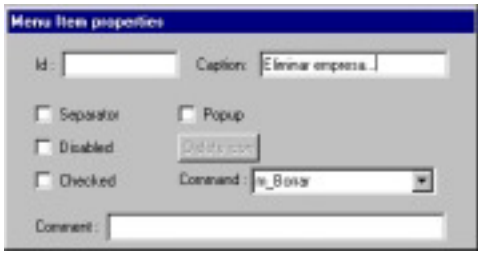


Figura 10.10. Añadir la nueva opción al menú.

Se crea un nuevo módulo dentro de la carpeta “Varios” llamado cBorrar. En este módulo se debe añadir una clase derivada de la clase predefinida Form para seleccionar la empresa que se desea borrar. Esta clase es la misma que la definida en el apartado 6 dentro del módulo mAbrir. En vez de definir una nueva clase se utilizará la ya definida. Para ello se selecciona dentro de la paleta del módulo mAbrir la clase cSeleccion y se arrastra sobre la sección Classes del módulo mBorrar. (Figura 10.11)

Figura 10.11. Arrastrar la clase cSeleccion del módulo mAbrir sobre el módulo mBorrar.

Se muestra una pantalla que nos permite modificar el nombre de la clase si se considera necesario. En este caso se mantiene el mismo nombre. El resultado es que el módulo mBorrar ya tiene definida la clase cSeleccion con todos sus métodos y propiedades. (Figura 10.12)

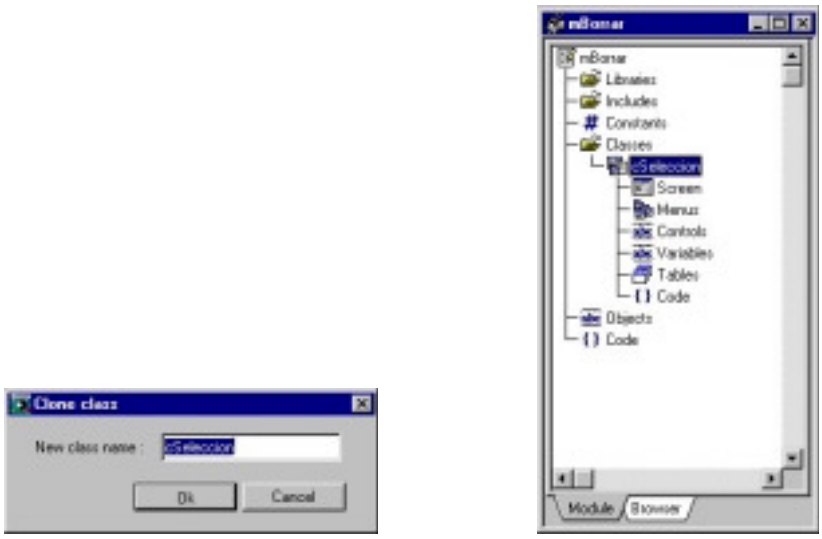


Figura 10.12. Aceptación del nombre y paleta del nuevo módulo.

El código de la clase cSeleccion se ha mantenido igual que en el módulo mAbrir. Desde el código del módulo mBorrar se hace la llamada al método “cargarDir” de esta clase dentro del método Main. Se solicita la confirmación del borrado de la base de datos y si es afirmativa la respuesta se ejecuta el método de acceso privado “borrarBD” que realiza el borrado efectivo de la base de datos de la empresa seleccionada.

```
main
objects
begin
 msg as boolean
 nombre as char
end
begin
// Se invoca la ejecución del método cargarDir de la clase
// cSeleccion
 nombre=oSeleccion.cargarDir;
//Solicitud de confirmación
 msg=Ok("Desea eliminar la empresa: "+nombre,
        "PRECAUCION",false);
 if msg then begin
   // invocacion al método borrarBD para realizar el borrado
   borrarBD(nombre);
   Ok("Empresa "+nombre+" borrada", "MENSAJE",true);
  end
 else begin
  Exit;
  Ok("Empresa "+nombre+" NO borrada", "MENSAJE",true);
 end
end
```

```
private borrarBD(nombre as char)
objects
begin
 path as char
end
begin
 // situarse en el directorio del proyecto
 ChDir(ProjectDir);
 ChDir("..");
 // almacenar el nombre de la base de datos a borrar
 path=GetCwd+"\"+nombre+".dbs";
 // borrar todos los ficheros del subdirectorio
 Delete(path+"\*.*");
 // borrar el subdirectorio
 RmDir(path);
end
```

La ejecución de este módulo es similar al del módulo mAbrir en cuanto a la selección de la empresa. Cuando se ha seleccionado una empresa, se muestra un mensaje donde se solicita la confirmación de borrado y finalmente se muestra un mensaje con la operación realizada. (Figura 10.13)

Figura 10.13. Borrado de una empresa.

8. Modificación del módulo mInicialización.

Este es el módulo que se diseño en el capítulo 9. Se va a modificar el nombre del menú principal que ejecuta este módulo. Para realizar esto seleccionamos la anterior entrada y modificamos el título por “Vaciar tablas...” (Figura 10.14)

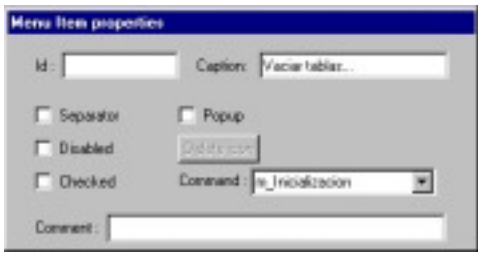


Figura 10.14. Modificación del nombre en el menú.

Para seleccionar el nombre de la empresa de la que se desea borrar los datos existentes, realizamos la copia de la clase cSeleccion desde el módulo mAbrir sobre el módulo mInicializacion, siguiendo los mismos pasos que en el apartado 7.

Dentro del código del módulo mInicialización se ha modificado el método Main para que se ejecute la selección de la empresa de la misma forma que en los apartados anteriores. El método “BorrarTablas” no se ha modificado.

```
main
objects
begin
 msg as boolean
 dir as char
end
begin
 // Se invoca la ejecución del método cargarDir de la clase
 // cSeleccion
 dir=oSeleccion.cargarDir;
 // modificar las variables de entorno de la ConexionGeneral
 SetIni("Environment ConexionGeneral","DBPATH",GetCwd,NULL);
 SetIni("Environment ConexionGeneral","DBNAME",dir,NULL);
 //conexion a la base de datos
 Sql.AttachConnection("ConexionGeneral");
 Sql.Connect;
 //Solicitud de confirmación
 msg=Ok("Desea vaciar el contenido de las tablas",
        "PRECAUCION",false);
 if msg then begin
   borrarTablas;
  end
 else begin
  Exit;
 end
 //desconexión de la base de datos
 Sql.Disconnect;
 Sql.DettachConnection;
end
```

De esta forma se pueden vaciar los datos de una empresa sin borrar la estructura de tablas de dicha empresa, es decir, manteniendo la base de datos.

9. Ejecución de la aplicación.

A continuación se muestra la ejecución completa de esta aplicación. En un principio no se tiene ninguna empresa creada, por tanto el primer paso es seleccionar la opción “Nueva empresa...” dentro del menú principal. (Figura 10.15)

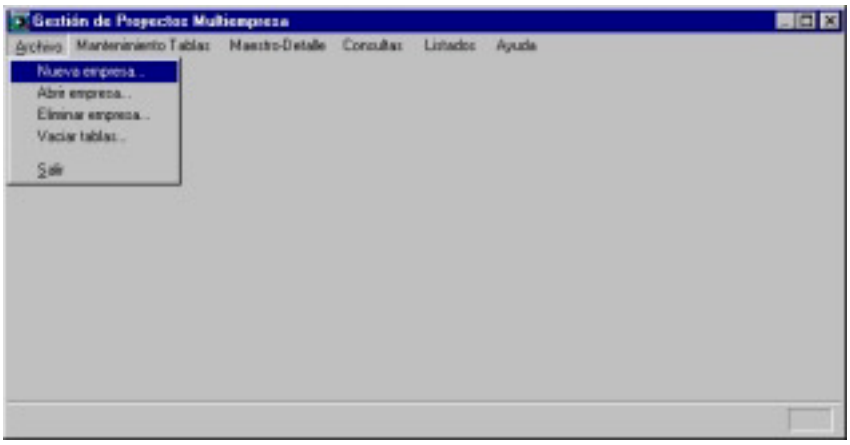


Figura 10.15. Ejecución de la aplicación.

Se muestra el cuadro de diálogo de la figura 10.16 donde se introduce el nombre de la nueva empresa, en este caso “Crisa”.

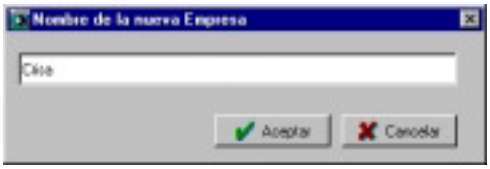


Figura 10.16. Introducir el nombre de la nueva empresa.

Como en el módulo mNuevo ya se modificaba el valor de las variables de entorno no es necesario seleccionar la opción “Abrir empresa...” por tanto ya podemos introducir datos a esta nueva empresa. Por ejemplo en la figura 10.17 se muestra la introducción de un nuevo Departamento.

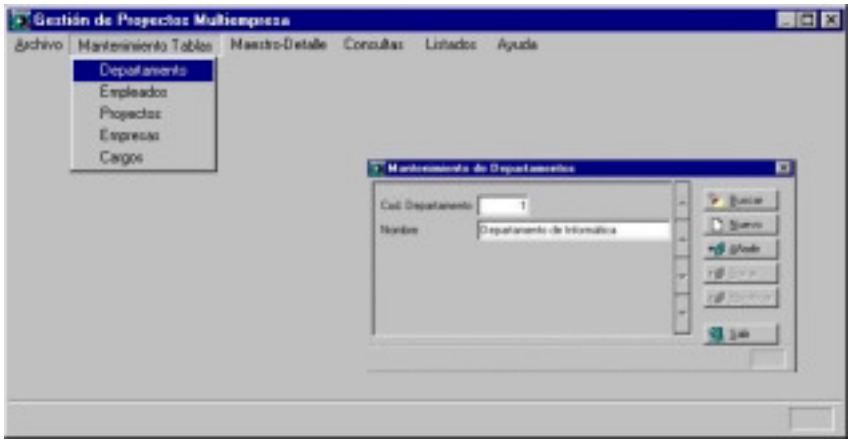


Figura 10.17. Añadir nuevo departamento.

Si se esta realizando la gestión de una empresa y se desea cambiar a otra empresa, se elegirá la opción “Abrir empresa...” y se selecciona la empresa a gestionar en este momento. (Figura 10.18)

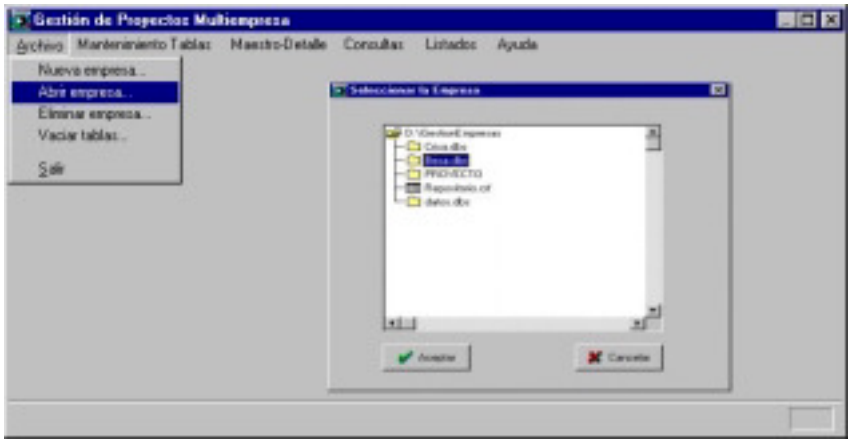


Figura 10.18. Abrir una nueva empresa.

En este momento ya se ha modificado las variables de entorno y se puede realizar la gestión de la nueva empresa.

Se ha añadido una opción de menú denominada Ayuda donde se muestra la entrada "Acerca de..." que muestra una pantalla con la información básica de la aplicación. (Figura 10.19)

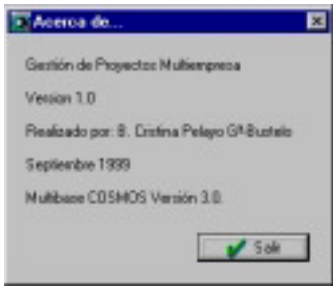


Figura 10.19. Cuadro de diálogo "Acerca de..."

10. Archivos generados en este capítulo.

Los archivos generados en este capítulo son:

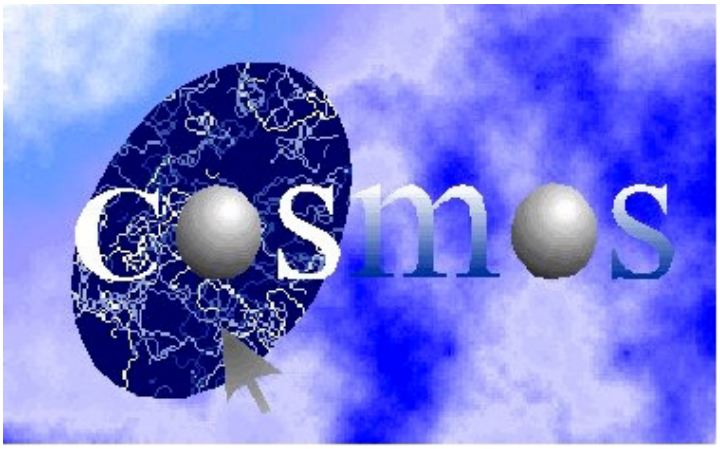
Archivo	Contenido
mAbrir.smd	Fichero que contiene el código del módulo para abrir una base de datos de una empresa.
mAbrir.omd	Fichero que contiene el código objeto del módulo mAbrir.
mNuevo.smd	Fichero que contiene el código del módulo para crear una base de datos de una empresa
mNuevo.omd	Fichero que contiene el código objeto del módulo mNuevo.
mBorrar.smd	Fichero que contiene el código del módulo para borrar una base de datos de una empresa
mBorrar.omd	Fichero que contiene el código objeto del módulo mBorrar.

El código completo de esta aplicación de gestión de proyectos multiempresas se muestra en el Anexo C.

ANEXO A

Templinc2.smd

1. Introducción.
2. Módulo Templinc2.
3. Código completo de Templinc2.smd.



1. Introducción.

A lo largo de los capítulos de este manual, se ha introducido un include denominado **templinc2**. En este anexo se explicará la funcionalidad de este include.

