

**RPP**

REVISTA PROFESIONAL PARA PROGRAMADORES

**SECCIONES:**

- ◆ **DELPHI:**  
Cómo programar DLLs
- ◆ **FORUM C:**  
Especificaciones de requerimientos
- ◆ **TRUCOS DEL LECTOR:**  
Cómo detectar si Windows está activo

# HERRAMIENTAS OS/2

**VISUAL AGE C++**  
**VISUAL AGE SMALLTALK**  
**VISPRO/REXX**  
**VX-REXX**  
**DB2/2 2.1**

## ¿QUÉ ES **JAVA?**

**PROGRAMAS RESIDENTES:**  
**USO DE LA INTERRUPCIÓN**  
**DE TECLADO**



[Canarias, Ceuta, Melilla 975 pts.]

00016



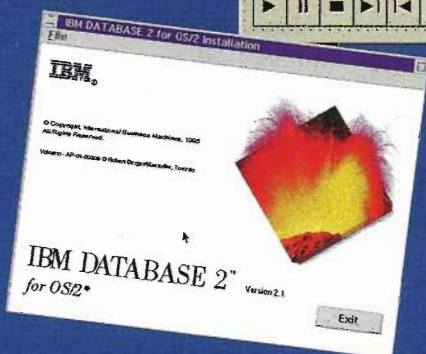
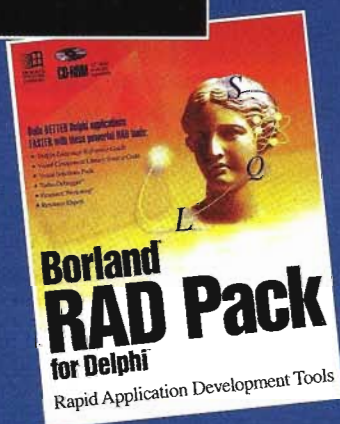
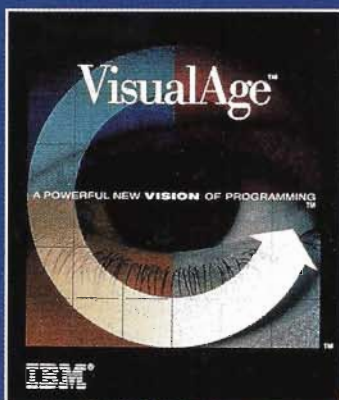
8 424094 405039



# EN ESTE NUMERO ...

Marzo 1996

Número 16



- 3 Editorial
- 5 En este número ...
- 6 Novedades
- 10 Opinion++

## HERRAMIENTAS OS/2

- 12 El sistema operativo OS/2  
La respuesta del Gigante Azul
- 17 VisualAge SmallTalk
- 21 VisualAge C++ para OS/2 3.0
- 26 Shareware para OS/2
- 31 Programación Visual en REXX  
VisPro/REXX y VX-REXX
- 37 Bases de datos relacionales  
Novedades en DB2/2 versión 2.1

## DELPHI

- 43 Rad Pack  
Herramientas de productividad para Delphi
- 49 Programación con Delphi  
Creación y utilización de librerías DLL con Delphi

## INVESTIGACIÓN

- 53 El lenguaje JAVA  
Programar para Internet
- 68 Operaciones booleanas entre sólidos (II)
- 87 Cómo programar el OLE 2.0 con Borland C++ 4.5 and Database Tools

## FORUM C

- 61 Especificaciones de Requerimientos
- 73 Diseño de programas residentes (V)  
Uso de la interrupción de teclado
- 79 Programación gráfica. Tipos y problemática de las proyecciones en perspectiva cónica
- 83 ObjectWindows 2.5  
La clase Tdialog  
Los cuadros de diálogo de Windows

- 93 Bolsa de trabajo/Cartas del lector
- 95 Trucos
- 96 Preguntas y Respuestas
- 98 Este mes nuestro CD-ROM incluye ...

# Programación Visual en REXX

## VisPro/REXX y VX-REXX

*En este artículo vamos a tratar los fundamentos de la programación visual en REXX y hablaremos de dos de sus grandes productos para OS/2: VisPro/REXX de Hockware y VX-REXX de Watcom.*

**H**ACE años, en 1985, tuve mi primer contacto con el lenguaje REXX en el entorno VM/CMS. He de decir que fue una experiencia realmente gratificante y positiva. Me enfrentaba a un lenguaje potente, flexible y a la vez sencillo. Un lenguaje que interaccionaba con el sistema de una manera asombrosamente natural. Un lenguaje estructurado que me permitía programar profesionalmente.

