El editor de informes Report Manager (1)

Pese a que no es un generador de informes muy conocido en el mundo Delphi, este editor de informes destaca por su sencillez (está en español) y que sobre todo es gratuito.

Si en algo aventaja a QuickReport es en que permite generar los informes en archivos con extensión REP para luego ser cargados desde Delphi e incluso podemos modificarlos en tiempo de ejecución. Tampoco da las explosiones que ocasiona el editor Rave cuando te equivocas, aunque hay que reconocer que es un poco raro en algunas cosas.

DESCARGANDO EL EDITOR

Puede descargarse de esta página web:

http://reportman.sourceforge.net/indexes.html

Dentro del apartado **Descarga** nos llevará a otra página de SourceForge donde se encuentra la versión 2.9a

Lo que ocurre es que solo nos bajamos el editor y también necesitamos los componentes. Para ello pulsamos en la misma página web el botón **View all files** y descargamos este archivo **repporman_delphi_builder_2_9.zip**:

INSTALANDO EL EDITOR

La ventana de instalación permite elegir el idioma español y solo hay que ir pulsando el botón **Siguiente**:

INSTALANDO LOS COMPONENTES

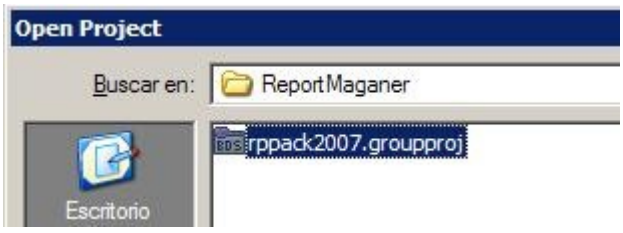
Creamos una carpeta llamada **ReportManager** dentro de la carpeta:

\CodeGear\RAD Studio\5.0\Componentes

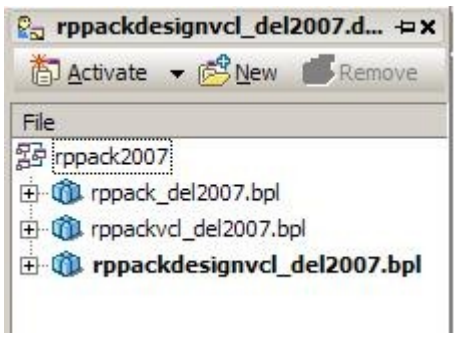
Y descomprimimos dentro de la misma el archivo que hemos descargado:

Repporman_delphi_builder_2_9.zip

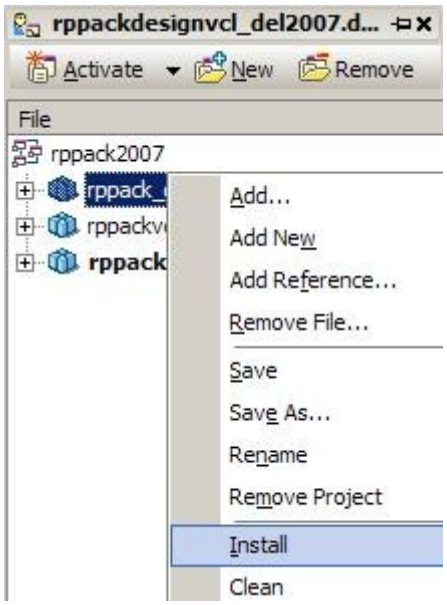
Nos vamos a Delphi y abrimos el archivo **rppack2007.groupproj**:



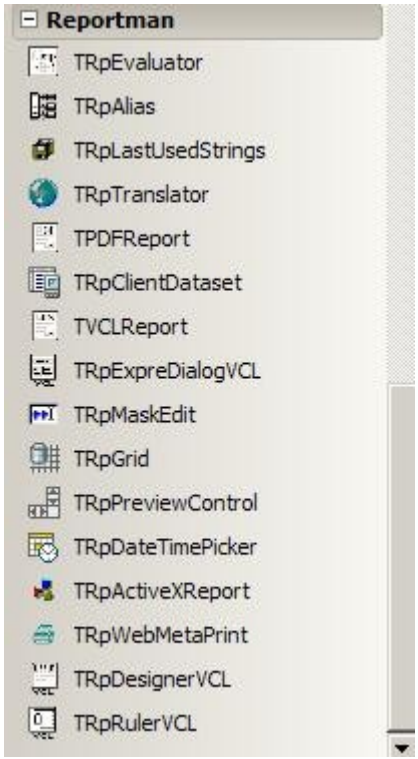
Veremos estos paquetes en la parte derecha de Delphi:



Pulsamos el paquete **rppack_del2007.bpl** con el botón derecho del ratón y seleccionamos **Install**:



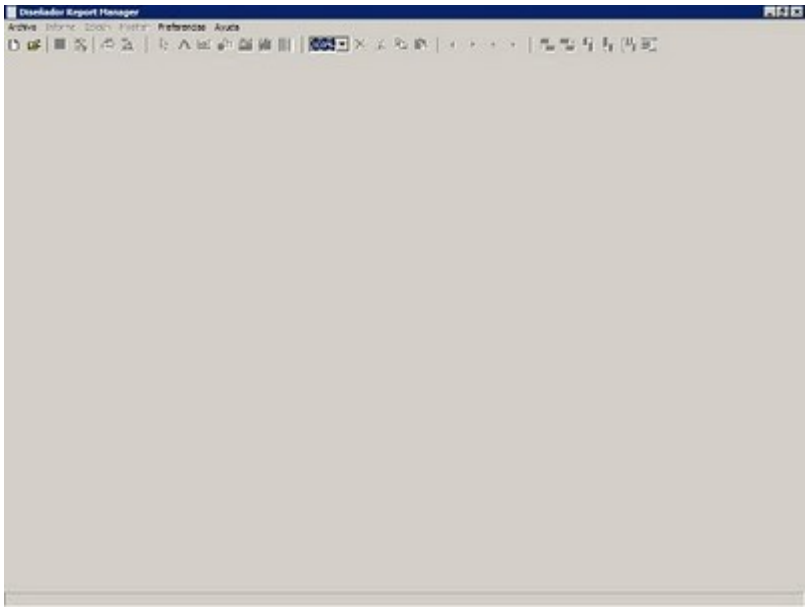
Con solo hacer esto se instalará todo. Si todo ha ido correctamente veremos esto en la paleta de componentes:



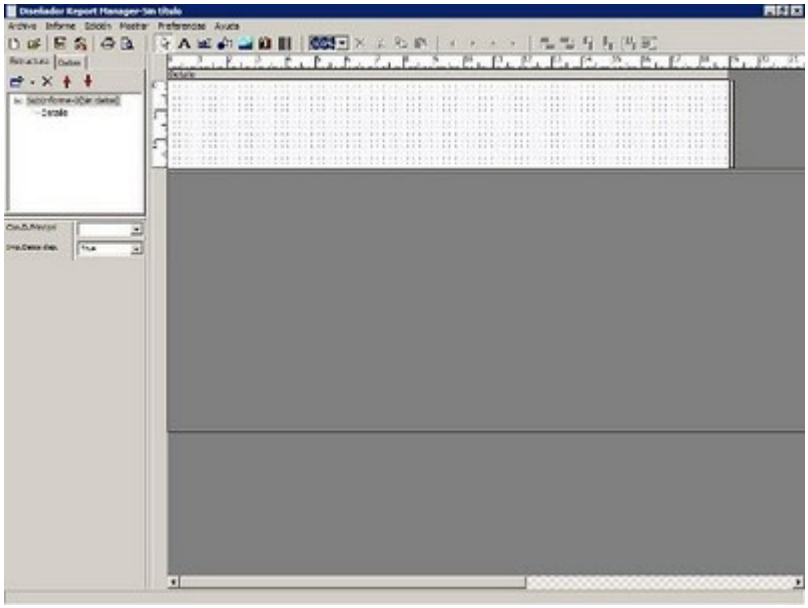
Pasemos a ver el entorno de trabajo de este editor.