2. Módulo Templinc2.

En este módulo se han generado una serie de clases para almacenan la funcionalidad de todos los posibles informes que se pueden realizar mediante los Wizards de Cosmos. Existe una clase básica que deriva directamente de la clase ***Page*** de Cosmos y luego hay una serie de clases que cada una deriva de las definidas anteriormente. Lo vemos en la sección de ***Classes*** del código de este módulo:

```
CLASSES BEGIN
    basicPage is  abstract Page
    absPage is  abstract basicPage
    stdPage is  abstract absPage
    absHLPage is  abstract stdPage
    absHLLines is  abstract absHLPage
    absBandPage is  abstract basicPage
    absBandHLPage is  abstract absBandPage
    absBandHLLines is  abstract absBandHLPage
    previewForm is Form begin

      {... definición del interface de la clase previewForm ...}

    end

    FStdAbsEd is  abstract Form
    FStdAbs is  abstract Form
END
```

En la paleta de este módulo que se muestra en la figura A.1. se puede observar cada una de estas clases.

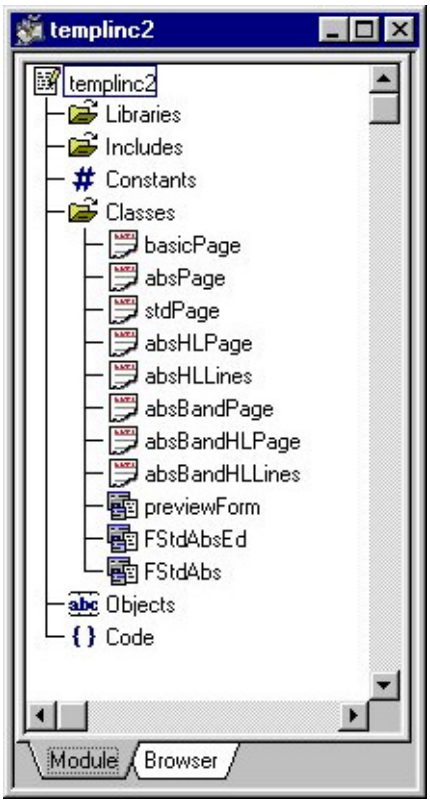


Figura A.1. Paleta del módulo templinc2.

Por ejemplo, la clase ***absPage*** deriva de la clase ***basicPage*** que a su vez deriva de la clase ***Page***, como se muestra en la figura A.2.

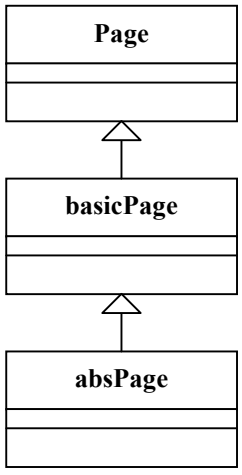


Figura A.2. Diagrama de herencia.

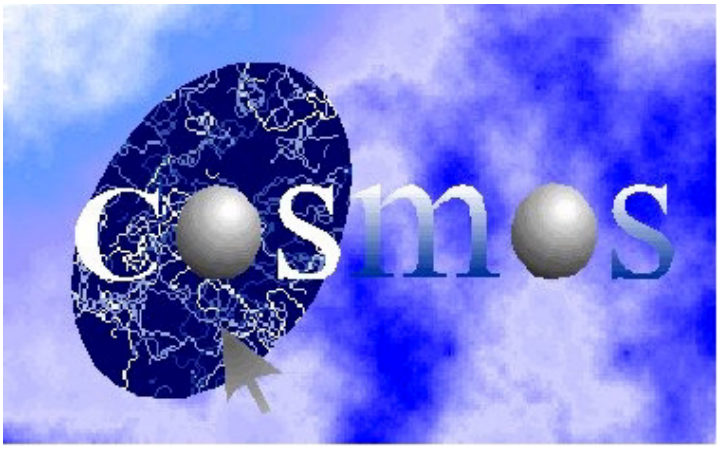
3. Código completo de templinc2.smd.

El código completo de este fichero se encuentra ubicado en la carpeta include dentro de la carpeta donde se encuentra instalado MultiBase COSMOS.

ANEXO B

Editor de Configuración y definición de conexiones

1. Introducción.
2. Fichero de configuración COSMOS.INI
3. Variables de Entorno.
4. Conectividad con Sistemas de Gestión de Bases de Datos.
5. Definir una conexión.
6. Código del Fichero COSMOS.INI



1. Introducción.

En este anexo se va realizar la definición de una conexión en el Editor de Configuración de Cosmos. A lo largo de este manual se han definido una serie de conexiones que se muestran en el código del fichero COSMOS.INI al final de este anexo.

2. Fichero de Configuración COSMOS.INI

El fichero de configuración contiene algunas de las características con las que deberán arrancar las aplicaciones Cosmos. Este fichero se encuentra en el subdirectorio etc del directorio donde se encuentre instalado Cosmos.

Este fichero está especialmente diseñado para:

- Evitar un número excesivo de parámetros en la línea de comando de las aplicaciones Cosmos.

- Facilitar el paso de variables de entorno entre aplicaciones.

- Facilitar el manejo de las aplicaciones Cosmos al evitar al usuario el tener que introducir un número excesivo de datos para poder acceder a ellas.

3. Variables de Entorno

Tanto el sistema operativo como Cosmos poseen unas variables de entorno, que se utilizan para personalizar el sistema.

En Cosmos estas variables sólo se pueden definir en el fichero de configuración "COSMOS.INI" mediante el editor de configuración.

Todas las variables de entorno, se tomarán del fichero de inicialización y no de las variables de entorno MS-DOS.

El siguiente cuadro recoge las variables de entorno atendiendo a su función.

- **De Conexión**.

DBCHARSET	DBNAME	DBPASSWD	DBUSER
DBHOST	DBPATH	DBSERVICE	STRANSDIR
XDBTEMP			

- **De Formato**

DBDATE	DBMONEY	DBTIME	

- **De EasyReport**

CRWPATH	GRWPATH	TRWPATH	TRWPRIV

- **CTSQL**

ISAMBUFS	MBISFILES	OUTOPT	OUTOPT2
OUTOPT3	SORTMEM		

- **Gateways**

DBEMBED	DBLONGCHAR	DBSERVER	DBSQL
DBPATH	DBSYN	II_SYSTEM	INFORMIXDIR
MBCOMMIT	MBLCKTIMEOUT	DB2INSTANCE	DB2_UID
INGRES_UID	INGRES_DBA	MBLOADCOMMIT	ORACLE_DECIMAL
ORACLE_HOME	ORACLE_PROC	ORACLE_SID	ORACLE_UID

- **Diversas**

DBDELIM	DBEDIT	DBQUOTED	DBTEMP
LANG			

4. Conectividad con Sistemas de Gestión de Bases de Datos

El Editor de Configuración permite definir conexiones locales, remotas (cliente-servidor) y ODBC para tener acceso a distintos servidores de base de datos.

Se da por lo tanto la posibilidad de establecer comunicaciones con otras bases de datos heterogéneas. Esto es posible gracias a la tecnología MultiWay soportada por el producto, que hace que el gestor SQL sea transparente a cualquier aplicación, ya sea ésta del propio Cosmos o bien del de otros fabricantes (Oracle, Informix, Ingres, Sybase, etc.).

5. Definir una conexión.

Cuando se ejecuta el Editor de Configuración de Cosmos se muestra la pestaña **Basic** que muestra el usuario actual que esta definido por defecto, en este caso unicamente existe un usuario que es el llamado **System**. (Figura B.1)

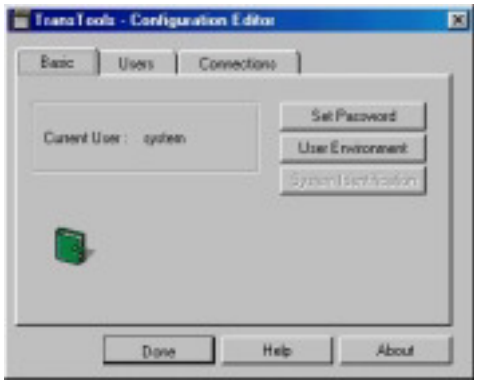


Figura B.1. Aspecto de la pestaña Basic del Editor de Configuración

En la pestaña Users se encuentra la definición de los usuarios actualmente definidos para la utilización de las diferentes aplicaciones Cosmos. (Figura B.2)

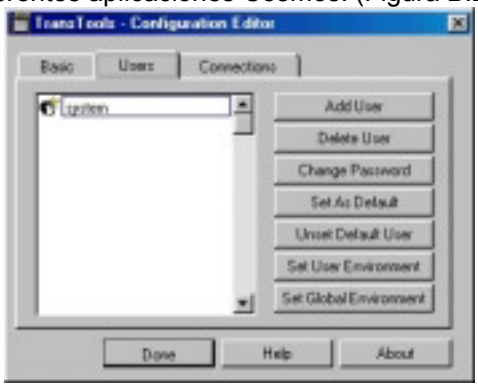


Figura B.2. Pestaña Users del Editor de Configuración.

En la pestaña Connections se pueden definir las diferentes conexiones que se utilizarán. A lo largo de este manual se han definido cuatro diferentes conexiones que se muestran en la figura B.3.

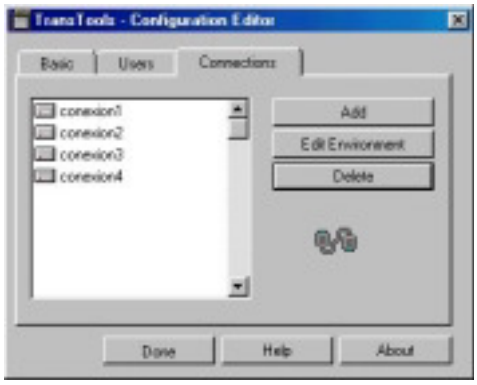


Figura B.3. Conexiones del manual.

Para definir una conexión nueva se debe pulsar el botón 

se muestra un cuadro de diálogo donde se debe introducir el nombre de esa nueva conexión y el tipo de la misma. En el caso de las conexiones que se encuentran definidas en el manual se ha utilizado servidores locales. (Figura B.4)

Figura B.4. Añadir una nueva conexión.

Cuando se pulsa el botón 

, se muestra el cuadro de diálogo donde se introducen las variables de entorno relativas a la conexión que se ha creado. Unicamente se han definido dos variables de entorno en las conexiones existentes: DBNAME y DBPATH. En la primera se asocia el nombre de una base de datos y en la segunda se indica la ubicación de dicha base de datos. (Figura B.5)

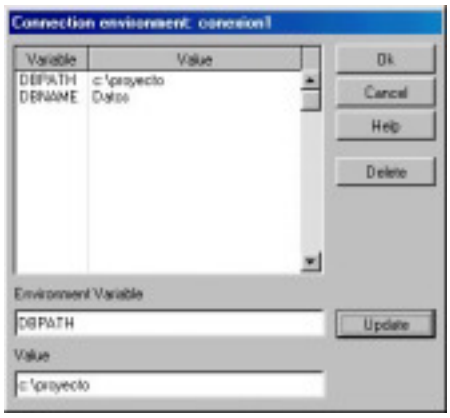


Figura B.5. Variables de entorno de la conexión1.

6. Código del fichero COSMOS.INI

Todas las modificaciones que se han realizado en el Editor de Configuración quedan reflejadas en este fichero. El código completo del mismo se muestra a continuación.

```
[Passwords]
system=010XQ\~ryh~ptb~

[Cosmos]
USER=system
Licensed Company=Universidad de Oviedo
Licensed User=B. Cristina Pelayo Garcia-Bustelo

[DOCTEMPLATES]
Indice de proyecto
standard=PRJ,docu\Templates\standard\prjindex.tpl
Proyecto standard=PRJ,docu\templates\standard\project.tpl
Repositorio
standard=CRF,docu\templates\standard\reposito.tpl
Indice de repositorio
standard=CRF,docu\templates\standard\repindex.tpl
Tabla standard=TAB,docu\templates\standard\table.tpl
Modulo standard=SMD,docu\templates\standard\source.tpl

[DOCUMENTATION]
SeparateTables=TRUE
CreateTablesDir=TRUE
PRJ=Indice de proyecto standard,.htm,
SMD=Modulo standard,.htm,
CRF=Indice de repositorio standard,.htm,R
TAB=Tabla standard,.htm,
CreateSourcesDir=FALSE
DOCUDIR=docu
IMAGEDIR=docu\images
SELCONF=Standard

[DOCCONFIGURATIONS]
Standard=
Standard (dividido)=

[DOCUMENTATION Standard]
SeparateTables=TRUE
```

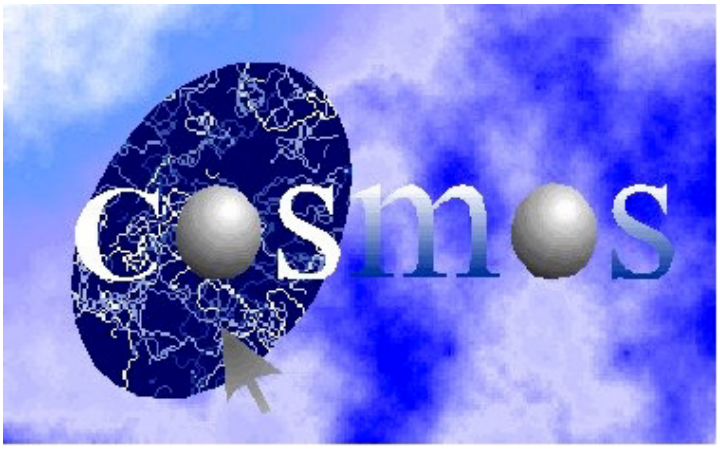
```
CreateTablesDir=TRUE
PRJ=Indice de proyecto standard,.htm,
SMD=Modulo standard,.htm,
CRF=Indice de repositorio standard,.htm,R
TAB=Tabla standard,.htm,
CreateSourcesDir=FALSE
DOCUDIR=docu
IMAGEDIR=

[DOCUMENTATION Standard (dividido)]
SeparateTables=FALSE
CreateTablesDir=FALSE
PRJ=Indice de proyecto standard,.htm,
SMD=Modulo standard,.htm,
CRF=Indice de repositorio standard,.htm,R
TAB=Tabla standard,,
CreateSourcesDir=FALSE
DOCUDIR=docu
IMAGEDIR=

[Environment]
DBTEMP=c:\tmp

[Icons]
Cosmos Icons=cosicons.BMP,   32,  32, RGB(192,192,192)
Small Buttons=tbu16x15.bmp,  16,  15, RGB(192,192,192)
Mini Buttons=TBUT8X8.BMP,    8,   8, RGB(192,192,192)
List Icons=listicon.bmp,  16,  15, RGB(255,255,255)
More List Icons=jcllist.bmp,  16,  15, RGB(255,255,255)
Icons Office=ICOOFFIC.BMP,  32,  32, RGB(192,192,192)
Country Flags=ICOFLAGS.BMP,  32,  20, RGB(192,192,192)
Icon Message Symbols=ICOSIMBO.BMP,  32,  32,
RGB(192,192,192)
Icons Arrows=ICOARROW.BMP,  32,  32, RGB(192,192,192)
Icon Hardware=ICOHARD.BMP,   32,  32, RGB(192,192,192)
Windows 95 Icons=IcoWin95.BMP,   32,  32, RGB(  0,128,128)

[Connections]
conexion1=local
conexion2=local
conexion3=local
conexion4=local

[Environment conexion1]
DBPATH=c:\Programa
DBNAME=Datos

[Environment conexion2]
DBPATH=c:\Programa2
DBNAME=datos2

[Repository Manager]
File1=C:\Programa4\Repositorio4.crf
File2=C:\Programa3\Repositorio3.crf
Table Labels=FALSE
Column Labels=FALSE

[Environment conexion3]
DBPATH=c:\Programa3
DBNAME=datos3
```

```
[Environment conexion4]
DBPATH=c:\Programa4
DBNAME=datos4
```

ANEXO C

Código completo de la aplicación "Gestión de Proyectos Multiempresas"

1. Introducción.
2. Módulo mInicio.
3. Módulo ABM_dpto.
4. Módulo ABM_empleados.
5. Módulo ABM_empresas.
6. Módulo ABM_proyectos.
7. Módulo ABM_cargos.
8. Módulo md_dpto_empleados.
9. Módulo md_empleado_pytos.
10. Módulo md_pytos_cargos.
11. Módulo md_empresas_pyto.
12. Módulo md_dpto_pytos.
13. Módulo mCargos.
14. Módulo mEmpDpto.
15. Módulo mInicializacion.
16. Módulo mAbrir.
17. Módulo mNuevo.
18. Módulo mBorrar.
19. Módulo mAcerca.