En el 85, el mundo de los PCs todavía estaba en pañales. Era impensable una implementación de REXX en PC. Pero las cosas fueron avanzando rápidamente y en pocos años, dispusimos de REXX tanto para DOS como para OS/2.

En el entorno microinformático REXX presentaba dos importantes novedades respecto a su implementación de *mainframe*: La primera consistía en ofrecer la posibilidad de ampliar las funciones incorporadas de REXX con funciones externas escritas en otros lenguajes. La segunda, en posibilitar llamadas al intérprete de REXX desde programas escritos en otro lenguaje.

Estas dos innovaciones son las que han permitido la existencia, en la actualidad, de aplicaciones que usan un entorno de programación visual basadas en REXX. Para poder acceder a las APIs de *Presentation Manager* (PM) de OS/2, necesitamos utilizar funciones externas escritas en un lenguaje distinto al REXX ya que éste no tiene acceso direc-

to a las APIs. Por otra parte, REXX no interactúa directamente con el entorno PM por lo que su código debe ser ejecutado desde otro programa que le proporcione el entorno PM. Aquí tenemos, pues las bases de la programación visual en REXX. Toda la lógica de nuestras aplicaciones está escrita en REXX. Cuando hay interacción con PM, utilizamos funciones externas y el entorno PM nos viene dado por el ejecutable que se genera.

Los productos que nos permiten la programación visual en REXX siguen un esquema muy similar. Proporcionan

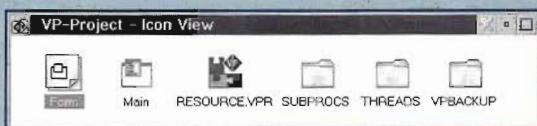
### **Francesc Rosés**

*Francesc Rosés, nacido en Barcelona en 1958, es licenciado en Filología Catalana por la Universidad de Barcelona. Desde 1985 desarrolla en el Centre d'Informàtica de la Universitat de Barcelona (CIUB) la coordinación informática de diversos proyectos de lexicografía computacional. Profesor del Máster de Lingüística Aplicada de la Universidad de Barcelona. Desde 1994 es el responsable de la sección de Microinformática del CIUB. Su dirección Internet es froses@fenix.ird.ub.es.*

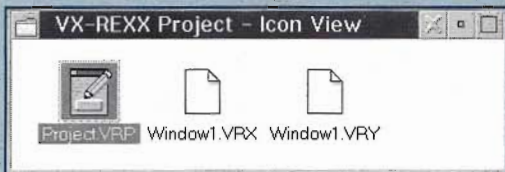
### **José María Blasco**

*José María Blasco, nacido en 1960, es Licenciado en Matemáticas por la Universidad de Barcelona. Ha sido profesor de Programación en la Facultad de Informática de Barcelona, y Coordinador Nacional de la red EARN en Alemania. Sus temas de interés más recientes son las redes, la programación orientada a objeto, las bases de datos, y todo lo relacionado con OS/2. Su dirección Internet es jmblasco@eim.ub.es.*

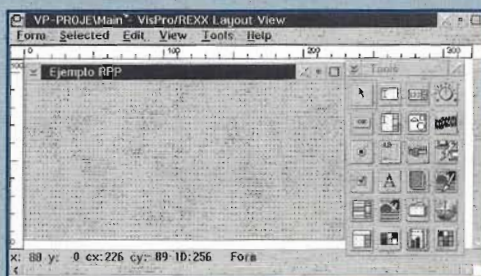




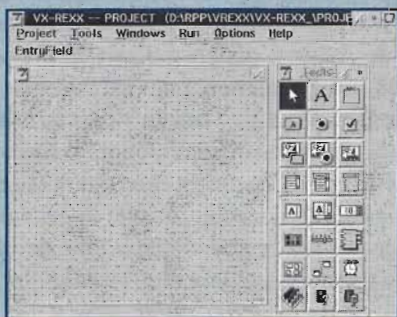
**Figura A** Aspecto inicial de una carpeta de proyectos de VisPro/REXX