EL ASPECTO DEL EDITOR

Al ejecutar por primera vez el editor de informes vemos que aparece completamente pelado:



Pulsamos el botón **Nuevo** y mostrará este aspecto:

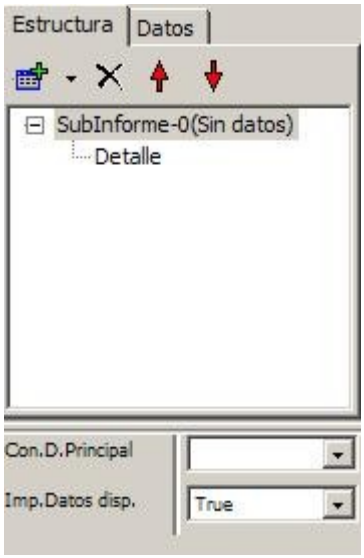


Lo mejor de todo es que se encuentra en nuestro idioma. Dentro del editor podemos distinguir estas partes:

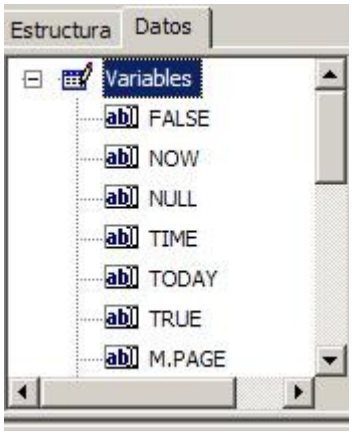
Barra de herramientas superior

Contiene las típicas opciones de abrir o guardar archivos, mostrar la vista previa, etc. Y abajo tenemos los iconos para insertar componentes: etiquetas, campos vinculados a tablas, formas geométricas, dibujos y códigos de barras. Ya iremos viendo detenidamente cada uno de ellos según los necesitemos.

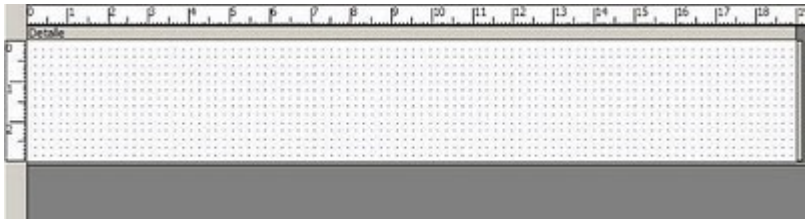
Estructura del informe



Aquí vemos las bandas de las que se compone el informe. Si seleccionamos por ejemplo el **Detalle** mostrará abajo el inspector de objetos con sus propiedades. A su lado tenemos la pestaña **Datos** donde veremos las variables globales que podemos utilizar:



La zona de trabajo



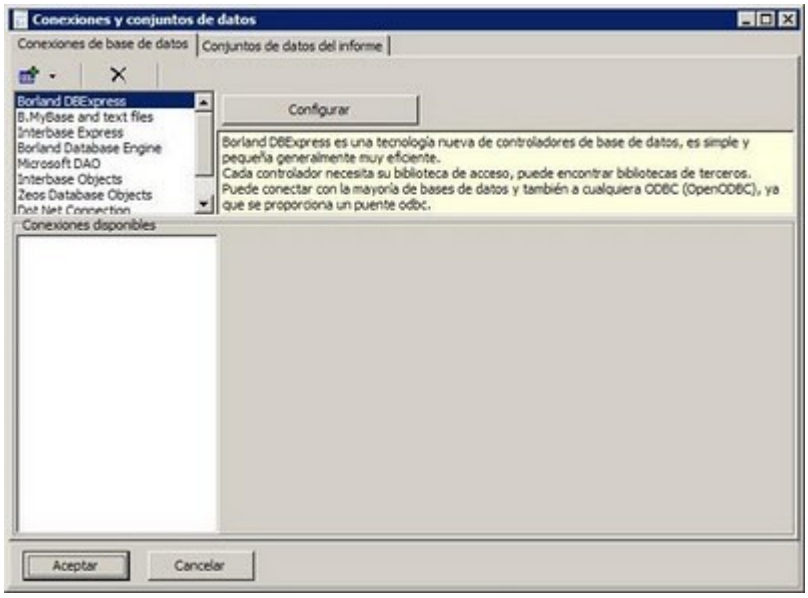
Aquí es donde iremos colocando componentes para crear el informe. No os dejéis engañar por la simpleza de este editor, ya que como veremos más adelante se pueden crear informes avanzados por programación interceptando en tiempo real los componentes al igual que hacemos con QuickReport.

CREANDO UN INFORME

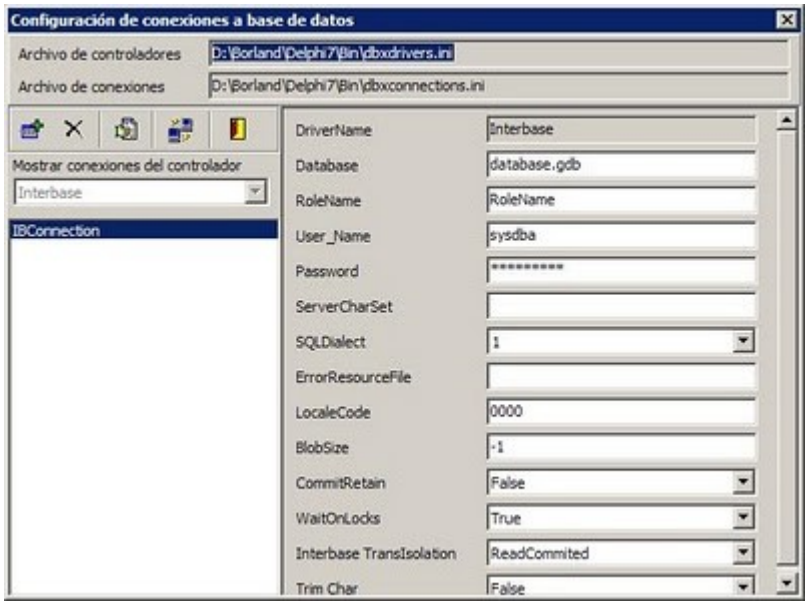
Para probar que todo ha ido bien, vamos a crear un pequeño informe a partir de una base de datos de Interbase que contiene una tabla de CONTACTOS. Lo primero que tenemos que hacer es establecer la fuente de donde van a venir los datos. La vinculación con el origen de datos es otra característica algo rara que confunde por sus múltiples pantallas.

Seleccionamos en el menú superior las opciones **Informe** - **Configuración de datos**:

Y aparecerá esta ventana:

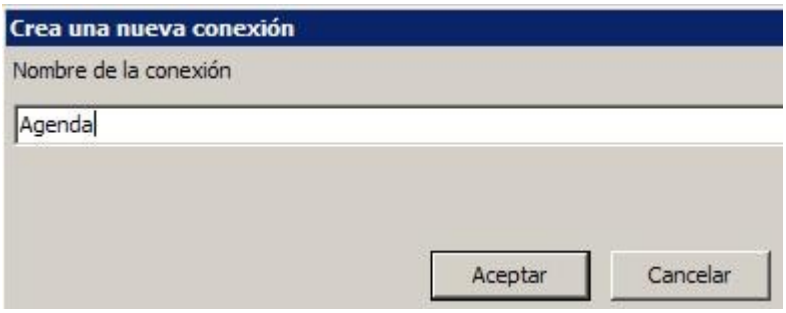


Seleccionamos **Interbase Express** que equivale realmente a una conexión DBExpress al pulsar el botón **Configurar**:

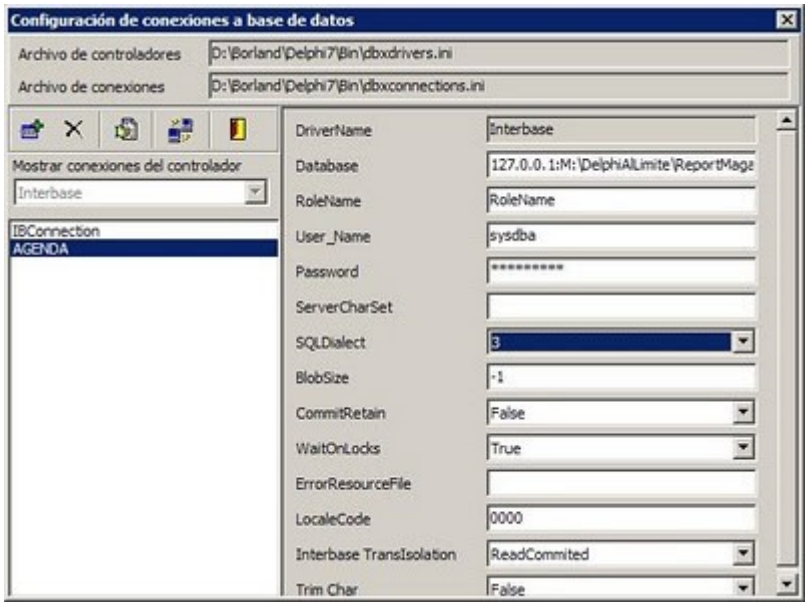


Aquí trae como ejemplo la configuración a una base de datos Interbase cuya conexión DBExpress está configurada dentro de Delphi 7, pero lo que vamos a hacer es crear una nueva conexión pulsando este botón:

En el nombre de la conexión le ponemos **AGENDA**:

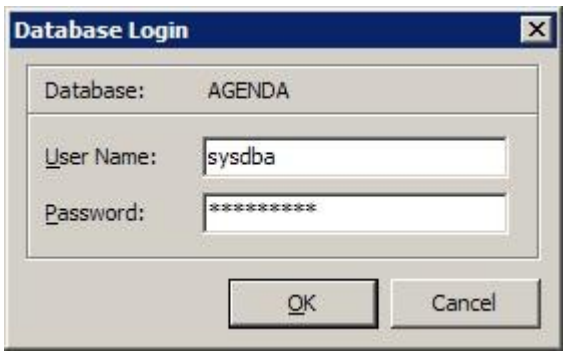


Entonces volverá a la pantalla anterior y configuramos donde se encuentra la base de datos:



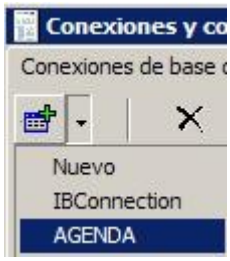
No hay que olvidar poner la propiedad **SQL Dialect** a **3**. Probamos la conexión pulsando el botón:

A continuación nos pedirá identificarnos:

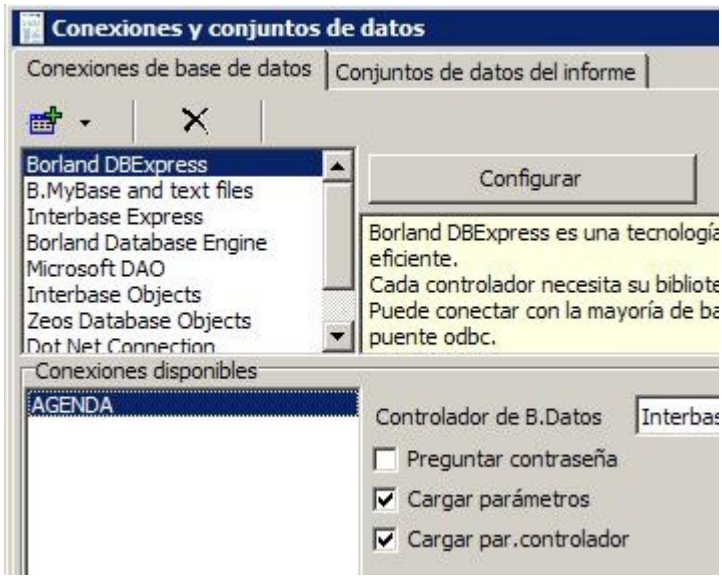


Y si todo ha ido bien nos dirá esto:

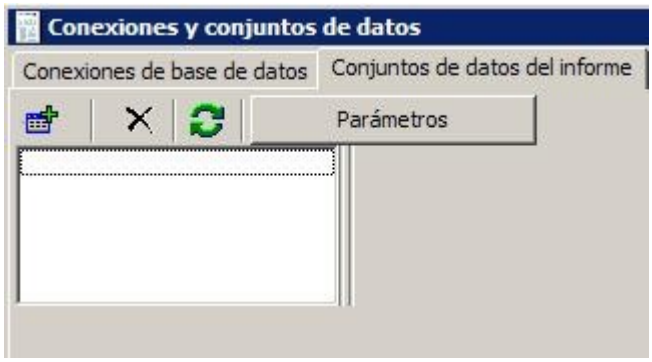
Volvemos a la primera pantalla y seleccionamos la conexión que hemos creado:



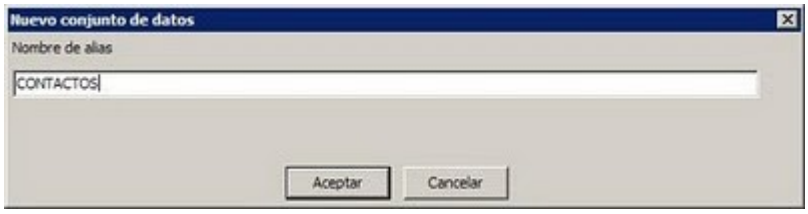
Lo añadirá a la lista de conexiones disponibles:



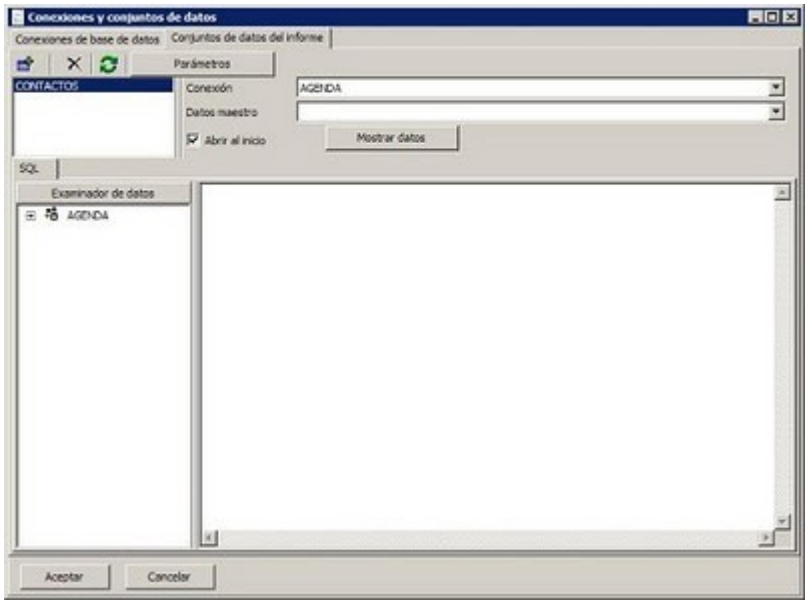
Ahora nos vamos a la pestaña **Conjunto de datos del informe** para elegir las tablas de la que queremos sacar la información:



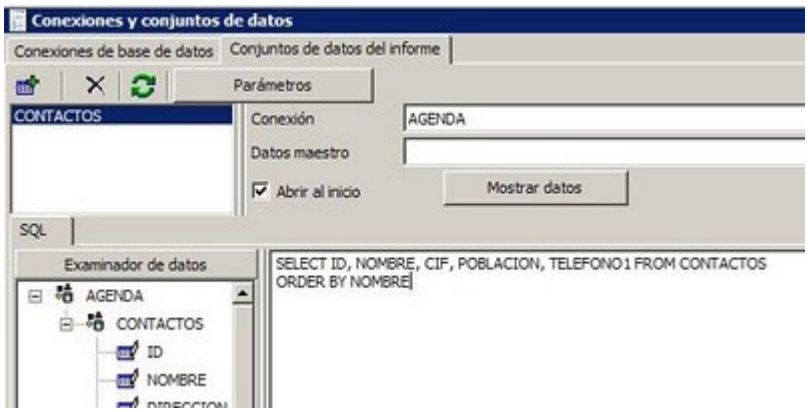
Pulsamos el botón **Nuevo conjunto de datos** y lo llamamos **CONTACTOS**:



La ventana cambiará a esta otra:

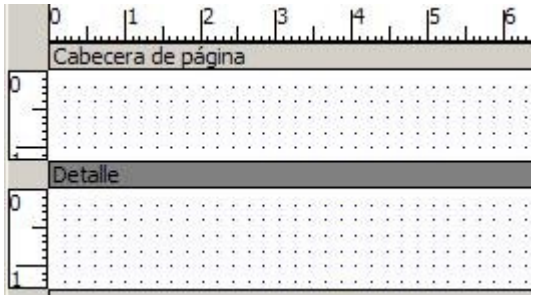


Escribimos la sentencia SQL de los datos que nos vamos a traer:



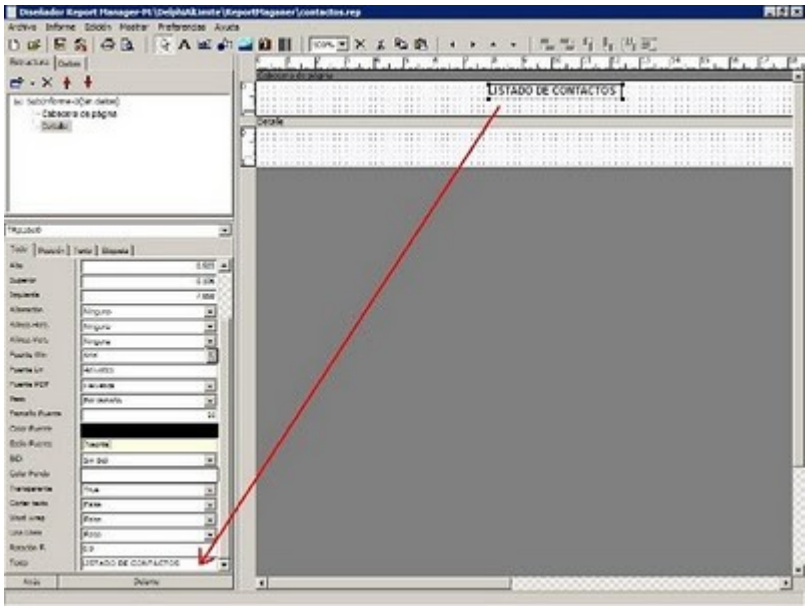
Pulsamos **Aceptar** y al volver al editor veremos en la pestaña **Datos** la tabla que hemos vinculado:

Para no perder la configuración guardamos el informe con **Archivo** – **Guardar como** – **Contactos.rep**. Como lo que quiero diseñar es un listado, tenemos el problema de que solo tenemos la banda de detalle. Para insertar la banda de la cabecera seleccionamos **Informe** – **Añadir** – **Cabecera de página**. Ya tenemos las dos bandas:



Vamos a pulsar este icono para añadir una etiqueta al informe:

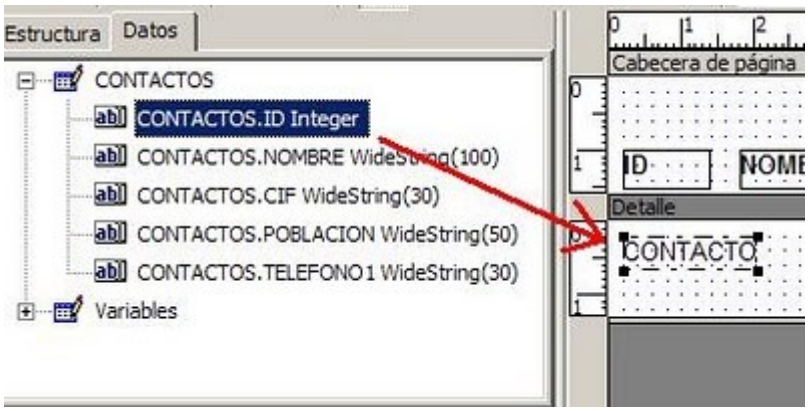
Una vez tenemos la etiqueta seleccionada nos vamos a la izquierda al inspector de objetos y en la propiedad texto escribimos LISTADO DE CONTACTOS:



Una de las cosas raras que hace este editor es que al insertar una etiqueta le da un anchura y altura muy pequeña. Tenemos que estirarla siempre. Así creamos el resto de etiquetas:

Para copiar etiquetas no podemos utilizar la combinación de teclas CTRL + C y CTRL + V, hay que hacerlo con ALT + C y ALT + V. Esta es también una característica algo rara.

Ahora insertamos los campos arrastrándolos desde la pestaña de datos a la banda del detalle:

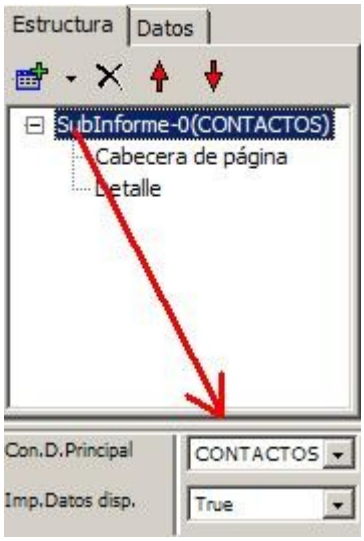


Así quedaría después de insertarlos:

Pulsamos el botón de vista previa para ver el informe:

Aquí encontramos el primer inconveniente:

Para que se listen todos los registros tenemos que vincular la tabla CONTACTOS al informe. Para ello nos vamos a la pestaña **Estructura** y en la propiedad **Con. D. Principal** seleccionamos CONTACTOS:



Ahora si sale bien:

En el próximo artículo veremos como cargar este informe desde nuestro proyecto de Delphi así como exportarlo a PDF y otros formatos. También veremos otros formatos más avanzados como una factura.

El editor de informes Report Manager (2)

Continuando con este discreto diseñador de informes, vamos a ver como abrir el informe desde nuestro proyecto en Delphi, haciendo que muestre la vista previa o exportándolo a otros formatos como PDF, TXT o HTML. También veremos como crear una factura que recoge datos de varias tablas.

ABRIENDO EL INFORME DESDE DELPHI

Continuando con el listado de contactos que creamos en el artículo anterior, vamos a crear un nuevo proyecto en Delphi con este formulario:

Para poder realizar las pruebas necesitamos añadir los componentes **VCLReport** y **PDFReport** que se encuentran en el apartado **Reportman** de la paleta de componentes.

Al pulsar el botón **Imprimir** podemos realizar una vista previa si está activado el checkbox o bien lanzarlo directamente a la impresora:

```
procedure TFContactos.BImprimirClick(Sender: TObject);
begin
  VCLReport.Title := 'Listado de contactos';
  VCLReport.Filename := ExtractFilePath(Application.ExeName) +
    'contactos.rep';
  VCLReport.Preview := VistaPrevia.Checked;
  VCLReport.Execute;
end;
```

Se asume que la ruta del informe **contactos.rep** se encuentra en la misma carpeta que el ejecutable. Si no es así, modificar la propiedad **Filename** como corresponda (lo mejor es configuralo en un archivo INI).

Para exportar el documento a PDF podemos hacerlo de dos formas. O bien utilizamos el componente **PDFReport** (recomendado):

```
procedure TFContactos.BExportarPDFClick(Sender: TObject);
begin
  // Método 1
  PDFReport.Title := 'Listado de contactos';
  PDFReport.Filename := ExtractFilePath(Application.ExeName) +
    'contactos.rep';
  PDFReport.PDFFilename := ExtractFilePath(Application.ExeName) +
    'contactos.pdf';
  PDFReport.Execute;
  Application.MessageBox(PChar('Listado exportado correctamente:' +
    #13 + #13 + PDFReport.PDFFilename), 'Exportando', MB_ICONINFORMATION);
end;
```

O bien el componente **VCLReport**:

```
procedure TFContactos.BExportarPDFClick(Sender: TObject);
begin
  // Método 2
  VCLReport.Title := 'Listado de contactos';
  VCLReport.Filename := ExtractFilePath(Application.ExeName) +
    'contactos.rep';
  VCLReport.SaveToPDF(ExtractFilePath(Application.ExeName) +
    'contactos.pdf');
  Application.MessageBox(PChar('Listado exportado correctamente:' +
    #13 + #13 + ExtractFilePath(Application.ExeName) + 'contactos.pdf'),
    'Exportando', MB_ICONINFORMATION);
end;
```

Si el autor ha creado el componente **PDFReport** sus razones tendrá. También podemos exportarlo a una página web HTML de este modo:

```
procedure TFContactos.BExportarHTMLClick(Sender: TObject);
begin
  VCLReport.Title := 'Listado de contactos';
  VCLReport.Filename := ExtractFilePath(Application.ExeName) +
    'contactos.rep';
  VCLReport.SaveToHTML(ExtractFilePath(Application.ExeName) +
    'contactos.html');
  Application.MessageBox(PChar('Listado exportado correctamente.' +
    #13 + #13 + ExtractFilePath(Application.ExeName) +
    'contactos.html'), 'Exportando', MB_ICONINFORMATION);
end;
```

La exportación que hace es bastante decente:

Aunque EurekaLog me ha detectado un fallo en un procedimiento de Report Manager que no libera memoria:

```
{$IFNDEF FORWEBAX}
procedure ExportReportToHtml(report:TRpBaseReport;
var
 pdfdriver:TRpPDFDriver;
 apdfdriver:TRpPrintDriver;
 oldprogres:TRpProgressEvent;
 astream:TMemoryStream;
 oldtwopass:boolean;
begin
 pdfdriver:=TRpPDFDriver.Create;
 pdfdriver.compressed:=true;
 astream:=TMemoryStream.Create;
 try
  pdfdriver.DestStream:=aStream;
  apdfdriver:=pdfdriver;
```

Por lo visto no libera el objeto **pdfdriver**.