1. Introducción.

En este anexo se muestra el código completo de la aplicación que se desarrolla a lo largo de los capítulos 5 al 10. El código se muestra desglosado en los diferentes módulos que componen la aplicación completa que se termino de definir en el capítulo 10.

2. Módulo mInicio.

```
REPOSITORY repositorio
CONSTANTS begin
end
CLASSES BEGIN
cInicio is Form begin
default menu menuPpal
objects begin
end
menu menuPpal
"mainMenu"
PULLDOWN
BEGIN
MENU
"&Archivo"
BEGIN
OPTION
"Nueva empresa..."
COMMAND m_Nuevo
OPTION
"Abrir empresa..."
COMMAND m_Abrir
OPTION
"Eliminar empresa..."
COMMAND m_Borrar
OPTION
"Vaciar tablas..."
COMMAND m_Inicializacion
OPTION
"&Salir"
SEPARATOR
COMMAND Close
END
MENU
"Mantenimiento Tablas"
ALLOWCHECK
BEGIN
OPTION
"Departamento"
COMMAND mABMdpto
OPTION
"Empleados"
COMMAND mABMempleados
OPTION
"Proyectos"
COMMAND mABMproyectos
OPTION
"Empresas"
COMMAND mABMempresas
```

```
OPTION
"Cargos"
COMMAND mABMcargos
END
MENU
"Maestro-Detalle"
BEGIN
OPTION
"Departamento-Empleados"
COMMAND mDptoEmpleados
OPTION
"Empresas-Proyectos"
COMMAND mEmpresasPyto
OPTION
"Empleados-Proyectos"
COMMAND mEmpleadoPyto
OPTION
"Departamento-Proyectos"
COMMAND mDptoPytos
OPTION
"Cargos-Proyectos"
COMMAND mPytosCargos
END
MENU
"Consultas"
BEGIN
OPTION
"Cargos de un proyecto"
COMMAND m_Cargos
END
MENU
"Listados"
BEGIN
OPTION
"Empleados de un Departamento"
COMMAND m_EmpDpto
END
MENU
"Ayuda"
BEGIN
OPTION
"Acerca de..."
COMMAND m_Acerca
END
END
INTERFACE
POSITION 0 0 658 339
LABEL "Gestión de Proyectos Multiempresa"
MAXIMIZE
SYSMENU
BEGIN
CONTROL AS BOX
POSITION 0 0 650 268
FOREGROUND RGB 0 0 0
ATTACH ALL
NOLABEL
NOBORDER
BEGIN
END
CONTROL AS BAR
POSITION 0 268 650 27
```

```
FOREGROUND RGB 0 0 0
ATTACH LEFT RIGHT BOTTOM
BORDER ETCHED EXTRA 3
BARTYPE STATUS
BEGIN.CONTROL AS PANEL
POSITION 3 0 600 19
FOREGROUND RGB 0 0 0
ATTACH ALL
NOBORDER
PANELTYPE COMMENT
CONTROL AS PANEL
POSITION 606 0 35 19
FOREGROUND RGB 0 0 0
ATTACH TOP RIGHT BOTTOM
BORDER DOWN
PANELTYPE EDITMODE
END
END
end
END
CODE CLASS cInicio BEGIN
//{{CODEBEGIN
On Command mABMdpto
begin
module.Run("ABM_dpto");
end
On Command mABMempleados
begin
module.Run("ABM_empleados");
end
On Command mABMempresas
begin
module.Run("ABM_empresas");
end
On Command mABMproyectos
begin
module.Run("ABM_proyectos");
end
On Command mABMcargos
begin
module.Run("ABM_cargos");
end
On Command mDptoEmpleados
begin
module.Run("md_dpto_empleados");
end
On Command mEmpleadoPyto
begin
module.Run("md_empleado_pytos");
end
On Command mPytosCargos
begin
module.Run("md_pytos_cargos");
end
On Command mEmpresasPyto
begin
module.Run("md_empresas_ptyo");
end
On Command mDptoPytos
begin
module.Run("md_dpto_pytos");.end
```

```
On Command m_Cargos
begin
module.Run("mCargos");
end
On Command m_EmpDpto
begin
module.Run("mEmpDpto");
end
On Command m_Nuevo
begin
module.Run("mNuevo");
end
On Command m_Abrir
begin
module.Run("mAbrir");
end
On Command m_Borrar
begin
module.Run("mBorrar");
end
On Command m_Inicializacion
begin
module.Run("mInicializacion");
end
On Command m_Acerca
begin
module.Run("mAcerca");
end
//{{CODEEND
END
CODE BEGIN
//{{CODEBEGIN
main
objects
begin
oCInicio as cInicio
end
begin
oCInicio.Run;
end
//{{CODEEND
END
```

3. Módulo ABM_dpto.

```
REPOSITORY repositorio
CONSTANTS begin
end
CLASSES BEGIN
CABM_dpto is Form begin
objects begin
end
Table departamentos is departamentos
begin
cod_dpto as column required
nombre as column
cod_pyto as column
end
```

```
INTERFACE
POSITION 2 0 437 231
LABEL "Mantenimiento de Departamentos"
SYSMENU
BEGIN
CONTROL AS BOX
POSITION 0 0 429 179
FOREGROUND RGB 0 0 0
ATTACH ALL
NOLABEL
NOBORDER
BEGIN
CONTROL AS BUTTON
POSITION 340 13 76 22
ATTACH RIGHT
LABEL "&Buscar"
ICON 6 FILE "Small Buttons"
COMMAND QueryByForm
CONTROL AS BUTTON
POSITION 340 65 76 22
ATTACH RIGHT
LABEL "&Añadir"
ICON 157 FILE "Small Buttons"
COMMAND Add
CONTROL AS BUTTON
POSITION 340 91 76 22
ATTACH RIGHT
LABEL "&Borrar"
ICON 159 FILE "Small Buttons"
COMMAND Delete
CONTROL AS BUTTON
POSITION 340 117 76 22
ATTACH RIGHT
LABEL "&Modificar"
ICON 158 FILE "Small Buttons"
COMMAND Update
CONTROL AS BUTTON
POSITION 340 39 76 22
ATTACH RIGHT
LABEL "&Nuevo"
ICON 84 FILE "Small Buttons"
COMMAND New
CONTROL AS BUTTON
POSITION 340 156 76 22.ATTACH RIGHT
LABEL "&Salir"
ICON 4 FILE "Small Buttons"
COMMAND Close
CONTROL IDMASTER AS BOX
POSITION 4 4 305 165
ATTACH ALL
NOLABEL
BORDER DOUBLE DOWN
TABLE departamentos
BEGIN
CONTROL AS TEXT
POSITION 7 13 92 22
NOBORDER
DATATYPE CHAR
TAGS "Cod. Departamento"
CONTROL AS EDIT
POSITION 105 13 52 22
```

```
LABEL "Cod. Departamento"
BORDER DOUBLE DOWN
VARIABLE departamentos.cod_dpto
LIKEVAR
CONTROL AS TEXT
POSITION 7 39 37 22
NOBORDER
DATATYPE CHAR
TAGS "Nombre"
CONTROL AS EDIT
POSITION 105 39 196 22
LABEL "Nombre"
BORDER DOUBLE DOWN
VARIABLE departamentos.nombre
LIKEVAR
END
CONTROL AS SPIN
POSITION 309 4 20 165
ATTACH TOP RIGHT BOTTOM
BORDER DOUBLE DOWN
VERTICAL
QUADRUPLE
COMMAND
END
CONTROL AS BAR
POSITION 0 179 429 27
FOREGROUND RGB 0 0 0
ATTACH LEFT RIGHT BOTTOM
BORDER ETCHED EXTRA 3
BARTYPE STATUS
BEGIN
CONTROL AS PANEL
POSITION 3 0 379 19
FOREGROUND RGB 0 0 0
ATTACH ALL
NOBORDER
PANELTYPE COMMENT
CONTROL AS PANEL
POSITION 385 0 35 19.FOREGROUND RGB 0 0 0
ATTACH TOP RIGHT BOTTOM
BORDER DOWN
PANELTYPE EDITMODE
END
END
end
END
CODE CLASS CABM_dpto BEGIN
//{{CODEBEGIN
//{{CODEEND
END
CODE BEGIN
//{{CODEBEGIN
main
objects
begin
oCABM_dpto as CABM_dpto
end
begin
Sql.AttachConnection("ConexionGeneral");
Sql.Connect;
oCABM_dpto.Run;
```

```
Sql.Disconnect;
Sql.DettachConnection;
end
//{{CODEEND
END
```

4. Módulo ABM_empleados.

```
REPOSITORY repositorio
CONSTANTS begin
end
CLASSES BEGIN
CABM_empleados is Form begin
objects begin
end
Table empleados is empleados
begin
dni as column required
nombre as column
apellidos as column
domicilio as column
localidad as column
cod_postal as column
telefono as column
cod_dpto as column
cod_pyto as column
end
INTERFACE
POSITION 0 0 587 288
LABEL "Mantenimiento de Empleados"
SYSMENU
BEGIN
CONTROL AS BOX
POSITION 0 0 579 236
FOREGROUND RGB 0 0 0
ATTACH ALL
NOLABEL
NOBORDER
BEGIN
CONTROL AS BUTTON
POSITION 490 13 76 22
ATTACH RIGHT
LABEL "&Buscar"
ICON 6 FILE "Small Buttons"
COMMAND QueryByForm
CONTROL AS BUTTON
POSITION 490 65 76 22
ATTACH RIGHT
LABEL "&Añadir"
ICON 157 FILE "Small Buttons"
COMMAND Add
CONTROL AS BUTTON
POSITION 490 91 76 22
ATTACH RIGHT
LABEL "&Borrar"
ICON 159 FILE "Small Buttons"
COMMAND Delete
CONTROL AS BUTTON
```

```
POSITION 490 117 76 22
ATTACH RIGHT
LABEL "&Modificar"
ICON 158 FILE "Small Buttons"
COMMAND Update
CONTROL AS BUTTON
POSITION 490 39 76 22.ATTACH RIGHT
LABEL "&Nuevo"
ICON 84 FILE "Small Buttons"
COMMAND New
CONTROL AS BUTTON
POSITION 490 156 76 22
ATTACH RIGHT
LABEL "&Salir"
ICON 4 FILE "Small Buttons"
COMMAND Close
CONTROL IDMASTER AS BOX
POSITION 4 4 455 222
ATTACH ALL
NOLABEL
BORDER DOUBLE DOWN
TABLE empleados
BEGIN
CONTROL AS TEXT
POSITION 7 13 50 22
NOBORDER
DATATYPE CHAR
TAGS "DNI y letra"
CONTROL AS EDIT
POSITION 105 13 70 22
LABEL "DNI y letra"
BORDER DOUBLE DOWN
VARIABLE empleados.dni
LIKEVAR
CONTROL AS TEXT
POSITION 7 39 37 22
NOBORDER
DATATYPE CHAR
TAGS "Nombre"
CONTROL AS EDIT
POSITION 105 39 166 22
LABEL "Nombre"
BORDER DOUBLE DOWN
VARIABLE empleados.nombre
LIKEVAR
CONTROL AS TEXT
POSITION 7 65 42 22
NOBORDER
DATATYPE CHAR
TAGS "Apellidos"
CONTROL AS EDIT
POSITION 105 65 196 22
LABEL "Apellidos"
BORDER DOUBLE DOWN
VARIABLE empleados.apellidos
LIKEVAR
CONTROL AS TEXT
POSITION 7 91 42 22
NOBORDER
DATATYPE CHAR
TAGS "Domicilio"
```

```
CONTROL AS EDIT
POSITION 105 91 196 22.LABEL "Domicilio"
BORDER DOUBLE DOWN
VARIABLE empleados.domicilio
LIKEVAR
CONTROL AS TEXT
POSITION 7 117 46 22
NOBORDER
DATATYPE CHAR
TAGS "Localidad"
CONTROL AS EDIT
POSITION 105 117 196 22
LABEL "Localidad"
BORDER DOUBLE DOWN
VARIABLE empleados.localidad
LIKEVAR
CONTROL AS TEXT
POSITION 7 143 54 22
NOBORDER
DATATYPE CHAR
TAGS "Cod. Postal"
CONTROL AS EDIT
POSITION 105 143 94 22
LABEL "Cod. Postal"
BORDER DOUBLE DOWN
VARIABLE empleados.cod_postal
LIKEVAR
CONTROL AS TEXT
POSITION 7 169 42 22
NOBORDER
DATATYPE CHAR
TAGS "Teléfono"
CONTROL AS EDIT
POSITION 105 169 94 22
LABEL "Teléfono"
BORDER DOUBLE DOWN
VARIABLE empleados.telefono
LIKEVAR
CONTROL AS TEXT
POSITION 7 195 92 22
NOBORDER
DATATYPE CHAR
TAGS "Cod. Departamento"
CONTROL AS EDIT
POSITION 105 195 52 22
LABEL "Cod. Departamento"
BORDER DOUBLE DOWN
VARIABLE empleados.cod_dpto
LIKEVAR
END
CONTROL AS SPIN
POSITION 459 4 20 222
ATTACH TOP RIGHT BOTTOM
BORDER DOUBLE DOWN
VERTICAL
QUADRUPLE
COMMAND
END.CONTROL AS BAR
POSITION 0 236 579 27
FOREGROUND RGB 0 0 0
ATTACH LEFT RIGHT BOTTOM
```

```
BORDER ETCHED EXTRA 3
BARTYPE STATUS
BEGIN
CONTROL AS PANEL
POSITION 3 0 529 19
FOREGROUND RGB 0 0 0
ATTACH ALL
NOBORDER
PANELTYPE COMMENT
CONTROL AS PANEL
POSITION 535 0 35 19
FOREGROUND RGB 0 0 0
ATTACH TOP RIGHT BOTTOM
BORDER DOWN
PANELTYPE EDITMODE
END
END
end
END
CODE CLASS CABM_empleados BEGIN
//{{CODEBEGIN
//{{CODEEND
END
CODE BEGIN
//{{CODEBEGIN
main
objects
begin
oCABM_empleados as CABM_empleados
end
begin
Sql.AttachConnection("ConexionGeneral");
Sql.Connect;
oCABM_empleados.Run;
Sql.Disconnect;
Sql.DettachConnection;
end
//{{CODEEND
END
```