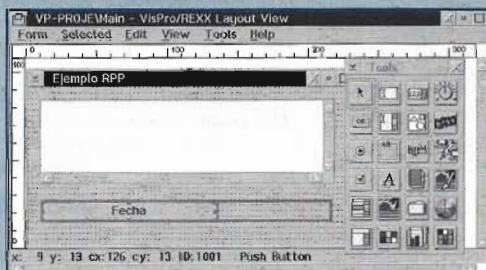
**Figura B** Aspecto inicial de una carpeta de proyectos de VX-REXX



**Figura C** Aspecto inicial de la GUI de VisPro/REXX



**Figura D** Aspecto inicial de la GUI de VX-REXX



**Figura E** Modificación del tamaño de un objeto en VisPro/REXX

una interfaz gráfica (*GUI*) que nos permite manejar los distintos objetos (campos de entrada, botones, contenedores, etc.) de manera visual (arrastrando y soltando para incluir objetos o cambiarlos de posición y tamaño). Además, la *GUI* sirve también para asignar comportamiento a esos objetos. Este comportamiento está sujeto a eventos. Que un botón sea pulsado es un evento. Cuando el evento se produce, podemos realizar una serie de acciones que configurarán su comportamiento. El lenguaje que utilizamos para crear los comportamientos es *REXX*. La *GUI* suele proporcionarnos un editor que nos facilita la escritura de esos comportamientos. Si seleccionamos un evento de un objeto, se nos abre una ventana de edición. Allí podemos escribir las acciones necesarias para ese evento.

Cada objeto tiene un conjunto de acciones posibles. Hay acciones comunes para todos los objetos (todo objeto puede hacerse visible o invisible) pero las hay que sólo pueden realizarse en algunos objetos. Podemos pulsar un botón, pero no un texto. No tiene sentido hacer un doble clic en un botón, pero puede tenerlo en una lista (*list box*). La *GUI* nos facilita la escritura de estas acciones para añadir comportamiento a un objeto.

Pongamos un ejemplo sencillo. Pretendemos escribir una aplicación sencilla que nos muestre una caja de diálogo con una lista y un botón. Queremos, además, que cada vez que pulsemos ese botón se nos añada al final de la lista la fecha actual. La *GUI* nos proporciona una ventana vacía y nos muestra el conjunto de objetos disponibles. Seleccionamos el objeto lista y lo arrastramos hasta la ventana vacía. Seleccionamos el objeto botón y repetimos la operación. Una vez situados en la ventana, modificamos su tamaño y damos un título al botón. Utilizando una *GUI*, estamos diseñando otra. Pero sólo tenemos los objetos. Hay que dar comportamiento a la aplicación. Para ello, accedemos a la lista de eventos del objeto botón



y escogemos el evento *when clicked* (“al hacer clic”). Aparece el editor de eventos y arrastramos hasta el objeto lista. Aparece el conjunto de acciones que podemos realizar con el objeto lista. Escogemos *add item to end* (“añadir un elemento al final”). Ahora sólo hay que especificar qué valor añadimos. Sencillo y efectivo.

Una vez tenemos la aplicación diseñada, sólo nos queda probarla. Para ello tenemos dos opciones. Crear un programa ejecutable y ejecutarla, o ejecutarla sin generar el programa ejecutable. Si aparece algún error, siempre podemos ejecutar la aplicación en modo depuración.

El esquema que aquí hemos esbozado a grandes rasgos es el que se utiliza en todos los productos disponibles en el mercado.

Hablaremos aquí de dos de los productos de más éxito en el ámbito de la programación visual para el entorno OS/2: *VisPro/REXX*, de *Hockware*, y *VX-REXX*, de *Watcom*. Ambos son unos excelentes productos que facilitan en gran manera el desa-

rollo de aplicaciones. Estos productos presentan posibilidades muy similares. Yo destacaría dos diferencias: la interfaz utilizada para el desarrollo y el método de generación de ejecutables.

Hablemos primero de las posibilidades que nos ofrecen. Ya hablaremos luego de las diferencias.