¿Habrá que hacer algo más?

Exportar el archivo a texto es algo similar:

```
procedure TFContactos.BExportarTXTClick(Sender: TObject);
begin
  VCLReport.Title := 'Listado de contactos';
  VCLReport.Filename := ExtractFilePath(Application.ExeName) +
    'contactos.rep';
  VCLReport.SaveToText(ExtractFilePath(Application.ExeName) +
    'contactos.txt');
  Application.MessageBox(PChar('Listado exportado correctamente.' +
    #13 + #13 + ExtractFilePath(Application.ExeName) +
    'contactos.txt'), 'Exportando', MB_ICONINFORMATION);
end;
```

Hay que reconocer que aquí no puede hacer maravillas:

También le ocurre que no libera este objeto:

```
function PrintReportToText(report:TRpReport;Caption:str
     allpages:boolean;frompage,topage,copies:integer;
     filename:string;collate:Boolean;oemconvert:boolean
var
 TextDriver:TRpTextDriver;
 aTextDriver:TRpPrintDriver;
 oldprogres:TRpProgressEvent;
begin
 if Length(Trim(filename))<0 then
  Raise Exception.Create(SRpNoFileNameProvided+':TXT');
 TextDriver:=TRpTextDriver.Create;
 TextDriver.ForceDriverName:=Trim(forcedrivername);
```

Aun así, ambas exportaciones funcionan bastante bien. Vamos con algo más difícil.

CREANDO EL INFORME DE UNA FACTURA

Supongamos que tenemos una base de datos de Interbase llamada **facturación.gdb** que tiene la tabla de CLIENTES (clic para ampliar):

ID	NUMERO	NOMBRE	DIRECCION	POBLACION	CP	PROVINCIA	TELEFONO	CIF
1	1.423	PABLO MARTINEZ GONZALEZ	C/ ASTURIAS, 8	MADRID	28080	MADRID	555125689	66888559K
2	5.674	MARIA PEREZ BLANCO	AVDA. EUROPA, 8	ALICANTE	03524	ALICANTE	555447858	11222333G

También tenemos la tabla de ARTICULOS:

ID	NUMERO	NOMBRE	PRECIO
1	450.324	TECLADO GENIUS	12,990
2	785.463	MONITOR BENQ	78,450
3	458.685	IMPRESORA MULTIFUNCION HP	140,990
4	695.874	PORTATIL SONY VIAO	1.099,000

Y para la cabecera de la factura esta la tabla FACTURAS:

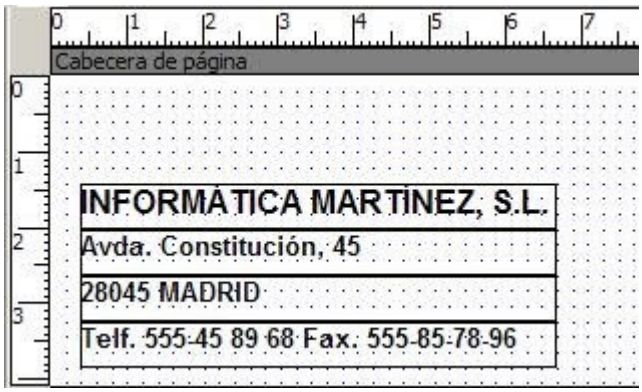
ID	NUMERO	IDCLIENTE	BASEIMPONIBLE	IVA	TOTAL
1	456	1	1.331,430	213,029	1.544,459

Y luego esta la tabla DETALLE para su contenido:

ID	IDFACTURA	IDARTICULO	PRECIO	PORIVA	IVA	TOTALLINEA
1	1	1	12,990	16,000	2,078	15,068
2	1	2	78,450	16,000	12,552	91,002
3	1	3	140,990	16,000	22,558	163,548
4	1	4	1.099,000	16,000	175,840	1.274,840

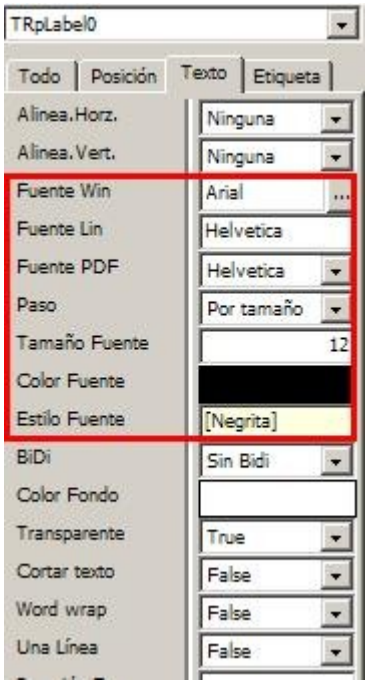
Nos vamos al editor **Report Manager**, creamos un nuevo informe y lo guardamos con el nombre **factura.rep**. Como solo tenemos la banda de detalle, tenemos que seleccionar **Informe** – **Añadir** – **Cabecera de página** e **Informe** – **Añadir** – **Pie de página**.

Vamos a comenzar por la cabecera. En la parte izquierda de la misma vamos metiendo etiquetas (**Insertar texto estático**) para poner los datos de nuestra empresa:



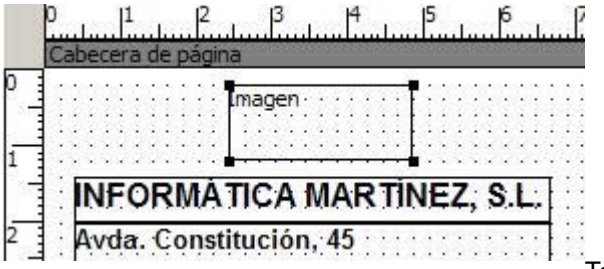
Para poder escribir el texto en una etiqueta tenemos que rellenar la propiedad **Texto** del inspector de objetos:

Y para seleccionar el tamaño de la fuente y el estilo (negrita, cursiva, subrayado) utilizamos las propiedades que se encuentran en la pestaña **Texto**:

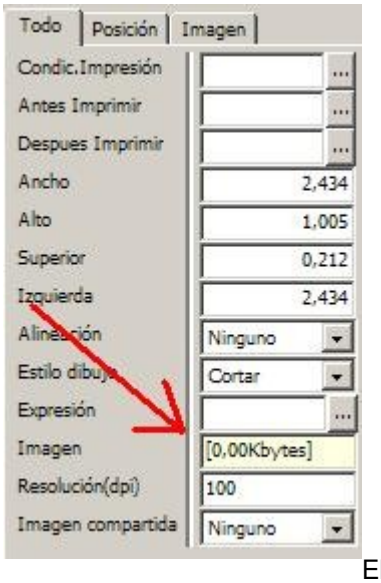


Al final te acostumbras, pero no hubiese venido mal una barra de estilos de fuente al estilo Microsoft. Ahora vamos a insertar el logotipo de la empresa utilizando el botón:


Es conveniente que el logotipo se encuentre la misma carpeta que el archivo **factura.rep**. Para insertar el logotipo le damos a ese botón y abrimos un rectángulo en el informe:



Teniendo la imagen seleccionada, nos vamos al inspector de objetos y hacemos clic sobre la propiedad **Imagen**:



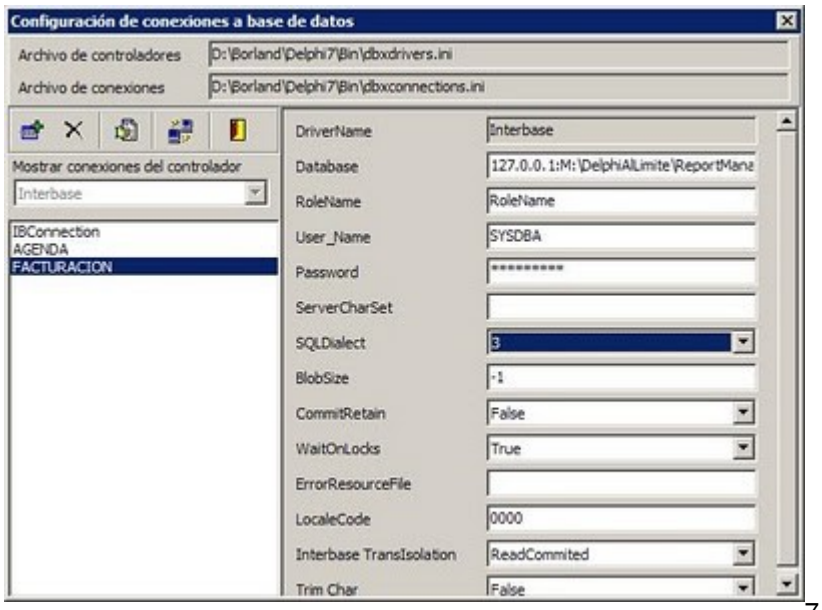
Elegimos la imagen y luego la ajustamos un poco:

CONFIGURANDO EL ORIGEN DE DATOS

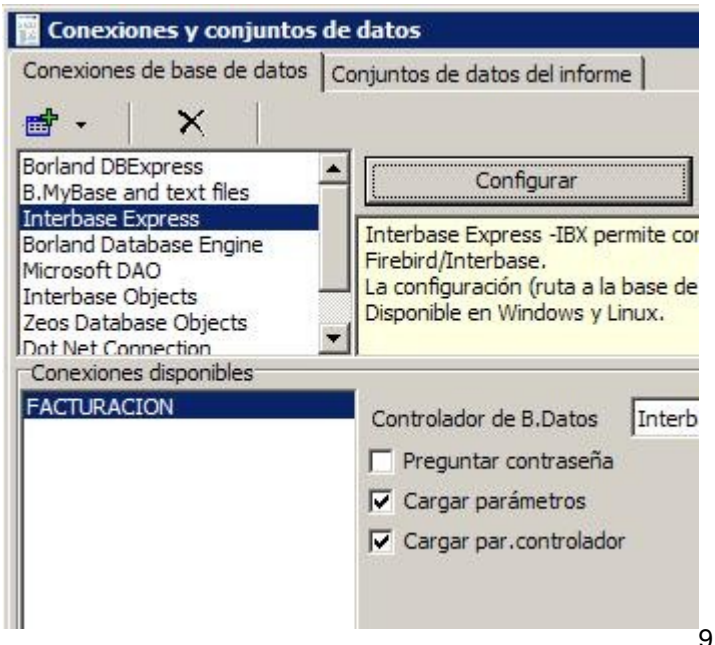
Ahora vamos a comenzar a enlazar datos de tablas. Lo que necesitamos es seleccionar la base de datos y crear las consultas de cabecera y detalle de la factura.

Los pasos para traernos la información son los siguientes:

1. Seleccionamos **Informe** – **Configuración de datos**.
2. Seleccionamos **Interbase Express**.
3. Pulsamos el botón **Configurar**.
4. Ahora pulsamos el botón **Añade una conexión…**
5. Nombre de la conexión: **FACTURACION**.
6. Configuramos la base de datos como hicimos en el artículo anterior:



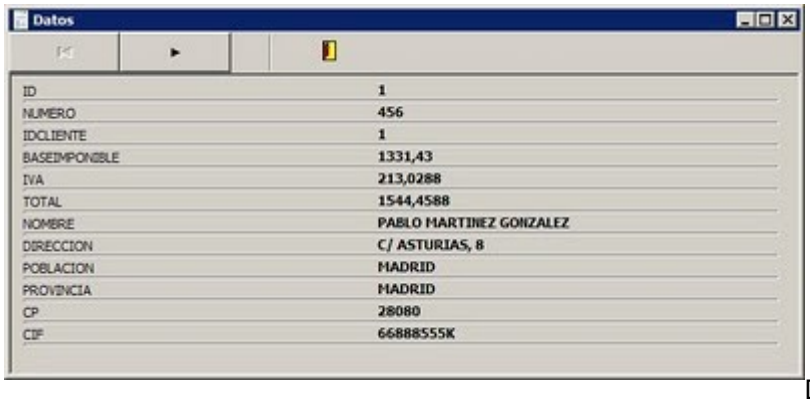
7. Pulsamos el botón **Activar** la conexión seleccionada y comprobamos si conecta correctamente.
8. Una vez establecida la conexión, cerramos esta ventana y añadimos la conexión recien creada en la ventana donde estamos:



9. Nos vamos a la pestaña **Configuración de datos del informe** y pulsamos el botón **Nuevo conjunto de datos**.
10. Nombre del alias: **FACTURAS**.
11. Le ponemos de SQL:

```
SELECT FACTURAS.*,CLIENTES.NOMBRE AS NOMBRE, CLIENTES.DIRECCION AS DIRECCION,
CLIENTES.POBLACION AS POBLACION, CLIENTES.PROVINCIA AS PROVINCIA,
CLIENTES.CP,CLIENTES.CIF FROM FACTURAS
LEFT JOIN CLIENTES ON FACTURAS.IDCLIENTE=CLIENTES.ID
WHERE ID=1
```

Si pulsamos el botón **Mostrar datos** nos mostrará la primera factura (solo para ver que la SQL es correcta):



Después de terminar las pruebas hay que quitarle la condición WHERE.

12. Igualmente vamos a añadir la tabla DETALLE:

```
SELECT DETALLE.*,ARTICULOS.NUMERO,
ARTICULOS.NOMBRE FROM DETALLE
LEFT JOIN ARTICULOS ON ARTICULOS.ID=DETALLE.IDARTICULO
```

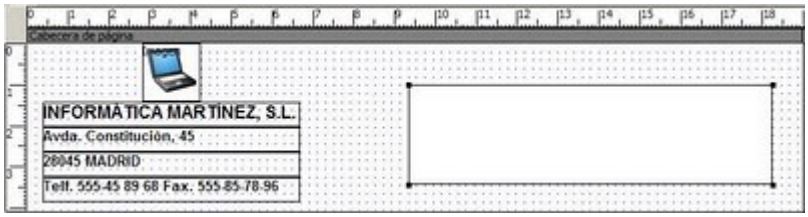
Las tablas de clientes y de artículos no necesitamos traerlas ya que las hemos vinculado con LEFT JOIN a las consultas de FACTURAS y DETALLE.

13. Pulsamos el botón **Aceptar** y ya podemos volver al informe para añadir campos.

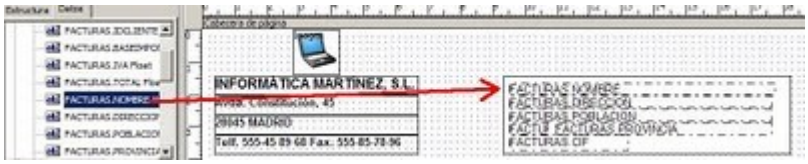
MOSTRANDO LOS DATOS DEL CLIENTE

Para enmarcar los datos del cliente añadimos un dibujo simple con el botón:

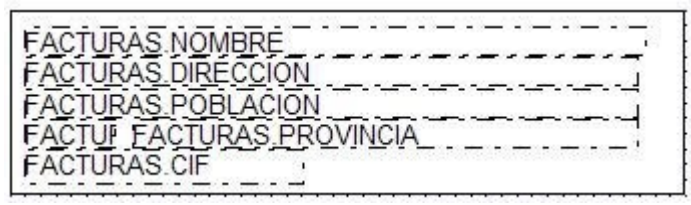
Lo colocamos más o menos así:



Ahora vamos a traernos el nombre del cliente arrastrando el campo NOMBRE desde la pestaña **Datos**:



Así nos traemos el resto de datos:



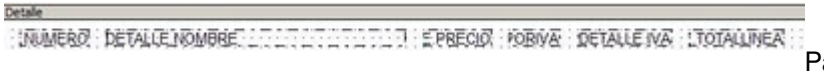
Si en cualquier momento os equivocáis de campo o componente y queréis eliminarlo, hay que seleccionar el campo y pulsar CTRL + SUPR.

Utilizando una etiqueta y un campo, vamos a poner debajo del logo el nº de factura y las cabeceras del detalle:

Os recuerdo que para copiar campos hay que utilizar la combinación de teclas ALT + C y ALT + V.