5. Módulo ABM_empresas.

```
REPOSITORY repositorio
CONSTANTS begin
end
CLASSES BEGIN
CABM_empresas is Form begin
objects begin
end
Table empresas is empresas
begin
cod_empresa as column required
nombre as column
domicilio as column
localidad as column
cod_postal as column
telefono as column
```

```
fax as column
e_mail as column
pag_web as column
end
INTERFACE
POSITION 0 0 739 288
LABEL "Mantenimiento de Empresas"
SYSMENU
BEGIN
CONTROL AS BOX
POSITION 0 0 731 236
FOREGROUND RGB 0 0 0
ATTACH ALL
NOLABEL
NOBORDER
BEGIN
CONTROL AS BUTTON
POSITION 642 13 76 22
ATTACH RIGHT
LABEL "&Buscar"
ICON 6 FILE "Small Buttons"
COMMAND QueryByForm
CONTROL AS BUTTON
POSITION 642 65 76 22
ATTACH RIGHT
LABEL "&Añadir"
ICON 157 FILE "Small Buttons"
COMMAND Add
CONTROL AS BUTTON
POSITION 642 91 76 22
ATTACH RIGHT
LABEL "&Borrar"
ICON 159 FILE "Small Buttons"
COMMAND Delete
CONTROL AS BUTTON
POSITION 642 117 76 22
ATTACH RIGHT
LABEL "&Modificar"
ICON 158 FILE "Small Buttons"
COMMAND Update
CONTROL AS BUTTON
POSITION 642 39 76 22.ATTACH RIGHT
LABEL "&Nuevo"
ICON 84 FILE "Small Buttons"
COMMAND New
CONTROL AS BUTTON
POSITION 642 156 76 22
ATTACH RIGHT
LABEL "&Salir"
ICON 4 FILE "Small Buttons"
COMMAND Close
CONTROL IDMASTER AS BOX
POSITION 4 4 607 222
ATTACH ALL
NOLABEL
BORDER DOUBLE DOWN
TABLE empresas
BEGIN
CONTROL AS TEXT
POSITION 7 13 66 22
NOBORDER
```

```
DATATYPE CHAR
TAGS "Cod. Empresa"
CONTROL AS EDIT
POSITION 79 13 52 22
LABEL "Cod. Empresa"
BORDER DOUBLE DOWN
VARIABLE empresas.cod_empresa
LIKEVAR
CONTROL AS TEXT
POSITION 7 39 37 22
NOBORDER
DATATYPE CHAR
TAGS "Nombre"
CONTROL AS EDIT
POSITION 79 39 196 22
LABEL "Nombre"
BORDER DOUBLE DOWN
VARIABLE empresas.nombre
LIKEVAR
CONTROL AS TEXT
POSITION 7 65 42 22
NOBORDER
DATATYPE CHAR
TAGS "Domicilio"
CONTROL AS EDIT
POSITION 79 65 196 22
LABEL "Domicilio"
BORDER DOUBLE DOWN
VARIABLE empresas.domicilio
LIKEVAR
CONTROL AS TEXT
POSITION 7 91 46 22
NOBORDER
DATATYPE CHAR
TAGS "Localidad"
CONTROL AS EDIT
POSITION 79 91 196 22.LABEL "Localidad"
BORDER DOUBLE DOWN
VARIABLE empresas.localidad
LIKEVAR
CONTROL AS TEXT
POSITION 7 117 54 22
NOBORDER
DATATYPE CHAR
TAGS "Cod. Postal"
CONTROL AS EDIT
POSITION 79 117 94 22
LABEL "Cod. Postal"
BORDER DOUBLE DOWN
VARIABLE empresas.cod_postal
LIKEVAR
CONTROL AS TEXT
POSITION 7 143 42 22
NOBORDER
DATATYPE CHAR
TAGS "Teléfono"
CONTROL AS EDIT
POSITION 79 143 94 22
LABEL "Teléfono"
BORDER DOUBLE DOWN
VARIABLE empresas.telefono
```

```
LIKEVAR
CONTROL AS TEXT
POSITION 7 169 17 22
NOBORDER
DATATYPE CHAR
TAGS "Fax"
CONTROL AS EDIT
POSITION 79 169 94 22
LABEL "Fax"
BORDER DOUBLE DOWN
VARIABLE empresas.fax
LIKEVAR
CONTROL AS TEXT
POSITION 7 195 28 22
NOBORDER
DATATYPE CHAR
TAGS "E-mail"
CONTROL AS EDIT
POSITION 79 195 196 22
LABEL "E-mail"
BORDER DOUBLE DOWN
VARIABLE empresas.e_mail
LIKEVAR
CONTROL AS TEXT
POSITION 282 13 59 22
NOBORDER
DATATYPE CHAR
TAGS "Página Web"
CONTROL AS EDIT
POSITION 347 13 256 22
LABEL "Página Web"
BORDER DOUBLE DOWN.VARIABLE empresas.pag_web
LIKEVAR
END
CONTROL AS SPIN
POSITION 611 4 20 222
ATTACH TOP RIGHT BOTTOM
BORDER DOUBLE DOWN
VERTICAL
QUADRUPLE
COMMAND
END
CONTROL AS BAR
POSITION 0 236 731 27
FOREGROUND RGB 0 0 0
ATTACH LEFT RIGHT BOTTOM
BORDER ETCHED EXTRA 3
BARTYPE STATUS
BEGIN
CONTROL AS PANEL
POSITION 3 0 681 19
FOREGROUND RGB 0 0 0
ATTACH ALL
NOBORDER
PANELTYPE COMMENT
CONTROL AS PANEL
POSITION 687 0 35 19
FOREGROUND RGB 0 0 0
ATTACH TOP RIGHT BOTTOM
BORDER DOWN
PANELTYPE EDITMODE
```

```
END
END
end
END
CODE BEGIN
//{{CODEBEGIN
main
objects
begin
oCABM_empresas as CABM_empresas
end
begin
Sql.AttachConnection("ConexionGeneral");
Sql.Connect;
oCABM_empresas.Run;
Sql.Disconnect;
Sql.DettachConnection;
end
//{{CODEEND
END
```

6. Módulo ABM_proyectos.

```
REPOSITORY repositorio
CONSTANTS begin
end
CLASSES BEGIN
CABM_proyectos is Form begin
objects begin
end
Table proyectos is proyectos
begin
cod_pyto as column required
titulo as column
f_inicio as column
f_fin as column
presupuesto as column
descripcion as column
end
INTERFACE
POSITION 0 0 525 292
LABEL "Mantenimiento de Proyectos"
SYSMENU
BEGIN
CONTROL AS BOX
POSITION 0 0 517 240
FOREGROUND RGB 0 0 0
ATTACH ALL
NOLABEL
NOBORDER
BEGIN
CONTROL AS BUTTON
POSITION 428 13 76 22
ATTACH RIGHT
LABEL "&Buscar"
ICON 6 FILE "Small Buttons"
COMMAND QueryByForm
CONTROL AS BUTTON
POSITION 428 65 76 22
```

```
ATTACH RIGHT
LABEL "&Añadir"
ICON 157 FILE "Small Buttons"
COMMAND Add
CONTROL AS BUTTON
POSITION 428 91 76 22
ATTACH RIGHT
LABEL "&Borrar"
ICON 159 FILE "Small Buttons"
COMMAND Delete
CONTROL AS BUTTON
POSITION 428 117 76 22
ATTACH RIGHT
LABEL "&Modificar"
ICON 158 FILE "Small Buttons"
COMMAND Update
CONTROL AS BUTTON
POSITION 428 39 76 22
ATTACH RIGHT
LABEL "&Nuevo"
ICON 84 FILE "Small Buttons".COMMAND New
CONTROL AS BUTTON
POSITION 428 156 76 22
ATTACH RIGHT
LABEL "&Salir"
ICON 4 FILE "Small Buttons"
COMMAND Close
CONTROL IDMASTER AS BOX
POSITION 4 4 393 226
ATTACH ALL
NOLABEL
BORDER DOUBLE DOWN
TABLE proyectos
BEGIN
CONTROL AS TEXT
POSITION 7 13 67 22
NOBORDER
DATATYPE CHAR
TAGS "Cod. Proyecto"
CONTROL AS EDIT
POSITION 101 13 70 22
LABEL "Cod. Proyecto"
BORDER DOUBLE DOWN
VARIABLE proyectos.cod_pyto
LIKEVAR
CONTROL AS TEXT
POSITION 7 39 28 22
NOBORDER
DATATYPE CHAR
TAGS "Título"
CONTROL AS EDIT
POSITION 101 39 196 22
LABEL "Título"
BORDER DOUBLE DOWN
VARIABLE proyectos.titulo
LIKEVAR
CONTROL AS TEXT
POSITION 7 65 58 22
NOBORDER
DATATYPE CHAR
TAGS "Fecha Inicio"
```

```
CONTROL AS EDIT
POSITION 101 65 82 22
LABEL "Fecha Inicio"
BORDER DOUBLE DOWN
VARIABLE proyectos.f_inicio
LIKEVAR
CONTROL AS TEXT
POSITION 7 91 88 22
NOBORDER
DATATYPE CHAR
TAGS "Fecha Finalización"
CONTROL AS EDIT
POSITION 101 91 82 22
LABEL "Fecha Finalización"
BORDER DOUBLE DOWN
VARIABLE proyectos.f_fin.LIKEVAR
CONTROL AS TEXT
POSITION 7 117 59 22
NOBORDER
DATATYPE CHAR
TAGS "Presupuesto"
CONTROL AS EDIT
POSITION 101 117 94 22
LABEL "Presupuesto"
BORDER DOUBLE DOWN
VARIABLE proyectos.presupuesto
LIKEVAR
CONTROL AS EDIT
POSITION 0 0 0 0
FOREGROUND RGB 0 0 0
BACKGROUND RGB 0 0 0
DATATYPE CHAR
CONTROL AS EDIT
POSITION 98 143 273 65
LABEL "Descripción"
BORDER DOUBLE DOWN
VARIABLE proyectos.descripcion
SPIN BUTTON
DATATYPE CHAR(32000)
END
CONTROL AS SPIN
POSITION 397 4 20 226
ATTACH TOP RIGHT BOTTOM
BORDER DOUBLE DOWN
VERTICAL
QUADRUPLE
COMMAND
CONTROL AS TEXT
POSITION 14 156 59 22
NOBORDER
DATATYPE CHAR
TAGS "Descripción"
END
CONTROL AS BAR
POSITION 0 240 517 27
FOREGROUND RGB 0 0 0
ATTACH LEFT RIGHT BOTTOM
BORDER ETCHED EXTRA 3
BARTYPE STATUS
BEGIN
CONTROL AS PANEL
```

```
POSITION 3 0 467 19
FOREGROUND RGB 0 0 0
ATTACH ALL
NOBORDER
PANELTYPE COMMENT
CONTROL AS PANEL
POSITION 473 0 35 19
FOREGROUND RGB 0 0 0
ATTACH TOP RIGHT BOTTOM
BORDER DOWN
PANELTYPE EDITMODE.END
END
end
END
CODE BEGIN
//{{CODEBEGIN
main
objects
begin
oCABM_proyectos as CABM_proyectos
end
begin
Sql.AttachConnection("ConexionGeneral");
Sql.Connect;
oCABM_proyectos.Run;
Sql.Disconnect;
Sql.DettachConnection;
end
//{{CODEEND
END
```

7. Módulo ABM_cargos.

```
REPOSITORY repositorio
CONSTANTS begin
end
CLASSES BEGIN
CABM_cargos is Form begin
objects begin
end
Table cargos is cargos
begin
cod_pyto as column
concepto as column
f_cargo as column
cuantia as column
end
INTERFACE
POSITION 0 0 532 231
LABEL "Mantenimiento de Cargos"
SYSMENU
BEGIN
CONTROL AS BOX
POSITION 0 0 524 179
FOREGROUND RGB 0 0 0
ATTACH ALL
NOLABEL
NOBORDER
```

```
BEGIN
CONTROL AS BUTTON
POSITION 435 13 76 22
ATTACH RIGHT
LABEL "&Buscar"
ICON 6 FILE "Small Buttons"
COMMAND QueryByForm
CONTROL AS BUTTON
POSITION 435 65 76 22
ATTACH RIGHT
LABEL "&Añadir"
ICON 157 FILE "Small Buttons"
COMMAND Add
CONTROL AS BUTTON
POSITION 435 91 76 22
ATTACH RIGHT
LABEL "&Borrar"
ICON 159 FILE "Small Buttons"
COMMAND Delete
CONTROL AS BUTTON
POSITION 435 117 76 22
ATTACH RIGHT
LABEL "&Modificar"
ICON 158 FILE "Small Buttons"
COMMAND Update
CONTROL AS BUTTON
POSITION 435 39 76 22
ATTACH RIGHT
LABEL "&Nuevo"
ICON 84 FILE "Small Buttons"
COMMAND New
CONTROL AS BUTTON.POSITION 435 156 76 22
ATTACH RIGHT
LABEL "&Salir"
ICON 4 FILE "Small Buttons"
COMMAND Close
CONTROL IDMASTER AS BOX
POSITION 4 4 400 165
ATTACH ALL
NOLABEL
BORDER DOUBLE DOWN
TABLE cargos
BEGIN
CONTROL AS TEXT
POSITION 7 39 67 22
NOBORDER
DATATYPE CHAR
TAGS "Cod. Proyecto"
CONTROL AS EDIT
POSITION 80 39 70 22
LABEL "Cod. Proyecto"
BORDER DOUBLE DOWN
VARIABLE cargos.cod_pyto
LIKEVAR
CONTROL AS TEXT
POSITION 7 65 46 22
NOBORDER
DATATYPE CHAR
TAGS "Concepto"
CONTROL AS EDIT
POSITION 80 65 316 22
```

```
LABEL "Concepto"
BORDER DOUBLE DOWN
VARIABLE cargos.concepto
LIKEVAR
CONTROL AS TEXT
POSITION 7 91 61 22
NOBORDER
DATATYPE CHAR
TAGS "Fecha Cargo"
CONTROL AS EDIT
POSITION 80 91 82 22
LABEL "Fecha Cargo"
BORDER DOUBLE DOWN
VARIABLE cargos.f_cargo
LIKEVAR
CONTROL AS TEXT
POSITION 7 117 38 22
NOBORDER
DATATYPE CHAR
TAGS "Cuantía"
CONTROL AS EDIT
POSITION 80 117 100 22
LABEL "Cuantía"
BORDER DOUBLE DOWN
VARIABLE cargos.cuantia
LIKEVAR
END.CONTROL AS SPIN
POSITION 404 4 20 165
ATTACH TOP RIGHT BOTTOM
BORDER DOUBLE DOWN
VERTICAL
QUADRUPLE
COMMAND
END
CONTROL AS BAR
POSITION 0 179 524 27
FOREGROUND RGB 0 0 0
ATTACH LEFT RIGHT BOTTOM
BORDER ETCHED EXTRA 3
BARTYPE STATUS
BEGIN
CONTROL AS PANEL
POSITION 3 0 474 19
FOREGROUND RGB 0 0 0
ATTACH ALL
NOBORDER
PANELTYPE COMMENT
CONTROL AS PANEL
POSITION 480 0 35 19
FOREGROUND RGB 0 0 0
ATTACH TOP RIGHT BOTTOM
BORDER DOWN
PANELTYPE EDITMODE
END
END
end
END
CODE BEGIN
//{{CODEBEGIN
main
objects
```

```
begin
oCABM_cargos as CABM_cargos
end
begin
Sql.AttachConnection("ConexionGeneral");
Sql.Connect;
oCABM_cargos.Run;
Sql.Disconnect;
Sql.DettachConnection;
end
//{{CODEEND
END
```