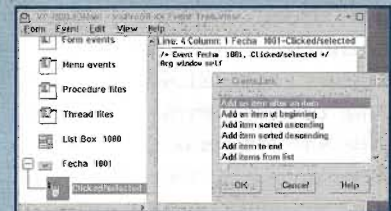
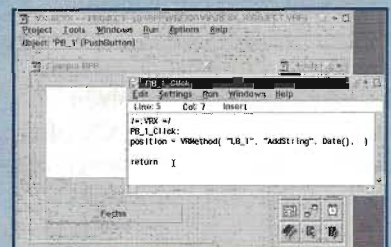
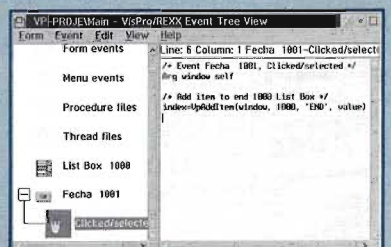
Ambos productos nos ofrecen casi al completo toda la gama de objetos disponibles en *PM* (ver **Tabla A**).

*VisPro/REXX*, dispone de algunos objetos realmente interesantes. En el apartado de multimedia hay que destacar el *MMPM Video Object* que nos permite visualizar un vídeo dentro de cualquier caja de diálogo que diseñemos. Además tenemos control sobre el vídeo. Podemos hacer una pausa, rebobinar, aumentar o disminuir el volumen o, simplemente, pararlo. También hay que agradecer a *Hockware* que nos haya incluido el *Circular Slider*, utilísimo para el control de volumen.

En el apartado de gráficos, *VisPro/REXX* nos obsequia con un nuevo

**Tabla A.- Objetos PM disponibles**

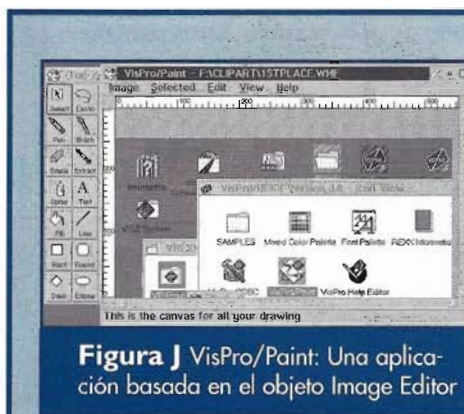
	<u>Vispro/REXX</u>	<u>VX-REXX</u>
Business Graphic	Si	Si
Circular Slider	Si	No
Color Selector	Si	No
Combo Box	Si	Si
Connection	No	Si
Container	Si	Si
Check Button	Si	Si
Check Button 3 states	Si	Si
DDE Client	Si	Si
Entry Field	Si	Si
Free Form Entry	Si	No
Graphic	Si	Si
Group Box	Si	Si
Image Editor	Si	No
Image Push Button	Si	Si
Image Radio Button	No	Si
List Box	Si	Si
MMPM/2 Video Object	Si	No
Multy Line Entry	Si	Si
Notebook	Si	Si
Pattern Selector	Si	No
Push Button	Si	Si
Query	No	Si
Radio Button	Si	Si
Slider	Si	Si
Spin Button	Si	Si
Text	Si	Si
Value Set	Si	Si


**Figura F** Lista de eventos para un botón en VisPro/REXX

**Figura G** Lista de acciones posibles para una lista en VisPro/REXX

**Figura H** Editor de eventos de VX-REXX

**Figura I** Editor de eventos de VisPro/REXX

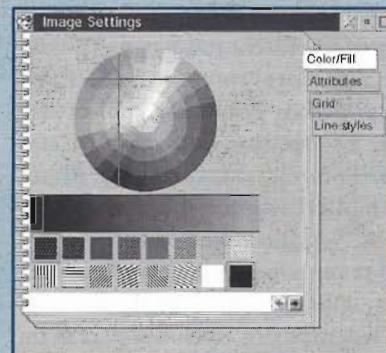


objeto en su versión 3.0: el *Image Editor*. Este objeto es, tal como su nombre indica, un editor de imágenes que se integra como cualquier otro objeto en nuestras aplicaciones. Tenemos un buen ejemplo de la potencia de este objeto en la aplicación *VisPro Paint* que se incluye en el producto. La **Figura J** nos muestra el aspecto de esta aplicación y la **Tabla B** nos indica algunas de sus posibilidades.