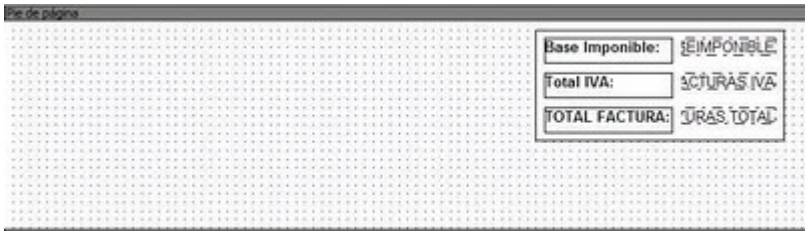
CREANDO EL DETALLE Y EL PIE DE LA FACTURA

El detalle de la factura ya tiene todos sus campos en la tabla DETALLE porque hemos vinculado con LEFT JOIN el número y nombre del artículo que nos hacía falta:



Para que aparezcan todas las líneas de detalle debemos vincular todo el informe a la tabla de detalle. Esto se hacía seleccionando la tabla DETALLE al informe:

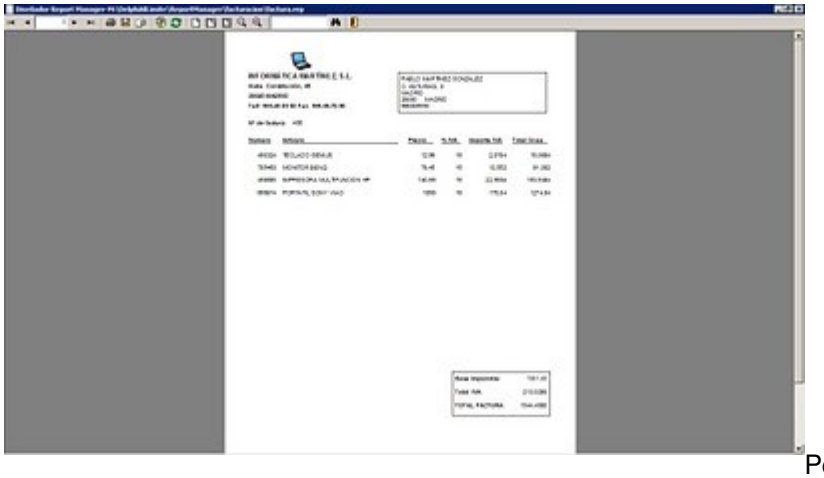
El pie es todavía más sencillo. Solo hay que traerse la base imponible, el importe del IVA y el total de la factura:



Antes de terminar, necesitamos filtrar el detalle para que no salga el detalle de todas las facturas, sólo debe salir el detalle de la factura que estamos imprimiendo. Para ello seleccionamos la banda del detalle y en su propiedad **Condic. Impresión** escribimos:

DETALLE.IDFACTURA=FACTURAS.ID

Si mostramos la vista previa de la factura debe salir esto:



Pero como todos sabemos, siempre nos obligan a añadir cosas raras en la factura como imágenes en tiempo real, mostrar datos de campos que no existen o colorear la fuente según ciertas condiciones. Eso lo veremos en el siguiente artículo.

Pruebas realizadas en RAD Studio 2007.

18 junio 2010

El editor de informes Report Manager (3)

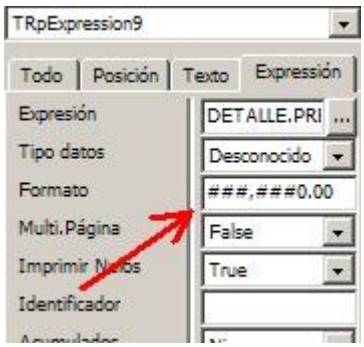
Continuando con la factura del artículo anterior, vamos a darle formato a los números reales y a modificar su comportamiento en tiempo real desde Delphi. También veremos como enviar parámetros SQL desde Delphi al informe.

AJUSTAR LOS DECIMALES

Antes de cargar el informe en nuestro proyecto de Delphi vamos a tratar de adecentarlo un poco comenzando por el detalle del documento. Lo primero que apreciamos es que el número de decimales sale como le da la gana:

Precio	% IVA	Importe IVA	Total línea
12,99	16	2,0784	15,0684
78,45	16	12,552	91,002
140,99	16	22,5584	163,5484
1099	16	175,84	1274,84

Para dar formato a dos decimales seleccionamos los campos PRECIO, PORIVA, IVA y TOTALLINEA y en su propiedad **Formato** le ponemos **###,##0.00**:



Ahora si sale como nosotros queremos:

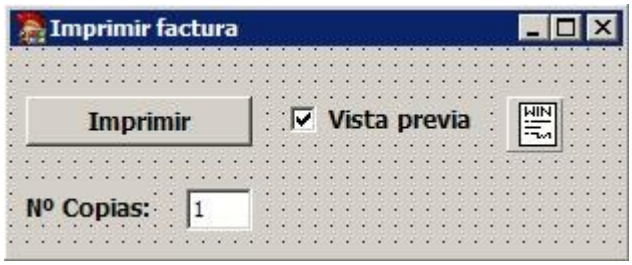
Precio	% IVA	Importe IVA	Total línea
12,99	16,00	2,08	15,07
78,45	16,00	12,55	91,00
140,99	16,00	22,56	163,55
1.099,00	16,00	175,84	1.274,84

Lo mismo tenemos que hacer con los totales del documento.

MODIFICAR EL INFORME EN TIEMPO REAL

Una de las cosas que siempre me ha gustado de QuickReport es tener la posibilidad de modificar en tiempo real los componentes del informe justo antes de enviarlo a la impresora.

Vamos a verlo con un nuevo proyecto en Delphi con el formulario que va a imprimir la factura:



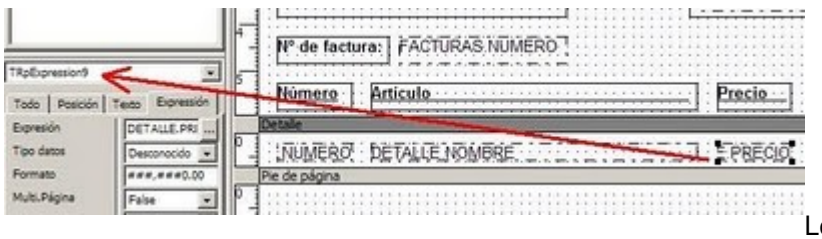
Al igual que el ejemplo de los contactos tenemos el componente **VCLReport** que configuramos al pulsar el botón **Imprimir**:

```
procedure TFImprimirFactura.BImprimirClick(Sender: TObject);
begin
  VCLReport.Title := 'Factura';
  VCLReport.Filename := ExtractFilePath(Application.ExeName) +
    'factura.rep';
  VCLReport.Report.Copies := StrToInt(NumCopias.Text);
  VCLReport.Preview := VistaPrevia.Checked;
  VCLReport.Execute;
end;
```

También controlamos el número de copias de que se imprimen según lo que tenemos puesto en el campo **NumCopias**. Una modificación que le vamos a hacer es que imprima en rojo y en negrita aquellos precios superiores a 1.000. Esto lo hacemos en el evento **BeforePrint** (antes de imprimir) del componente **VCLReport**:

```
procedure TFImprimirFactura.VCLReportBeforePrint(Sender: TObject);
var
i: Integer;
begin
  with VCLReport.Report do
  begin
    if FindComponent('TRpExpression9') is TRpExpression then
    begin
      if (FindComponent('TRpExpression9') as TRpExpression).IdenExpression.Value
>
        1000 then
      begin
        (FindComponent('TRpExpression9') as TRpExpression).FontColor := clRed;
        (FindComponent('TRpExpression9') as TRpExpression).FontStyle := 1;
      end;
    end;
  end;
end;
```

Para poder acceder a la clase **TRpExpression** debemos añadir la unidad **rplabelitem** al formulario donde vamos a imprimir. El nombre que busco (**TRpExpression9**) nos lo dice el mismo editor:



Lo que hacemos es interceptarlo justo antes de imprimirlo y según el valor que va a imprimir tomamos la acción que queramos:

Precio	% IVA	Importe IVA	Total línea
12,99	16,00	2,08	15,07
78,45	16,00	12,55	91,00
140,99	16,00	22,56	163,55
1.099,00	16,00	175,84	1.274,84

Los componentes de Report Manager tienen la propiedad **Visible** pero aunque la activemos no hacen ni caso. Así que si quieres hacer desaparecer un componente elige el color blanco:

```
(FindComponent('TRpExpression9') as TRpExpression).FontColor := clWhite;
```

Esto puede ser muy interesante para mostrar u ocultar ciertas etiquetas según las propiedades de la factura. Por ejemplo, según la forma de pago a lo mejor me interesa mostrar la etiqueta de que se ha pagado al contado:

Como antes de imprimir la factura ya sabemos si debemos imprimir esta etiqueta o no, podemos hacerla invisible sin utilizar el evento **BeforePrint**:

```
procedure TFImprimirFactura.BImprimirClick(Sender: TObject);
begin
  VCLReport.Title := 'Factura';
  VCLReport.Filename := ExtractFilePath(Application.ExeName) +
    'factura.rep';
  VCLReport.Report.Copies := StrToInt(NumCopias.Text);
  VCLReport.Preview := VistaPrevia.Checked;
  if not Contado.Checked then
    with VCLReport.Report do
      if FindComponent('TRpLabel14') is TRpLabel then
       (FindComponent('TRpLabel14') as TRpLabel).FontColor := clWhite;
  VCLReport.Execute;
end;
```