8. Módulo md_dpto_empleados.

```
REPOSITORY repositorio
CONSTANTS begin
end
CLASSES BEGIN
Cmd_dpto_empleados is Form begin
objects begin
end
Table departamentos is departamentos
begin
cod_dpto as column required
nombre as column
end
Table empleados is empleados
depend on departamentos join cr_dptos
begin
dni as column required
nombre as column
apellidos as column
domicilio as column
localidad as column
cod_postal as column
telefono as column
end
INTERFACE
POSITION 0 0 457 347
LABEL "Departamento-Empleados"
SYSMENU
BEGIN
CONTROL AS BOX
POSITION 0 0 449 295
FOREGROUND RGB 0 0 0
ATTACH ALL
NOLABEL
NOBORDER
BEGIN
CONTROL IDMASTER AS BOX
POSITION 7 5 343 74
ATTACH ALL
NOLABEL
BORDER DOUBLE FRAME
TABLE departamentos
BEGIN
CONTROL AS SPIN
POSITION 314 0 17 62
```

```
ATTACH TOP RIGHT BOTTOM
NOBORDER
VERTICAL
QUADRUPLE
COMMAND
CONTROL AS TEXT
POSITION 7 13 92 22
NOBORDER
DATATYPE CHAR
TAGS "Cod. Departamento"
CONTROL AS EDIT
POSITION 105 13 52 22
LABEL "Cod. Departamento".BORDER DOUBLE DOWN
VARIABLE departamentos.cod_dpto
LIKEVAR
CONTROL AS TEXT
POSITION 7 39 37 22
NOBORDER
DATATYPE CHAR
TAGS "Nombre"
CONTROL AS EDIT
POSITION 105 39 196 22
LABEL "Nombre"
BORDER DOUBLE DOWN
VARIABLE departamentos.nombre
LIKEVAR
END
CONTROL AS BUTTON
POSITION 368 5 76 22
ATTACH RIGHT
LABEL "&Buscar"
ICON 6 FILE "Small Buttons"
COMMAND QueryByForm
CONTROL AS BUTTON
POSITION 368 31 76 22
ATTACH RIGHT
LABEL "&Nuevo"
ICON 84 FILE "Small Buttons"
COMMAND New
CONTROL AS BUTTON
POSITION 368 57 76 22
ATTACH RIGHT
LABEL "&Añadir"
ICON 157 FILE "Small Buttons"
COMMAND Add
CONTROL AS BUTTON
POSITION 368 83 76 22
ATTACH RIGHT
LABEL "&Borrar"
ICON 159 FILE "Small Buttons"
COMMAND Delete
CONTROL AS BUTTON
POSITION 368 109 76 22
ATTACH RIGHT
LABEL "&Modificar"
ICON 158 FILE "Small Buttons"
COMMAND Update
CONTROL AS BUTTON
POSITION 364 264 76 22
ATTACH RIGHT BOTTOM
LABEL "&Salir"
```

```
ICON 4 FILE "Small Buttons"
COMMAND Close
CONTROL AS BOX
POSITION 7 79 343 210
ATTACH LEFT RIGHT BOTTOM
NOLABEL
BORDER DOUBLE FRAME
BEGIN.CONTROL IDLINES AS GRID
POSITION 0 0 331 175
ATTACH ALL
BORDER DOWN
SCROLL VERTICAL
TABLE empleados
PARENTGRID
BEGIN
CONTROL AS EDIT
POSITION 0 0 66 22
LABEL "DNI y letra"
NOBORDER
VARIABLE empleados.dni
LIKEVAR
CONTROL AS EDIT
POSITION 67 0 162 22
LABEL "Nombre"
NOBORDER
VARIABLE empleados.nombre
LIKEVAR
CONTROL AS EDIT
POSITION 230 0 192 22
LABEL "Apellidos"
NOBORDER
VARIABLE empleados.apellidos
LIKEVAR
CONTROL AS EDIT
POSITION 423 0 192 22
LABEL "Domicilio"
NOBORDER
VARIABLE empleados.domicilio
LIKEVAR
CONTROL AS EDIT
POSITION 616 0 192 22
LABEL "Localidad"
NOBORDER
VARIABLE empleados.localidad
LIKEVAR
CONTROL AS EDIT
POSITION 809 0 90 22
LABEL "Cod. Postal"
NOBORDER
VARIABLE empleados.cod_postal
LIKEVAR
CONTROL AS EDIT
POSITION 900 0 90 22
LABEL "Teléfono"
NOBORDER
VARIABLE empleados.telefono
LIKEVAR
END
CONTROL AS BUTTON
POSITION 166 176 165 22
ATTACH RIGHT BOTTOM
```

```
LABEL "Modificar Linea"
ICON 158 FILE "Small Buttons" CENTER
COMMAND UpdateLine.CONTROL AS BUTTON
POSITION 165 176 1 22
ATTACH LEFT RIGHT BOTTOM
LABEL "Borrar Linea"
ICON 159 FILE "Small Buttons" CENTER
COMMAND DeleteLine
CONTROL AS BUTTON
POSITION 0 176 165 22
ATTACH LEFT BOTTOM
LABEL "Añadir Linea"
ICON 157 FILE "Small Buttons" CENTER
COMMAND AddLine
END
END
CONTROL AS BAR
POSITION 0 295 449 27
FOREGROUND RGB 0 0 0
ATTACH LEFT RIGHT BOTTOM
BORDER ETCHED EXTRA 3
BARTYPE STATUS
BEGIN
CONTROL AS PANEL
POSITION 3 0 399 19
FOREGROUND RGB 0 0 0
ATTACH ALL
NOBORDER
PANELTYPE COMMENT
CONTROL AS PANEL
POSITION 405 0 35 19
FOREGROUND RGB 0 0 0
ATTACH TOP RIGHT BOTTOM
BORDER DOWN
PANELTYPE EDITMODE
END
END
end
END
CODE CLASS Cmd_dpto_empleados BEGIN
//{{CODEBEGIN
On command AddLine
begin
MasterTable.ChildTable.Add;
end
On command DeleteLine
begin
MasterTable.ChildTable.Delete;
end
On command UpdateLine
begin
MasterTable.ChildTable.Update;
end
//{{CODEEND
END
CODE BEGIN
//{{CODEBEGIN.main
objects
begin
oCmd_dpto_empleados as Cmd_dpto_empleados
end
```

```
begin
Sql.AttachConnection("ConexionGeneral");
Sql.Connect;
oCmd_dpto_empleados.Run;
Sql.Disconnect;
Sql.DettachConnection;
end
//{{CODEEND
END
```

9. Módulo md_empleado_pytos.

```
REPOSITORY repositorio
CONSTANTS begin
end
CLASSES BEGIN
Cmd_empleado_pytos is Form begin
objects begin
end
Table proyectos is proyectos
begin
cod_pyto as column required
titulo as column
end
Table empleados is empleados
depend on proyectos join cr_emp_pyto
begin
dni as column required
nombre as column
apellidos as column
cod_dpto as column
end
INTERFACE
POSITION 0 0 457 347
LABEL "Proyecto-Empleados"
SYSMENU
BEGIN
CONTROL AS BOX
POSITION 0 0 449 295
FOREGROUND RGB 0 0 0
ATTACH ALL
NOLABEL
NOBORDER
BEGIN
CONTROL IDMASTER AS BOX
POSITION 7 5 343 74
ATTACH ALL
NOLABEL
BORDER DOUBLE FRAME
TABLE proyectos
BEGIN
CONTROL AS SPIN
POSITION 314 0 17 62
ATTACH TOP RIGHT BOTTOM
NOBORDER
VERTICAL
QUADRUPLE
COMMAND
```

```
CONTROL AS TEXT
POSITION 7 13 67 22
NOBORDER
DATATYPE CHAR
TAGS "Cod. Proyecto"
CONTROL AS EDIT
POSITION 80 13 70 22
LABEL "Cod. Proyecto"
BORDER DOUBLE DOWN
VARIABLE proyectos.cod_pyto
LIKEVAR.CONTROL AS TEXT
POSITION 7 39 28 22
NOBORDER
DATATYPE CHAR
TAGS "Título"
CONTROL AS EDIT
POSITION 80 39 196 22
LABEL "Título"
BORDER DOUBLE DOWN
VARIABLE proyectos.titulo
LIKEVAR
END
CONTROL AS BUTTON
POSITION 368 5 76 22
ATTACH RIGHT
LABEL "&Buscar"
ICON 6 FILE "Small Buttons"
COMMAND QueryByForm
CONTROL AS BUTTON
POSITION 368 31 76 22
ATTACH RIGHT
LABEL "&Nuevo"
ICON 84 FILE "Small Buttons"
COMMAND New
CONTROL AS BUTTON
POSITION 368 57 76 22
ATTACH RIGHT
LABEL "&Añadir"
ICON 157 FILE "Small Buttons"
COMMAND Add
CONTROL AS BUTTON
POSITION 368 83 76 22
ATTACH RIGHT
LABEL "&Borrar"
ICON 159 FILE "Small Buttons"
COMMAND Delete
CONTROL AS BUTTON
POSITION 368 109 76 22
ATTACH RIGHT
LABEL "&Modificar"
ICON 158 FILE "Small Buttons"
COMMAND Update
CONTROL AS BUTTON
POSITION 364 264 76 22
ATTACH RIGHT BOTTOM
LABEL "&Salir"
ICON 4 FILE "Small Buttons"
COMMAND Close
CONTROL AS BOX
POSITION 7 79 343 210
ATTACH LEFT RIGHT BOTTOM
```

```
NOLABEL
BORDER DOUBLE FRAME
BEGIN
CONTROL IDLINES AS GRID
POSITION 0 0 331 175
ATTACH ALL.BORDER DOWN
SCROLL VERTICAL
TABLE empleados
PARENTGRID
BEGIN
CONTROL AS EDIT
POSITION 0 0 66 22
LABEL "DNI y letra"
NOBORDER
VARIABLE empleados.dni
LIKEVAR
CONTROL AS EDIT
POSITION 67 0 162 22
LABEL "Nombre"
NOBORDER
VARIABLE empleados.nombre
LIKEVAR
CONTROL AS EDIT
POSITION 230 0 192 22
LABEL "Apellidos"
NOBORDER
VARIABLE empleados.apellidos
LIKEVAR
CONTROL AS EDIT
POSITION 423 0 48 22
LABEL "Cod. Departamento"
NOBORDER
VARIABLE empleados.cod_dpto
LIKEVAR
END
CONTROL AS BUTTON
POSITION 166 176 165 22
ATTACH RIGHT BOTTOM
LABEL "Modificar Linea"
ICON 158 FILE "Small Buttons" CENTER
COMMAND UpdateLine
CONTROL AS BUTTON
POSITION 165 176 1 22
ATTACH LEFT RIGHT BOTTOM
LABEL "Borrar Linea"
ICON 159 FILE "Small Buttons" CENTER
COMMAND DeleteLine
CONTROL AS BUTTON
POSITION 0 176 165 22
ATTACH LEFT BOTTOM
LABEL "Añadir Linea"
ICON 157 FILE "Small Buttons" CENTER
COMMAND AddLine
END
END
CONTROL AS BAR
POSITION 0 295 449 27
FOREGROUND RGB 0 0 0
ATTACH LEFT RIGHT BOTTOM
BORDER ETCHED EXTRA 3
BARTYPE STATUS
```

```
BEGIN.CONTROL AS PANEL
POSITION 3 0 399 19
FOREGROUND RGB 0 0 0
ATTACH ALL
NOBORDER
PANELTYPE COMMENT
CONTROL AS PANEL
POSITION 405 0 35 19
FOREGROUND RGB 0 0 0
ATTACH TOP RIGHT BOTTOM
BORDER DOWN
PANELTYPE EDITMODE
END
END
end
END
CODE CLASS Cmd_empleado_pytos BEGIN
//{{CODEBEGIN
On command AddLine
begin
MasterTable.ChildTable.Add;
end
On command DeleteLine
begin
MasterTable.ChildTable.Delete;
end
On command UpdateLine
begin
MasterTable.ChildTable.Update;
end
//{{CODEEND
END
CODE BEGIN
//{{CODEBEGIN
main
objects
begin
oCmd_empleado_pytos as Cmd_empleado_pytos
end
begin
Sql.AttachConnection("ConexionGeneral");
Sql.Connect;
oCmd_empleado_pytos.Run;
Sql.Disconnect;
Sql.DettachConnection;
end
//{{CODEEND
END
```