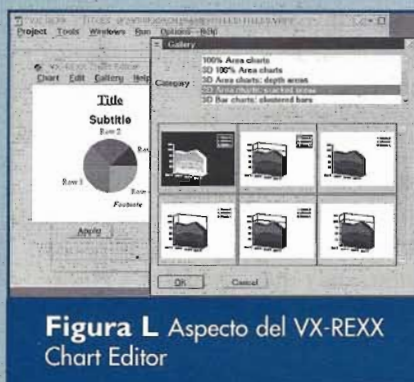
También en el apartado de imágenes *VisPro/REXX* incluye dos objetos interesantes: el *Color Selector*, un objeto que nos permite seleccionar fácilmente un color, y el *Pattern Selector*, que nos permite seleccionar una trama. La **Figura K** nos muestra el aspecto de estos objetos. Por otra parte, cabe remarcar el editor de gráficos de negocios de *VX-REXX*. Es realmente magnífico tanto por su sencillez como por el abanico de modalidades gráficas que presenta. Las acciones disponibles para cada objeto, también son similares. *VisPro/REXX* ofrece, además, la posibilidad de enviar cualquier mensaje *PM* a un objeto.



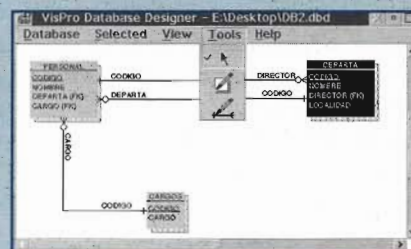
**Figura J** VisPro/Paint: Una aplicación basada en el objeto Image Editor



**Figura K** Ejemplo de uso de los objetos Color Selector y Pattern Selector en VisPro/REXX



**Figura L** Aspecto del VX-REXX Chart Editor



**Figura M** Aspecto del VisPro Database Designer

Ambos productos ofrecen una serie de capacidades destacables: Control de *Drag & Drop* para todos los obje-

tos, editor de menús, menús emergentes, ayuda interactiva mediante conexiones a *help*, posibilidad de activar una línea de información, editor de recursos, capacidades *multithread*, depurador, acceso a bases de datos, etc. Las diferencias se establecen, quizá, en las posibilidades que presentan estas capacidades. Así, el depurador de *VisPro/REXX* permite el seguimiento de los distintos *threads* de las aplicaciones. No se ha observado esta capacidad en el depurador de *VX-REXX*.

Uno de los aspectos que más están cuidando últimamente los desarrolladores es la interfaz con las bases de datos. Tanto *Hockware* como *Watcom* han apostado por ello y nos ofrecen dos buenas soluciones aunque con aproximaciones distintas.

*VX-REXX* nos ofrece una aproximación más clásica presentándonos en un *container* el conjunto de tablas

**Tabla B.-** Posibilidades del objeto Image Editor

<b>Formatos soportados:</b>	BMP (OS/2 y Windows 3.0), TIF, PCX, GIF, Truevision Targa/Vista bitmap, Amiga IFF / ILBM Interleaved bitmap, YUV12C M-Motion Frame Buffer, IBM KIPS, IBM Image Access eExecutive, X Windows bitmap, Archimedes Sprite format from RiscOS, IBM Printer Page Segment.
<b>Acciones:</b>	Capturador de pantallas Conversión a blanco y negro Impresión en blanco y negro o color en thread aparte Importación-Exportación a(-de) todos los formatos gráficos soportados. Posibilidad de copiar y pegar desde el clipboard Posibilidad de recortar la imagen (crop) Obtener atributos de una imagen Rotar la imagen Escalar la imagen Utilizar hasta 14 herramientas para manipular la imagen Etc.



de una base de datos (**Figura N**). La información de cada tabla se nos presenta en un *Notebook* (**Figura O**).

*VisPro/REXX* opta por una aproximación gráfica (**Figura M**). Nos presenta las tablas con su estructura y sus enlaces. Para obtener más datos de una tabla, hacemos doble clic en ella y nos aparece un *notebook* con la información detallada de su estructura. El mismo entorno gráfico nos sirve para diseñar nuestras propias bases de datos. Una vez diseñada la base de datos, podemos exportar su defini-

ción a una *DDL Macro* o a una *DDL*. La *DDL Macro* consiste en un *CMD* que contiene las instrucciones necesarias para crear la base de datos, sus tablas y las relaciones entre las tablas tal y como la hemos definido gráficamente (**Listado 1**). Una *DDL* no es más que el conjunto de instrucciones necesarias para crear la base de datos (**Listado 2**). *VisPro Database Designer* permite también exportar el esquema visual de la base de datos al portapapeles o a un archivo *Metafile*. Esta facilidad es realmente práctica si tene-

mos que generar documentación sobre la estructura de una base de datos.