10. Módulo md_pytos_cargos.

```
REPOSITORY repositorio
CONSTANTS begin
end
CLASSES BEGIN
Cmd_pytos_cargos is Form begin
objects begin
end
```

```
Table proyectos is proyectos
begin
cod_pyto as column required
titulo as column
end
Table cargos is cargos
depend on proyectos join cr_cargo_pyto
begin
concepto as column
f_cargo as column
cuantia as column
end
INTERFACE
POSITION 0 0 557 378
LABEL "Proyecto-Cargos"
SYSMENU
BEGIN
CONTROL AS BOX
POSITION 0 0 549 326
FOREGROUND RGB 0 0 0
ATTACH ALL
NOLABEL
NOBORDER
BEGIN
CONTROL IDMASTER AS BOX
POSITION 7 5 443 105
ATTACH ALL
NOLABEL
BORDER DOUBLE FRAME
TABLE proyectos
BEGIN
CONTROL AS SPIN
POSITION 414 0 17 93
ATTACH TOP RIGHT BOTTOM
NOBORDER
VERTICAL
QUADRUPLE
COMMAND
CONTROL AS TEXT
POSITION 7 13 67 22
NOBORDER
DATATYPE CHAR
TAGS "Cod. Proyecto"
CONTROL AS EDIT
POSITION 80 13 70 22
LABEL "Cod. Proyecto"
BORDER DOUBLE DOWN
VARIABLE proyectos.cod_pyto
LIKEVAR
CONTROL AS TEXT.POSITION 7 39 28 22
NOBORDER
DATATYPE CHAR
TAGS "Título"
CONTROL AS EDIT
POSITION 80 39 196 22
LABEL "Título"
BORDER DOUBLE DOWN
VARIABLE proyectos.titulo
LIKEVAR
END
CONTROL AS BUTTON
```

```
POSITION 468 5 76 22
ATTACH RIGHT
LABEL "&Buscar"
ICON 6 FILE "Small Buttons"
COMMAND QueryByForm
CONTROL AS BUTTON
POSITION 468 31 76 22
ATTACH RIGHT
LABEL "&Nuevo"
ICON 84 FILE "Small Buttons"
COMMAND New
CONTROL AS BUTTON
POSITION 468 57 76 22
ATTACH RIGHT
LABEL "&Añadir"
ICON 157 FILE "Small Buttons"
COMMAND Add
CONTROL AS BUTTON
POSITION 468 83 76 22
ATTACH RIGHT
LABEL "&Borrar"
ICON 159 FILE "Small Buttons"
COMMAND Delete
CONTROL AS BUTTON
POSITION 468 109 76 22
ATTACH RIGHT
LABEL "&Modificar"
ICON 158 FILE "Small Buttons"
COMMAND Update
CONTROL AS BUTTON
POSITION 464 295 76 22
ATTACH RIGHT BOTTOM
LABEL "&Salir"
ICON 4 FILE "Small Buttons"
COMMAND Close
CONTROL AS BOX
POSITION 7 110 443 210
ATTACH LEFT RIGHT BOTTOM
NOLABEL
BORDER DOUBLE FRAME
BEGIN
CONTROL IDLINES AS GRID
POSITION 0 0 431 175
ATTACH ALL
BORDER DOWN.SCROLL VERTICAL
TABLE cargos
PARENTGRID
BEGIN
CONTROL AS EDIT
POSITION 0 0 175 22
LABEL "Concepto"
NOBORDER
VARIABLE cargos.concepto
LIKEVAR
CONTROL AS EDIT
POSITION 176 0 78 22
LABEL "Fecha Cargo"
NOBORDER
VARIABLE cargos.f_cargo
LIKEVAR
CONTROL AS EDIT
```

```
POSITION 255 0 109 22
LABEL "Cuantía"
NOBORDER
VARIABLE cargos.cuantia
LIKEVAR
END
CONTROL AS BUTTON
POSITION 266 176 165 22
ATTACH RIGHT BOTTOM
LABEL "Modificar Linea"
ICON 158 FILE "Small Buttons" CENTER
COMMAND UpdateLine
CONTROL AS BUTTON
POSITION 165 176 101 22
ATTACH LEFT RIGHT BOTTOM
LABEL "Borrar Linea"
ICON 159 FILE "Small Buttons" CENTER
COMMAND DeleteLine
CONTROL AS BUTTON
POSITION 0 176 165 22
ATTACH LEFT BOTTOM
LABEL "Añadir Linea"
ICON 157 FILE "Small Buttons" CENTER
COMMAND AddLine
END
END
CONTROL AS BAR
POSITION 0 326 549 27
FOREGROUND RGB 0 0 0
ATTACH LEFT RIGHT BOTTOM
BORDER ETCHED EXTRA 3
BARTYPE STATUS
BEGIN
CONTROL AS PANEL
POSITION 3 0 499 19
FOREGROUND RGB 0 0 0
ATTACH ALL
NOBORDER
PANELTYPE COMMENT
CONTROL AS PANEL.POSITION 505 0 35 19
FOREGROUND RGB 0 0 0
ATTACH TOP RIGHT BOTTOM
BORDER DOWN
PANELTYPE EDITMODE
END
END
end
END
CODE CLASS Cmd_pytos_cargos BEGIN
//{{CODEBEGIN
On command AddLine
begin
MasterTable.ChildTable.Add;
end
On command DeleteLine
begin
MasterTable.ChildTable.Delete;
end
On command UpdateLine
begin
MasterTable.ChildTable.Update;
```

```
end
//{{CODEEND
END
CODE BEGIN
//{{CODEBEGIN
main
objects
begin
oCmd_pytos_cargos as Cmd_pytos_cargos
end
begin
Sql.AttachConnection("ConexionGeneral");
Sql.Connect;
oCmd_pytos_cargos.Run;
Sql.Disconnect;
Sql.DettachConnection;
end
//{{CODEEND
END
```

11. Módulo md_empresas_pyto.

```
REPOSITORY repositorio
CONSTANTS begin
end
CLASSES BEGIN
Cmd_empresas_ptyo is Form begin
objects begin
end
Table proyectos is proyectos
begin
cod_pyto as column required
titulo as column
end
Table empresas is empresas
depend on proyectos join cr_empresas_pytos
begin
cod_empresa as column required
nombre as column
end
INTERFACE
POSITION 0 0 457 347
LABEL "Proyecto-Empresas"
SYSMENU
BEGIN
CONTROL AS BOX
POSITION 0 0 449 295
FOREGROUND RGB 0 0 0
ATTACH ALL
NOLABEL
NOBORDER
BEGIN
CONTROL IDMASTER AS BOX
POSITION 7 5 343 74
ATTACH ALL
NOLABEL
BORDER DOUBLE FRAME
TABLE proyectos
```

```
BEGIN
CONTROL AS SPIN
POSITION 314 0 17 62
ATTACH TOP RIGHT BOTTOM
NOBORDER
VERTICAL
QUADRUPLE
COMMAND
CONTROL AS TEXT
POSITION 7 13 67 22
NOBORDER
DATATYPE CHAR
TAGS "Cod. Proyecto"
CONTROL AS EDIT
POSITION 80 13 70 22
LABEL "Cod. Proyecto"
BORDER DOUBLE DOWN
VARIABLE proyectos.cod_pyto
LIKEVAR
CONTROL AS TEXT
POSITION 7 39 28 22.NOBORDER
DATATYPE CHAR
TAGS "Título"
CONTROL AS EDIT
POSITION 80 39 196 22
LABEL "Título"
BORDER DOUBLE DOWN
VARIABLE proyectos.titulo
LIKEVAR
END
CONTROL AS BUTTON
POSITION 368 5 76 22
ATTACH RIGHT
LABEL "&Buscar"
ICON 6 FILE "Small Buttons"
COMMAND QueryByForm
CONTROL AS BUTTON
POSITION 368 31 76 22
ATTACH RIGHT
LABEL "&Nuevo"
ICON 84 FILE "Small Buttons"
COMMAND New
CONTROL AS BUTTON
POSITION 368 57 76 22
ATTACH RIGHT
LABEL "&Añadir"
ICON 157 FILE "Small Buttons"
COMMAND Add
CONTROL AS BUTTON
POSITION 368 83 76 22
ATTACH RIGHT
LABEL "&Borrar"
ICON 159 FILE "Small Buttons"
COMMAND Delete
CONTROL AS BUTTON
POSITION 368 109 76 22
ATTACH RIGHT
LABEL "&Modificar"
ICON 158 FILE "Small Buttons"
COMMAND Update
CONTROL AS BUTTON
```

```
POSITION 364 264 76 22
ATTACH RIGHT BOTTOM
LABEL "&Salir"
ICON 4 FILE "Small Buttons"
COMMAND Close
CONTROL AS BOX
POSITION 7 79 343 210
ATTACH LEFT RIGHT BOTTOM
NOLABEL
BORDER DOUBLE FRAME
BEGIN
CONTROL IDLINES AS GRID
POSITION 0 0 331 175
ATTACH ALL
BORDER DOWN
SCROLL VERTICAL.TABLE empresas
PARENTGRID
BEGIN
CONTROL AS EDIT
POSITION 0 0 98 22
LABEL "Cod. Empresa"
NOBORDER
VARIABLE empresas.cod_empresa
LIKEVAR
CONTROL AS EDIT
POSITION 99 0 192 22
LABEL "Nombre"
NOBORDER
VARIABLE empresas.nombre
LIKEVAR
END
CONTROL AS BUTTON
POSITION 166 176 165 22
ATTACH RIGHT BOTTOM
LABEL "Modificar Linea"
ICON 158 FILE "Small Buttons" CENTER
COMMAND UpdateLine
CONTROL AS BUTTON
POSITION 165 176 1 22
ATTACH LEFT RIGHT BOTTOM
LABEL "Borrar Linea"
ICON 159 FILE "Small Buttons" CENTER
COMMAND DeleteLine
CONTROL AS BUTTON
POSITION 0 176 165 22
ATTACH LEFT BOTTOM
LABEL "Añadir Linea"
ICON 157 FILE "Small Buttons" CENTER
COMMAND AddLine
END
END
CONTROL AS BAR
POSITION 0 295 449 27
FOREGROUND RGB 0 0 0
ATTACH LEFT RIGHT BOTTOM
BORDER ETCHED EXTRA 3
BARTYPE STATUS
BEGIN
CONTROL AS PANEL
POSITION 3 0 399 19
FOREGROUND RGB 0 0 0
```

```
ATTACH ALL
NOBORDER
PANELTYPE COMMENT
CONTROL AS PANEL
POSITION 405 0 35 19
FOREGROUND RGB 0 0 0
ATTACH TOP RIGHT BOTTOM
BORDER DOWN
PANELTYPE EDITMODE
END
END.end
END
CODE CLASS Cmd_empresas_ptyo BEGIN
//{{CODEBEGIN
On command AddLine
begin
MasterTable.ChildTable.Add;
end
On command DeleteLine
begin
MasterTable.ChildTable.Delete;
end
On command UpdateLine
begin
MasterTable.ChildTable.Update;
end
//{{CODEEND
END
CODE BEGIN
//{{CODEBEGIN
main
objects
begin
oCmd_empresas_ptyo as Cmd_empresas_ptyo
end
begin
Sql.AttachConnection("ConexionGeneral");
Sql.Connect;
oCmd_empresas_ptyo.Run;
Sql.Disconnect;
Sql.DettachConnection;
end
//{{CODEEND
END
```

12. Módulo md_dpto_pytos.

```
REPOSITORY repositorio
CONSTANTS begin
end
CLASSES BEGIN
Cmd_dpto_pytos is Form begin
objects begin
end
Table proyectos is proyectos
begin
cod_pyto as column required
titulo as column
```

```
end
Table departamentos is departamentos
depend on proyectos join cr_dep_pyto
begin
cod_dpto as column required
nombre as column
end
INTERFACE
POSITION 0 0 457 347
LABEL "Departamento-Proyectos"
SYSMENU
BEGIN
CONTROL AS BOX
POSITION 0 0 449 295
FOREGROUND RGB 0 0 0
ATTACH ALL
NOLABEL
NOBORDER
BEGIN
CONTROL IDMASTER AS BOX
POSITION 7 5 343 74
ATTACH ALL
NOLABEL
BORDER DOUBLE FRAME
TABLE proyectos
BEGIN
CONTROL AS SPIN
POSITION 314 0 17 62
ATTACH TOP RIGHT BOTTOM
NOBORDER
VERTICAL
QUADRUPLE
COMMAND
CONTROL AS TEXT
POSITION 7 13 67 22
NOBORDER
DATATYPE CHAR
TAGS "Cod. Proyecto"
CONTROL AS EDIT
POSITION 80 13 70 22
LABEL "Cod. Proyecto"
BORDER DOUBLE DOWN
VARIABLE proyectos.cod_pyto
LIKEVAR
CONTROL AS TEXT
POSITION 7 39 28 22.NOBORDER
DATATYPE CHAR
TAGS "Título"
CONTROL AS EDIT
POSITION 80 39 196 22
LABEL "Título"
BORDER DOUBLE DOWN
VARIABLE proyectos.titulo
LIKEVAR
END
CONTROL AS BUTTON
POSITION 368 5 76 22
ATTACH RIGHT
LABEL "&Buscar"
ICON 6 FILE "Small Buttons"
COMMAND QueryByForm
```

```
CONTROL AS BUTTON
POSITION 368 31 76 22
ATTACH RIGHT
LABEL "&Nuevo"
ICON 84 FILE "Small Buttons"
COMMAND New
CONTROL AS BUTTON
POSITION 368 57 76 22
ATTACH RIGHT
LABEL "&Añadir"
ICON 157 FILE "Small Buttons"
COMMAND Add
CONTROL AS BUTTON
POSITION 368 83 76 22
ATTACH RIGHT
LABEL "&Borrar"
ICON 159 FILE "Small Buttons"
COMMAND Delete
CONTROL AS BUTTON
POSITION 368 109 76 22
ATTACH RIGHT
LABEL "&Modificar"
ICON 158 FILE "Small Buttons"
COMMAND Update
CONTROL AS BUTTON
POSITION 364 264 76 22
ATTACH RIGHT BOTTOM
LABEL "&Salir"
ICON 4 FILE "Small Buttons"
COMMAND Close
CONTROL AS BOX
POSITION 7 79 343 210
ATTACH LEFT RIGHT BOTTOM
NOLABEL
BORDER DOUBLE FRAME
BEGIN
CONTROL IDLINES AS GRID
POSITION 0 0 331 175
ATTACH ALL
BORDER DOWN
SCROLL VERTICAL.TABLE departamentos
PARENTGRID
BEGIN
CONTROL AS EDIT
POSITION 0 0 48 22
LABEL "Cod. Departamento"
NOBORDER
VARIABLE departamentos.cod_dpto
LIKEVAR
CONTROL AS EDIT
POSITION 49 0 192 22
LABEL "Nombre"
NOBORDER
VARIABLE departamentos.nombre
LIKEVAR
END
CONTROL AS BUTTON
POSITION 166 176 165 22
ATTACH RIGHT BOTTOM
LABEL "Modificar Linea"
ICON 158 FILE "Small Buttons" CENTER
```

```
COMMAND UpdateLine
CONTROL AS BUTTON
POSITION 165 176 1 22
ATTACH LEFT RIGHT BOTTOM
LABEL "Borrar Linea"
ICON 159 FILE "Small Buttons" CENTER
COMMAND DeleteLine
CONTROL AS BUTTON
POSITION 0 176 165 22
ATTACH LEFT BOTTOM
LABEL "Añadir Linea"
ICON 157 FILE "Small Buttons" CENTER
COMMAND AddLine
END
END
CONTROL AS BAR
POSITION 0 295 449 27
FOREGROUND RGB 0 0 0
ATTACH LEFT RIGHT BOTTOM
BORDER ETCHED EXTRA 3
BARTYPE STATUS
BEGIN
CONTROL AS PANEL
POSITION 3 0 399 19
FOREGROUND RGB 0 0 0
ATTACH ALL
NOBORDER
PANELTYPE COMMENT
CONTROL AS PANEL
POSITION 405 0 35 19
FOREGROUND RGB 0 0 0
ATTACH TOP RIGHT BOTTOM
BORDER DOWN
PANELTYPE EDITMODE
END
END.end
END
CODE CLASS Cmd_dpto_pytos BEGIN
//{{CODEBEGIN
On command AddLine
begin
MasterTable.ChildTable.Add;
end
On command DeleteLine
begin
MasterTable.ChildTable.Delete;
end
On command UpdateLine
begin
MasterTable.ChildTable.Update;
end
//{{CODEEND
END
CODE BEGIN
//{{CODEBEGIN
main
objects
begin
oCmd_dpto_pytos as Cmd_dpto_pytos
end
begin
```

```
Sql.AttachConnection("ConexionGeneral");
Sql.Connect;
oCmd_dpto_pytos.Run;
Sql.Disconnect;
Sql.DettachConnection;
end
//{{CODEEND
END
```