También dispone de un generador de consultas guiado que resulta bastante práctico y que aprovecha la información de las tablas que hayamos seleccionado. Hay que hacer notar que *Watcom* exige que se haga un *bind* a cada base de datos que queramos utilizar con *VX-REXX*. Esto no es necesario con *VisPro/REXX*.

La manera como se integra una base de datos con la aplicación a desarrollar, también difiere. *Watcom* obliga

### Listado 1.- Fragmento de una DDL Macro generada por el VisPro Database Designer

```

/* register the rexx functions */

rc = rxfuncadd( 'SQLDBS', 'SQLAR', 'SQLDBS' );
rc = rxfuncadd( 'SQLEXEC', 'SQLAR', 'SQLEXEC' );

call sqlpbs 'CREATE DATABASE EJEMPLO'
if SQLCA.SQLCODE <> 0 then do
  say SQLCA.SQLCODE SQLCA.SQMSG
  call sqlpbs 'STOP USING DATABASE'
  say 'Press Enter to continue'
  pull
  exit 1
end
call sqlpbs 'START USING DATABASE EJEMPLO'
if SQLCA.SQLCODE <> 0 then do
  say SQLCA.SQLCODE SQLCA.SQMSG
  call sqlpbs 'STOP USING DATABASE'
  say 'Press Enter to continue'
  pull
  exit 1
end
prep_string = 'DROP TABLE IDUSUAR.PERSONAL'

call sqlexec 'EXECUTE IMMEDIATE :prep_string'
prep_string = 'CREATE TABLE IDUSUAR.PERSONAL'
( CODIGO          INTeGer NOT NULL, '
  NOMBRE          VArChAR(50) NOT NULL, '
  DEPARTA        INTeGer NOT NULL, '
  CARGO           INTeGer NOT NULL, '
  PRIMARY KEY (CODIGO) '
)

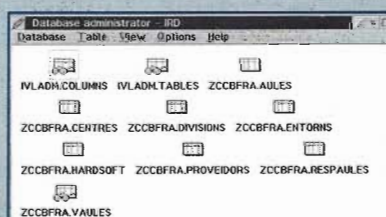
call sqlexec 'EXECUTE IMMEDIATE :prep_string'

[... (
prep_string = 'ALTER TABLE IDUSUAR.PERSONAL'
FOREIGN KEY (DEPARTA) '
REFERENCES IDUSUAR.DEPARTA '
FOREIGN KEY (CARGO) '
REFERENCES IDUSUAR.CARGOS '

call sqlexec 'EXECUTE IMMEDIATE :prep_string'

if SQLCA.SQLCODE <> 0 then do
  say SQLCA.SQLCODE SQLCA.SQMSG
  call sqlpbs 'STOP USING DATABASE'
  say 'Press Enter to continue'
  pull
  exit 1
end

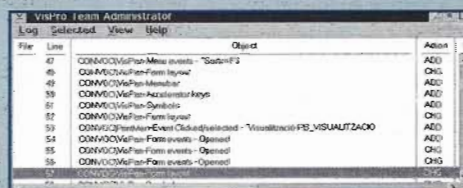
```



**Figura N** VX-REXX Database Administrator: Lista de tablas de una base de datos



**Figura O** VX-REXX Database Administrator: Características de una tabla



**Figura P** Aspecto del Team Administrator



a incluir dos objetos en la caja de diálogo. Uno de conexión y otro de interrogación. A partir del de interrogación, en el que habremos especificado nuestra consulta, se nos generan los objetos del panel que ha de acceder a la base de datos. *VisPro/REXX* opta, una vez más, por una solución visual. Arrastrando una o más tablas desde el *Database Designer* hasta una *form* (panel, ventana, caja de diálogo), se nos generan los objetos necesarios para el acceso a las tablas y el código *REXX* correspondiente.

Hay también dos utilidades interesantes que nos ofrece *VisPro/REXX* y que no encontramos en *VX-REXX*: El *Help Editor* y el *Team Administrator*. El *Help Editor*, como su nombre indica, es editor de archivos *IPF*. Es bastante completo y permite compilar los archivos en formato *HLP* y en formato *INF*.