13. Módulo mCargos.

```
REPOSITORY repositorio
CONSTANTS begin
end
CLASSES BEGIN
cCargos is Form begin
objects begin
end
Table proyectos is proyectos
begin
cod_pyto as column required
titulo as column
f_inicio as column
f_fin as column
presupuesto as column
descripcion as column
end
Table cargos is cargos
depend on proyectos join cr_cargo_pyto
begin
concepto as column
f_cargo as column
cuantia as column
cr_cargo__cod_pyto as proyectos.cod_pyto
join cr_cargo_pyto lookup "cod_pyto"
end
INTERFACE
POSITION 0 0 488 271
LABEL "Cargos de un proyecto"
SYSMENU
BEGIN
CONTROL AS BOX
POSITION 0 0 480 246
FOREGROUND RGB 0 0 0
ATTACH ALL
NOLABEL
NOBORDER
BEGIN
CONTROL AS BOX
POSITION 9 0 470 133
NOLABEL
NOBORDER
TABLE proyectos
BEGIN
CONTROL AS TEXT
POSITION 18 48 126 22
NOBORDER
DATATYPE CHAR
```

```
TAGS "Título del proyecto"
CONTROL AS TEXT
POSITION 18 80 123 22
NOBORDER
DATATYPE CHAR
TAGS "Presupuesto Total"
CONTROL edit2 AS EDIT
POSITION 180 80 105 22
BORDER DOUBLE DOWN
VARIABLE proyectos.presupuesto.ALIGNMENT RIGHT
DATATYPE INTEGER
CONTROL edit1 AS EDIT
POSITION 180 48 182 22
BORDER DOUBLE DOWN
VARIABLE proyectos.titulo
DATATYPE CHAR
CONTROL navegador AS SPIN
POSITION 441 0 21 128
BORDER DOWN
VERTICAL
QUADRUPLE
CONTROL AS TEXT
POSITION 18 16 96 22
NOBORDER
DATATYPE CHAR
TAGS "Cod. Proyecto"
CONTROL edit0 AS EDIT
POSITION 180 16 70 22
LABEL "Cod. Proyecto"
BORDER DOUBLE DOWN
VARIABLE proyectos.cod_pyto
LIKEVAR
END
CONTROL label3 AS TEXT
POSITION 27 176 179 22
NOBORDER
DATATYPE CHAR
TAGS "Total de cargos al proyecto"
CONTROL edit4 AS EDIT
POSITION 225 176 105 22
FOREGROUND COLOR blue
FONT "MS Sans Serif" 8
BORDER DOUBLE DOWN
DATATYPE CHAR
CONTROL AS BUTTON
POSITION 396 208 77 26
LABEL "Salir"
ICON 4 FILE "_MBSTDBMP_" CENTER
COMMAND Close
CONTROL label4 AS TEXT
POSITION 27 144 161 22
NOBORDER
DATATYPE CHAR
TAGS "Número Total de cargos"
CONTROL edit3 AS EDIT
POSITION 225 144 105 22
FOREGROUND RGB 0 0 0
FONT "MS Sans Serif" 8
BORDER DOUBLE DOWN
DATATYPE CHAR
CONTROL label5 AS TEXT
```

```
POSITION 27 207 170 22
NOBORDER
DATATYPE CHAR
TAGS "Pendiente de Asignación"
CONTROL edit5 AS EDIT.POSITION 225 207 105 22
FOREGROUND COLOR red
FONT "MS Sans Serif" 8
BORDER DOUBLE DOWN
DATATYPE CHAR
END
END
end
END
OBJECTS BEGIN
oCargos AS cCargos
END
CODE CLASS cCargos BEGIN
//{{CODEBEGIN
On Open
begin
edit0.Disabled=true;
edit1.Disabled=true;
edit2.Disabled=true;
Query;
bCalcular;
end
private function bCalcular
objects
begin
nombre codigo as char
scursor as SqlCursor
presu cuantia total dif as money(15,2)
contador as smallint
end
begin
nombre=edit1.Text;
codigo=edit0.Text;
presu=edit2.Text;
total=0;
contador=0;
scursor.Prepare("Select cuantia from cargos where cargos.cod_pyto=?");
scursor.Open(codigo);
scursor.Fetch(cuantia);
while scursor.Found do begin
contador+=1;
total+=cuantia;
scursor.Fetch(cuantia);
end
scursor.Close;
scursor.Free;
edit3.Text=contador;
edit4.Text=total;
edit5.Text=presu-total;
end.On SpinDown navegador
begin
proyectos.Next;
bCalcular;
end
On SpinUp navegador
begin
proyectos.Prev;
```

```
bCalcular;
end
On SpinTop navegador
begin
proyectos.First;
bCalcular;
end
On SpinBottom navegador
begin
proyectos.Last;
bCalcular;
end
//{{CODEEND
END
CODE BEGIN
//{{CODEBEGIN
main
begin
Sql.AttachConnection("ConexionGeneral");
Sql.Connect;
oCargos.Run;
Sql.Disconnect;
Sql.DettachConnection;
end
//{{CODEEND
END
```

14. Módulo mEmpDpto.

```
REPOSITORY repositorio
CONSTANTS begin
end
CLASSES BEGIN
cEmp is Page begin
objects begin
apellidos as empleados.apellidos
dni as empleados.dni
nombre as empleados.nombre
end
INTERFACE
SIZE 595 842
MARGIN 33 18 39 27
BEGIN
CONTROL AS TEXT
POSITION 178 48 194 19
FOREGROUND COLOR blue
FONT "Comic Sans MS" 14 BOLD ITALIC
TAGS "Listado de Empleados"
CONTROL AS TEXT
POSITION 427 10 43 19
FOREGROUND COLOR blue
FONT "Comic Sans MS" 12
TAGS "Página:"
CONTROL pag AS VAR
POSITION 470 10 32 19
FOREGROUND COLOR blue
FONT "Arial" 12
ALIGNMENT RIGHT
```

```
DATATYPE SMALLINT MASK 2
CONTROL miGrupo AS GROUP
POSITION 5 77 525 720
BACKGROUND COLOR transparent
BORDER ALL
GRID VERTICAL HORIZONTAL
AS LINES
REPEAT 0
BEGIN
CONTROL AS VAR
POSITION 2 0 106 19
LABEL "DNI"
VARIABLE dni
LIKEVAR
CONTROL AS VAR
POSITION 113 0 202 19
LABEL "Apellidos"
VARIABLE apellidos
LIKEVAR
CONTROL AS VAR
POSITION 317 0 207 19
LABEL "Nombre"
VARIABLE nombre
LIKEVAR
END
CONTROL dpto AS VAR
POSITION 11 10 356 19
FOREGROUND COLOR blue.FONT "Comic Sans MS" 18 BOLD
DATATYPE CHAR(30)
END
end
cDpto is Form begin
objects begin
end
Table departamentos is departamentos
begin
cod_dpto as column required
nombre as column
cod_pyto as column
cr_dep_py_cod_pyto as proyectos.cod_pyto
join cr_dep_pyto lookup "cod_pyto"
end
INTERFACE
POSITION 0 0 318 254
LABEL "Seleccionar Departamento"
SYSMENU
BEGIN
CONTROL AS BOX
POSITION 0 0 310 229
FOREGROUND RGB 0 0 0
ATTACH ALL
NOLABEL
NOBORDER
TABLE departamentos
BEGIN
CONTROL AS BUTTON
POSITION 27 192 112 25
LABEL "Aceptar"
ICON 2 FILE "_MBSTDBMP_" CENTER
COMMAND Aceptado
CONTROL AS BUTTON
```

```
POSITION 171 192 112 25
LABEL "Cancelar"
ICON 3 FILE "_MBSTDBMP_" CENTER
COMMAND Close
CONTROL lista AS LIST
POSITION 27 16 261 160
LABEL "Cod|Departamento"
BORDER DOUBLE DOWN
ICON FILE "_MBSTDBMP_"
LISTTYPE LISTCOLS 2
CURSORTYPE SQL
SELECTTYPE SIMPLE
END
END
end
END
OBJECTS BEGIN
oDpto AS cDpto
END
CODE CLASS cEmp BEGIN
//{{CODEBEGIN
public Listar (cod as smallint, nomdpto as char)
objects.begin
nlin npag as smallint
scursor as SqlCursor
prn as PrnDocument
end
begin
dpto.Text=nomdpto;
scursor.Prepare("select dni, nombre, apellidos from empleados where
cod_dpto=?"
+" order by nombre" );
if scursor.Error then
"Error en el Cursor".Trace;
scursor.Into(dni,nombre,apellidos);
scursor.Open(cod);
nlin=npag=1;
scursor.Fetch;
pag.Text=npag;
while scursor.Found do begin
if nlin>miGrupo.Count then begin
pag.Text=npag;
++npag;
prn.SendPage(Self);
Self.Clear;
nlin=1;
end
miGrupo.CurrentRow=nlin;
miGrupo.PrintVars;
scursor.Fetch;
++nlin;
end
scursor.Close;
scursor.Free;
prn.SendPage(Self);
prn.Preview;
end
//{{CODEEND
END
CODE CLASS cDpto BEGIN
//{{CODEBEGIN
```

```
On Open
begin
lista.LoadSelect("select cod_dpto,nombre from departamentos order by
cod_
end
On Command Aceptado
objects
begin
oEmp as cEmp
posicion codigo as smallint
nombre as char(50)
end
begin
posicion=lista.Selected;
codigo=lista.GetListColumnText(posicion,1);
nombre=lista.GetListColumnText(posicion,2);
oEmp.Listar(codigo,nombre);
Close;.end
//{{CODEEND
END
CODE BEGIN
//{{CODEBEGIN
main
begin
Sql.AttachConnection("ConexionGeneral");
Sql.Connect;
oDpto.Run;
Sql.Disconnect;
Sql.DettachConnection;
end
//{{CODEEND
END
```

15. Módulo mInicializacion.

```
CONSTANTS begin
end
CLASSES BEGIN
cSeleccion is Form begin
objects begin
nombre as Char
end
INTERFACE
POSITION 0 0 415 384
LABEL "Seleccionar la Empresa "
SYSMENU
BEGIN
CONTROL AS BOX
POSITION 0 0 407 359
FOREGROUND RGB 0 0 0
ATTACH ALL
NOLABEL
NOBORDER
BEGIN
CONTROL arbol AS LIST
POSITION 54 32 288 256
LABEL "List"
BORDER DOUBLE DOWN
```

```
ICON FILE "_MBSTDBMP_"
CURSORTYPE TREEVIEW
SELECTTYPE SIMPLE
SCROLL VERTICAL HORIZONTAL
CONTROL AS BUTTON
POSITION 54 304 90 32
LABEL "Aceptar"
ICON 2 FILE "_MBSTDBMP_" CENTER
COMMAND aceptar
CONTROL AS BUTTON
POSITION 252 304 90 32
FOCUS
LABEL "Cancelar"
ICON 3 FILE "_MBSTDBMP_" CENTER
COMMAND Close
END
END
end
END
OBJECTS BEGIN
oSeleccion AS cSeleccion
END
CODE CLASS cSeleccion BEGIN
//{{CODEBEGIN
On Open
objects
begin
fich as char
err icono handle as integer
end
begin
//se posiciona en el directorio padre del proyecto
ChDir(ProjectDir);.ChDir("..");
//Añade al arbol el nodo del directorio padre
arbol.AddTreeString(0,GetCwd,true,150,149);
//abre el directorio
handle=OpenDir(GetCwd,fich);
do
begin
if TestFile(fich,"f")
then icono=164;
else icono=149;
arbol.AddTreeString(1,fich,false,150,icono);
err=ReadDir(handle,fich);
end
while err<>0;
end

On Command aceptar
objects
begin
pos as smallint
end
begin
pos=arbol.Selected;
if pos<>0
then begin
nombre=arbol.GetListText(pos);
if (nombre.Count(".dbs"))==0
then Ok("Debe seleccionar una base de datos", "MENSAJE",true);
else begin
```

```
nombre.Replace(".dbs","",1);
Close;
end
end
else Ok("Debe seleccionar una base de datos o pulsar el botón Nuevo",
end

public function cargarDir return char
begin
Self.Run;
return nombre;
end
//{{CODEEND
END
CODE BEGIN
//{{CODEBEGIN
private borrarTablas
objects
begin
st as SqlStatement
end
begin.//Borrar el contenido de la tabla Empleados
st.Prepare("delete from empleados");
st.Execute;
if st.Error<0 then
st.ErrMsg.Trace;
("Tabla Empleados: "+st.Rows+" tuplas").Trace;
//Libera la memoria asignada
st.Free;
//Borrar el contenido de la tabla Departamentos
st.Prepare("delete from departamentos");
st.Execute;
if st.Error<0 then
st.ErrMsg.Trace;
("Tabla Departamentos: "+st.Rows+" tuplas").Trace;
//Libera la memoria asignada
st.Free;
//Borrar el contenido de la tabla Empresas
st.Prepare("delete from empresas");
st.Execute;
if st.Error<0 then
st.ErrMsg.Trace;
("Tabla Empresas: "+st.Rows+" tuplas").Trace;
//Libera la memoria asignada
st.Free;
//Borrar el contenido de la tabla Proyectos
st.Prepare("delete from proyectos");
st.Execute;
if st.Error<0 then
st.ErrMsg.Trace;
("Tabla Proyectos: "+st.Rows+" tuplas").Trace;
//Libera la memoria asignada
st.Free;
//Borrar el contenido de la tabla Cargos
st.SqlExec("delete from cargos");
if st.Error<0 then
st.ErrMsg.Trace;
("Tabla Cargos: "+st.Rows+" tuplas").Trace;
//Libera la memoria asignada
st.Free;
end
```

```
main
objects
begin
msg as boolean
dir dirCompleto as char
nuevo as boolean
end
begin
// Se invoca la ejecución del método cargarDir de la clase cSeleccion
nuevo=false;.dir=oSeleccion.cargarDir;
SetIni("Environment ConexionGeneral","DBPATH",GetCwd,NULL);
SetIni("Environment ConexionGeneral","DBNAME",dir,NULL);
//conexion a la base de datos
Sql.AttachConnection("ConexionGeneral");
Sql.Connect;
//Solicitud de confirmación
msg=Ok("Desea vaciar el contenido de las tablas", "PRECAUCION",false);
if msg then begin
borrarTablas;
end
else begin
Exit;
end
Sql.Disconnect;
Sql.DettachConnection;
end
//{{CODEEND
END
```