El *Team Administrator* permite a los desarrolladores comparar los niveles de cambio y las versiones de un proyecto de manera sencilla y clara. Permite deshacer un cambio y monitorizar los progresos del desarrollo. Todos los cambios se almacenan automáticamente y las *forms* pueden ser bloqueadas para asegurarse de que sólo un programador las modifica a la vez. También incluye la posibilidad de ver una *form* en modalidad de "sólo lectura" y la creación de sombras de *forms* para facilitar el acceso en un entorno de sistema distribuido.

*VisPro/REXX* incorpora en su versión 3.0 el nuevo concepto de *Sombras de Forms*. Las sombras de *forms* son similares a las sombras de *WPS*. Te permiten incluir una *form* de *VisPro/REXX* en otra carpeta de proyecto. La *form* original puede estar local en tu máquina o en un servidor remoto. Esto permite compartir *forms* de uso común en diversos proyectos.

Las sombras de *forms* se crean de la misma manera que las de *WPS*. Podemos arrastrar una *form* a una nueva carpeta de proyecto manteniendo pulsadas las teclas [Ctrl]+[Mayús] o seleccionar *Create*

*Shadow* del menú emergente de un icono de *form*.

Las sombras de *forms* difieren de las de *WPS* en algo fundamental. Mientras las de *WPS* no tienen representación en el sistema de archivos por almacenarse en el archivo *OS2.INI*, las de *VisPro/REXX* son directorios en un proyecto. Un archivo llamado *!!LINK* existe en el directorio de la *form* original. Esto previene la pérdida de la estructura del proyecto debido a cambios en los archivos de sistema de *OS/2* y mejora la fiabilidad sobre la implementación de las sombras en *OS/2*.


Las sombras de *forms* se crean por defecto como objetos de sólo lectura. Para cambiar este atributo, hay que seleccionar *Read only* en el menú emergente de la *form*.

Más arriba hemos hablado de dos grandes diferencias entre *VisPro/*

*REXX* y *VX-REXX*: La interfaz de usuario y la generación de ejecutables.

Mi opinión es que la interfaz de usuario es más coherente y más sencilla en el caso de *VisPro/REXX*. Mientras *VX-REXX* no nos muestra como objetos aparte cada ventana, *VisPro/REXX* nos presenta cada *form* como un objeto en la carpeta de proyectos. La sintaxis de *VisPro/REXX* también me parece más clara.

En cuanto a la generación de ejecutables, *VX-REXX* exige la presencia de una *DLL runtime* de unas 900 KB. Con *VisPro/REXX*, podemos optar por entregar al cliente un solo ejecutable o un ejecutable y una *DLL runtime* (en este caso la *DLL* ocupa 191 KB).

Por todo ello, mi opción ha sido la de utilizar *VisPro/REXX*, sin dejar de estar atento a un producto realmente magnífico como *VX-REXX*. 

## Listado 2.- Fragmento de una DDL generada por el VisPro Database Designer

```

DROP TABLE PERSONAL

CREATE TABLE IDUSUAR.PERSONAL
(CODIGO                INTeGer NOT NULL,
 NOMBRE                VARCHAR(50) NOT NULL,
 DEPARTA              INTeGer NOT NULL,
 CARGO                 INTeGer NOT NULL,
 PRIMARY KEY (CODIGO),
 FOREIGN KEY (DEPARTA)
 REFERENCES DEPARTA,
 FOREIGN KEY (CARGO)
 REFERENCES CARGOS)

CREATE INDEX X1PERSONAL ON IDUSUAR.PERSONAL (NOMBRE ASC)

DROP TABLE DEPARTA

CREATE TABLE IDUSUAR.DEPARTA
(CODIGO                INTeGer NOT NULL,
 NOMBRE                VARCHAR(100) NOT NULL,
 DIRECTOR              INTeGer NOT NULL,
 LOCALIDAD            VARCHAR(50) NOT NULL,
 PRIMARY KEY (CODIGO),
 FOREIGN KEY (DIRECTOR)
 REFERENCES PERSONAL)

CREATE INDEX X1DEPARTA ON IDUSUAR.DEPARTA (NOMBRE ASC)

CREATE INDEX X2DEPARTA ON IDUSUAR.DEPARTA (LOCALIDAD ASC)

DROP TABLE CARGOS

CREATE TABLE IDUSUAR.CARGOS
(CODIGO                INTeGer NOT NULL,
 CARGO                 VARCHAR(100) NOT NULL,
 PRIMARY KEY (CODIGO))

CREATE INDEX X1CARGOS ON IDUSUAR.CARGOS (CARGO ASC)

```