16. Módulo mAbrir.

```
CONSTANTS begin
end
CLASSES BEGIN
cSeleccion is Form begin
objects begin
nombre as Char
end
INTERFACE
POSITION 0 0 415 384
LABEL "Seleccionar la Empresa "
SYSMENU
BEGIN
CONTROL AS BOX
POSITION 0 0 407 359
FOREGROUND RGB 0 0 0
ATTACH ALL
NOLABEL
NOBORDER
BEGIN
CONTROL arbol AS LIST
POSITION 54 32 288 256
LABEL "List"
BORDER DOUBLE DOWN
ICON FILE "_MBSTDBMP_"
CURSORTYPE TREEVIEW
SELECTTYPE SIMPLE
SCROLL VERTICAL HORIZONTAL
```

```
CONTROL AS BUTTON
POSITION 54 304 90 32
LABEL "Aceptar"
ICON 2 FILE "_MBSTDBMP_" CENTER
COMMAND aceptar
CONTROL AS BUTTON
POSITION 252 304 90 32
LABEL "Cancelar"
ICON 3 FILE "_MBSTDBMP_" CENTER
COMMAND Close
END
END
end
END
OBJECTS BEGIN
oSeleccion AS cSeleccion
END
CODE CLASS cSeleccion BEGIN
//{{CODEBEGIN

On Open
objects
begin
fich as char
err icono handle as integer
end
begin
//se posiciona en el directorio padre del proyecto
ChDir(ProjectDir);
ChDir("..");.//Añade al arbol el nodo del directorio padre
arbol.AddTreeString(0,GetCwd,true,150,149);
//abre el directorio
handle=OpenDir(GetCwd,fich);
do
begin
if TestFile(fich,"f")
then icono=164;
else icono=149;
arbol.AddTreeString(1,fich,false,150,icono);
err=ReadDir(handle,fich);
end
while err<>0;
end

On Command aceptar
objects
begin
pos as smallint
end
begin
pos=arbol.Selected;
if pos<>0
then begin
nombre=arbol.GetListText(pos);
if (nombre.Count(".dbs"))==0
then Ok("Debe seleccionar una base de datos", "MENSAJE",true);
else begin
nombre.Replace(".dbs","",1);
Close;
end
end
```

```
else Ok("Debe seleccionar una base de datos o pulsar el botón Nuevo",
end

public function cargarDir return char
begin
Self.Run;
return nombre;
end
//{{CODEEND
END
CODE BEGIN
//{{CODEBEGIN
main
objects
begin
dir as char
end
begin
// Se invoca la ejecución del método cargarDir de la clase
cSeleccion.dir=oSeleccion.cargarDir;
SetIni("Environment ConexionGeneral","DBPATH",GetCwd,NULL);
SetIni("Environment ConexionGeneral","DBNAME",dir,NULL);
end
//{{CODEEND
END
```

17. Módulo mNuevo.

```
CONSTANTS begin
end
CLASSES BEGIN
cNuevo is Form begin
objects begin
empresa as Char
end
INTERFACE
POSITION 0 0 377 125
LABEL "Nombre de la nueva Empresa"
SYSMENU
BEGIN
CONTROL AS BOX
POSITION 0 0 369 100
FOREGROUND RGB 0 0 0
ATTACH ALL
NOLABEL
NOBORDER
BEGIN
CONTROL edit0 AS EDIT
POSITION 9 16 342 25
BORDER DOUBLE DOWN
DATATYPE CHAR
CONTROL AS BUTTON
POSITION 162 64 90 25
LABEL "Aceptar"
ICON 2 FILE "_MBSTDBMP_" CENTER
COMMAND aceptar
CONTROL AS BUTTON
```

```
POSITION 261 64 90 25
LABEL "Cancelar"
ICON 3 FILE "_MBSTDBMP_" CENTER
COMMAND Close
END
END
end
END
OBJECTS BEGIN
oNuevo AS cNuevo
END
CODE CLASS cNuevo BEGIN
//{{CODEBEGIN
On command aceptar
begin
empresa=edit0.Text;
if empresa<>""
then Close;
else Ok("Introduzca un nombre de empresa", "MENSAJE",true);
end
public nuevoNombre return char
begin
Self.Run;
return empresa;
end
//{{CODEEND
END.CODE BEGIN
//{{CODEBEGIN
private crearNuevo(nombre as char)
objects
begin
pathOrigen pathDestino as char
end
begin
// situarse en el directorio del proyecto
ChDir(ProjectDir);
ChDir("..");
pathOrigen=GetCwd+"\datos.dbs";
pathDestino=GetCwd+"\"+nombre+".dbs";
// crear el nuevo subdirectorio
Self.MkDir(pathDestino);
// copiar la tablas base al nuevo subdirectorio
if Copy(pathOrigen+"\*.*",pathDestino+"\")
then ("Directorio "+pathDestino+" creado con exito").Trace;
else begin
("No se ha podido crear el Directorio "+pathDestino).Trace;
RmDir(pathDestino);
end
end
main
objects
begin
msg as boolean
nombre as char
end
begin
// Se invoca la ejecución del método cargarDir de la clase cSeleccion
nombre=oNuevo.nuevoNombre;
crearNuevo(nombre);
SetIni("Environment ConexionGeneral","DBPATH",GetCwd,NULL);
SetIni("Environment ConexionGeneral","DBNAME",nombre,NULL);
```

```
end
//{{CODEEND
END
```

18. Módulo mBorrar.

```
CONSTANTS begin
end
CLASSES BEGIN
cSeleccion is Form begin
objects begin
nombre as Char
end
INTERFACE
POSITION 0 0 415 384
LABEL "Seleccionar la Empresa "
SYSMENU
BEGIN
CONTROL AS BOX
POSITION 0 0 407 359
FOREGROUND RGB 0 0 0
ATTACH ALL
NOLABEL
NOBORDER
BEGIN
CONTROL arbol AS LIST
POSITION 54 32 288 256
LABEL "List"
BORDER DOUBLE DOWN
ICON FILE "_MBSTDBMP_"
CURSORTYPE TREEVIEW
SELECTTYPE SIMPLE
SCROLL VERTICAL HORIZONTAL
CONTROL AS BUTTON
POSITION 54 304 90 32
LABEL "Aceptar"
ICON 2 FILE "_MBSTDBMP_" CENTER
COMMAND aceptar
CONTROL AS BUTTON
POSITION 252 304 90 32
LABEL "Cancelar"
ICON 3 FILE "_MBSTDBMP_" CENTER
COMMAND Close
END
END
end
END
OBJECTS BEGIN
oSeleccion AS cSeleccion
END
CODE CLASS cSeleccion BEGIN
//{{CODEBEGIN

On Open
objects
begin
fich as char
err icono handle as integer
```

```
end
begin
//se posiciona en el directorio padre del proyecto
ChDir(ProjectDir);
ChDir("..");.//Añade al arbol el nodo del directorio padre
arbol.AddTreeString(0,GetCwd,true,150,149);
//abre el directorio
handle=OpenDir(GetCwd,fich);
do
begin
if TestFile(fich,"f")
then icono=164;
else icono=149;
arbol.AddTreeString(1,fich,false,150,icono);
err=ReadDir(handle,fich);
end
while err<>0;
end

On Command aceptar
objects
begin
pos as smallint
end
begin
pos=arbol.Selected;
if pos<>0
then begin
nombre=arbol.GetListText(pos);
if (nombre.Count(".dbs"))==0
then Ok("Debe seleccionar una base de datos", "MENSAJE",true);
else begin
nombre.Replace(".dbs","",1);
Close;
end
end
else Ok("Debe seleccionar una base de datos o pulsar el botón Nuevo",
end

public function cargarDir return char
begin
Self.Run;
return nombre;
end
//{{CODEEND
END
CODE BEGIN
//{{CODEBEGIN
private borrarBD(nombre as char)
objects
begin
path as char
end
begin
// situarse en el directorio del proyecto.ChDir(ProjectDir);
ChDir("..");
path=GetCwd+"\"+nombre+".dbs";
// borrar todos los ficheros del subdirectorio
Delete(path+"\*.*");
// borrar el subdirectorio
RmDir(path);
```

```
end
main
objects
begin
msg as boolean
nombre as char
end
begin
// Se invoca la ejecución del método cargarDir de la clase cSeleccion
nombre=oSeleccion.cargarDir;
//Solicitud de confirmación
msg=Ok("Desea eliminar la empresa: "+nombre, "PRECAUCION",false);
if msg then begin
borrarBD(nombre);
Ok("Empresa "+nombre+" borrada", "MENSAJE",true);
end
else begin
Exit;
Ok("Empresa "+nombre+" NO borrada", "MENSAJE",true);
end
end
//{{CODEEND
END
```

19. Módulo mAcerca.

```
CONSTANTS begin
end
CLASSES BEGIN
cAcerca is Form begin
objects begin
end
INTERFACE
POSITION 0 0 256 219
LABEL "Acerca de..."
SYSMENU
BEGIN
CONTROL AS BOX
POSITION 0 0 248 194
FOREGROUND RGB 0 0 0
ATTACH ALL
NOLABEL
NOBORDER
BEGIN
CONTROL AS TEXT
POSITION 14 13 175 22
NOBORDER
DATATYPE CHAR
TAGS "Gestión de Proyectos Multiempresa"
CONTROL AS TEXT
POSITION 14 39 175 22
NOBORDER
DATATYPE CHAR
TAGS "Version 1.0"
CONTROL AS TEXT
POSITION 14 91 175 22
NOBORDER
DATATYPE CHAR
TAGS "Septiembre 1999"
```

```
CONTROL AS TEXT
POSITION 14 117 175 22
NOBORDER
DATATYPE CHAR
TAGS "Multibase COSMOS Versión 3.0."
CONTROL AS BUTTON
POSITION 147 156 87 22
FOCUS
LABEL "Salir"
ICON 2 FILE "_MBSTDBMP_" CENTER
COMMAND Close
CONTROL AS TEXT
POSITION 14 65 217 22
NOBORDER
DATATYPE CHAR
TAGS "Realizado por: B. Cristina Pelayo Gª-Bustelo"
END
END
end
END
CODE BEGIN
//{{CODEBEGIN
main
objects.begin
oAcerca as cAcerca
end
begin
oAcerca.Run;
end
//{{CODEEND
END
```

Títulos de la Colección

Ingeniería Informática

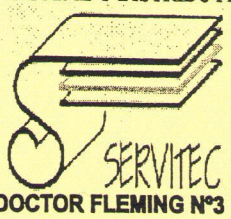
Nº CUADERNO	TITULO	ISBN
1	ANÁLISIS Y DISEÑO ORIENTADO A OBJETOS	84-8497-802-8
2	DISEÑO DE PÁGINAS WEB USANDO HTML	84-8497-803-6
3	FORMATOS GRÁFICOS	84-8497-804-4
4	PROGRAMACIÓN EN LENGUAJE C	84-8497-805-2
5	ADMINISTRACIÓN DE WINDOWS N 4.0	84-8497-899-0
6	GUÍA DE USUARIO DE DERIVE PARA WUIDOWS	84-8497-919-9
7	GUÍA DE REFERÊNCIA DE MULTIBASE COSMOS	84-8416-001-7
8	GESTIÓN DE UN TALLER MULTIBASE CÓSMOS	84-8416-034-3
9	EJERCICIOS DE LÓGICA INFOMÁTICA	84-8416-357-1
10	CONEPTOS BÁSICOS DE PROCESADORES DE LENGUAJE	54-8416-889-1
11	INTRODUCCIÓN A LA ADMINISTRACIÓN DE UNIX	84-8416-570-1
12	LÓGICA PROPOSICIONAL PARA LA INFOMÁTICA	84-8416-613-9
13	PROGRAMACIÓN PRÁCTICA EN PROLOG	84-8416-612-0
14	LA HERRAMIENTA CASE META COSMOS	84-699-0054-4
15	GUIA DE LENGUAJE COOL DE MULTIBASE COSMOS	84-699-0053-6
16	GUÍA DE REFERÊNCIA PROGRESS	84-699-2083-9
17	GESTIÓN DE DEPOSITO DENTAL CON PROGRESS	84-699-2082-0
18	GUIA DE DISEÑO Y CONS. REPOSICIÓN MUL. COSMOS	84-699-2081-2
19	GESTIÓN Y ADMIN. DE UN COLEGIO MAYOR MUL-COSMOS	84-699-2080-4
20	MODELADO DE SOFWARE EN UML	84-699-2079-0
21	EL LENGUAJE DE PROGRAMACIÓN JAVA	84-699-3880-0
22	PRINCIPIOS DE ALGORITMIA	84-699-6537-9
23	LISTAS, PILAS Y COLAS	84-699-6536-0
24	RECURSIVIDAD	84-699-6535-2
25	ARBOLES	84-699-6849-1
26	CONJUNTOS Y TABLAS DE DISPERSIÓN	84-699-7398-3
27	ALGORITMOS DE ORDENACIÓN	84-699-7397-5
28	TEORÍA DE GRAFOS	84-699-8362-8
29	IINTR. AL PROCES. EFEC. DE TEXTOS CON MICROSOFT WORD	84-699-9618-5
30	ORACLE SQL	84-688-0420-7
31	TÉCNICAS DE DISEÑO DE ALGORITMOS	84-688-1764-3
32	LA PLATAFORMA . NET	84-688-3449-1
33	EJERC. DE HOJAS DE CÁL. EXCEL APLICADOS A LA GES.EMPRES.	84-688-3450-5
34	TIPOS ABSTRACTOS DE DATOS	84-688-4209-5
35	INTERPRETES T DISEÑO DE LENGUAJES DE PROGRAMACIÓN	84-688-4210-9
36	LENGUAJES Y AUTOMATAS EN PROCESADORES DE LENGUAJE	84-688-4211-7
37	METODOLOGIA DE LA PROGRAMACIÓN: GUIA DEL ALUMNO	84-688-5901-4
38	ANÁLISIS SEMÁNTICO EN PROCESADORES DE LENGUAJE	84-688-6208-8
39	ARQUITECTURA WEB EN APLICACIONES JAVA/J2EE	84-688-6580-9
40	ALGORITMICA CON FORTRAN 90	84-688-7250-4
41	TABLAS DE SÍMBOLOS EN PROCESADORES DE LENGUAJES	84-688-7631-3
42	INTRODUCCIÓN A LA COMINICACIÓN PERSONA MÁQUINA	84-688-8362-X
43	INTRODUC. A LA PROG. ORIENTADA A OBJETOS CON JAVA	84-688-8826-6
44	DESARR. APLIC. EN SISTEM. DISTRIBUIDOS E INTERNET	84-689-3379-1
45	LÓGICA DE PREDICADOS	84-689-3380-5
46	LENGUAJE C#	84-689-5893-X
47	INTEGRACIÓN DE APLICACIONES OFIMÁTICAS	84-689-5892-1
48	INFORMÁTICA GENERAL	84-689-7033-6
49	PROYECTOS INFORMÁTICOS	

IMPRIME Y DISTRIBUYE

C/DOCTOR FLEMING Nº3
33005 OVIEDO
TLF-FAX 985 250581
E-mail:copisteriaservitec@fade.es

Consultor Editorial
Juan Manuel Cueva Lovelle
cueva@lsi.uniovi.es

IMPRIME Y DISTRIBUYE



C/DOCTOR FLEMING Nº3
33005 OVIEDO
TLF-FAX 985 250581
www.fade.es/copisteriaservitec
E-mail:copisteriaservitec@fade.es