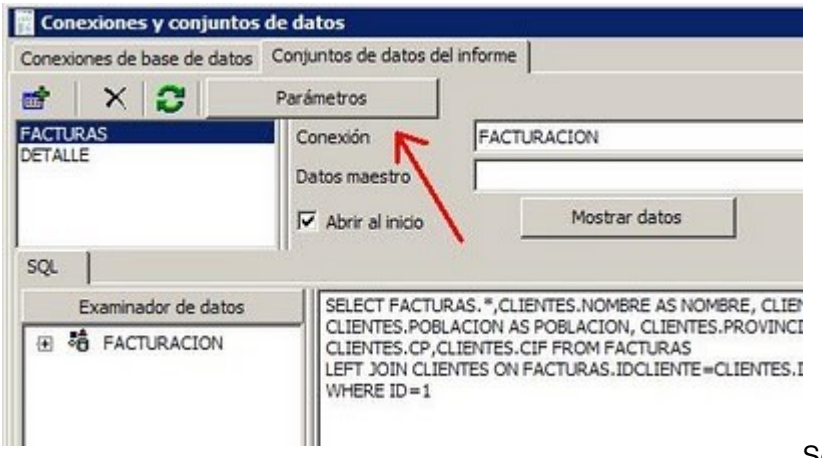
Si no está activado en la factura el **CheckBox** de **Contado** entonces lo hacemos invisible. Para hacerla invisible también le podemos quitarle el texto:

```
if not Contado.Checked then
  with VCLReport.Report do
    if FindComponent('TRpLabel14') is TRpLabel then
      (FindComponent('TRpLabel14') as TRpLabel).Text := '';
```

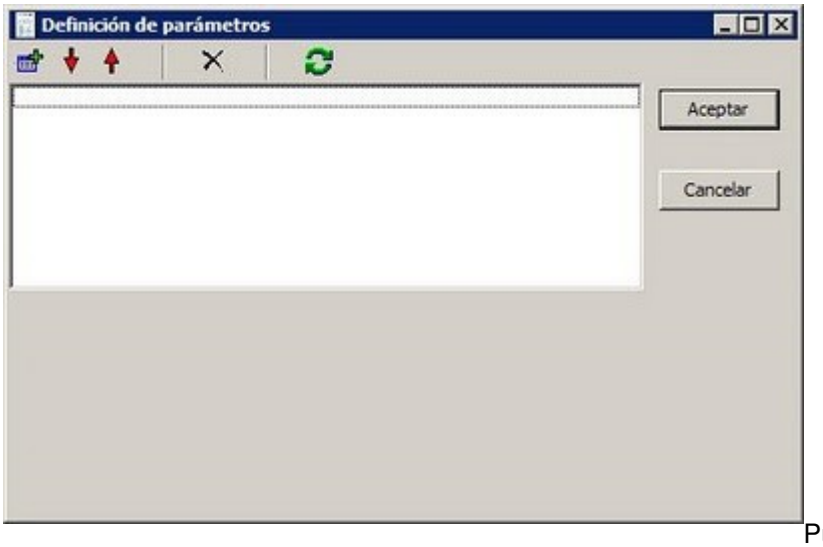
De este modo podemos controlar si se imprimen o no las cuentas bancarias, los distintos tipos de IVA, los recargos de equivalencia o las retenciones. Aunque Report Manager tiene buenas funciones para evaluar expresiones, no hay nada como Delphi para meterle mano a los documentos.

ENVIAR PARÁMETROS AL INFORME

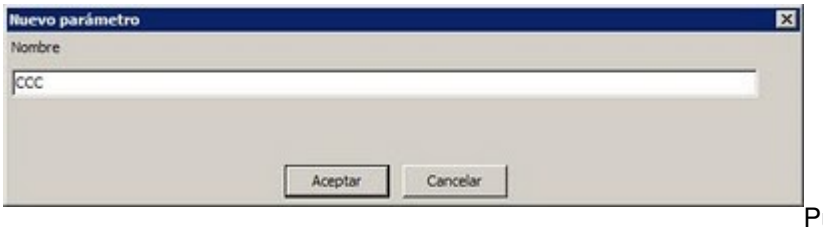
También podemos enviar parámetros formalmente al informe creando parámetros en el mismo. Esto se hace seleccionando las opciones **Informe** – **Configuración de datos** y en el formulario que aparece pulsamos el botón **Parámetros**:



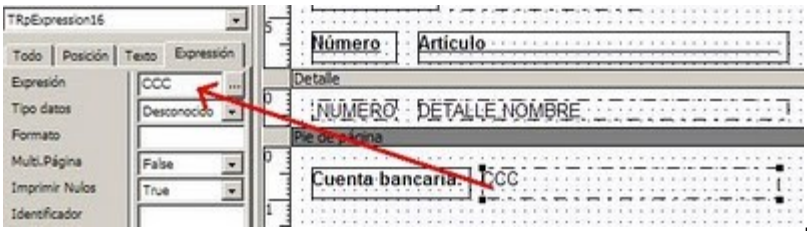
Se abrirá esta ventana:



Pulsamos el botón **Añadir un nuevo parámetro** y como queremos enviarle la cuenta bancaria le ponemos de nombre **CCC**:



Pulsamos **Aceptar** y otra vez **Aceptar** en la ventana anterior. Ahora vamos a crear una expresión al pie de la factura que recoja el valor de este parámetro:

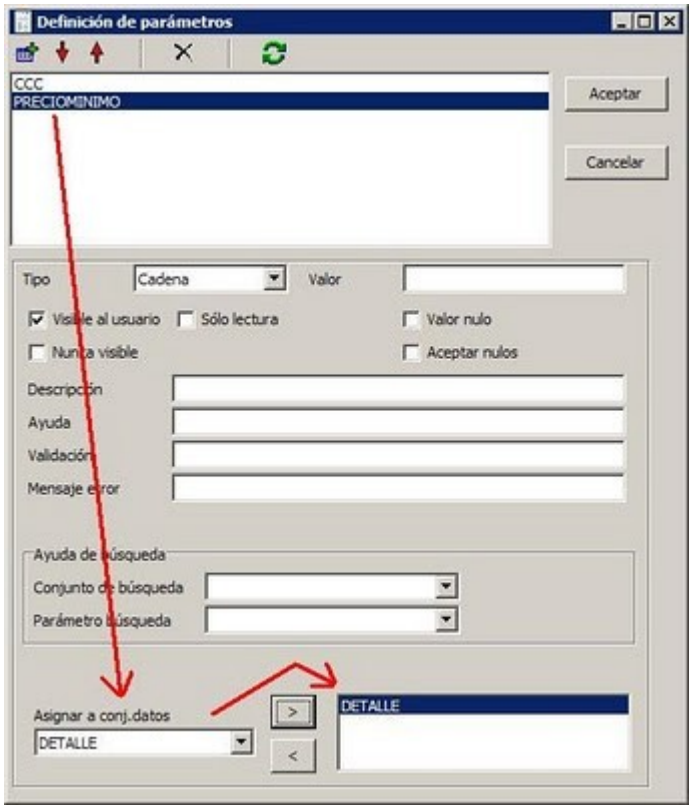


Para mandarle el parámetro al imprimir desde Delphi hacemos esto:

```
procedure TFImprimirFactura.BImprimirClick(Sender: TObject);
begin
  VCLReport.Title := 'Factura';
  VCLReport.Filename := ExtractFilePath(Application.ExeName) +
    'factura.rep';
  VCLReport.Report.Copies := StrToInt(NumCopias.Text);
  VCLReport.Preview := VistaPrevia.Checked;
  VCLReport.Report.Params.ParamByName('CCC').AsString := '1234-5678-44-
1234567890';
  VCLReport.Report.PrepareParamsBeforeOpen;
  VCLReport.Execute;
end;
```

Debemos llamar al método **PrepareParamsBeforeOpen** para que el informe se chupe los parámetros que le mandamos:

Pero es que además podemos utilizar los parámetros para cambiar las condiciones de la SQL de la tabla. Vamos a crear un parámetro para el detalle de la factura llamado **PRECIOMINIMO** para que muestre sólo aquellas líneas de detalle cuyo precio rebase el que le pasamos como parámetro:

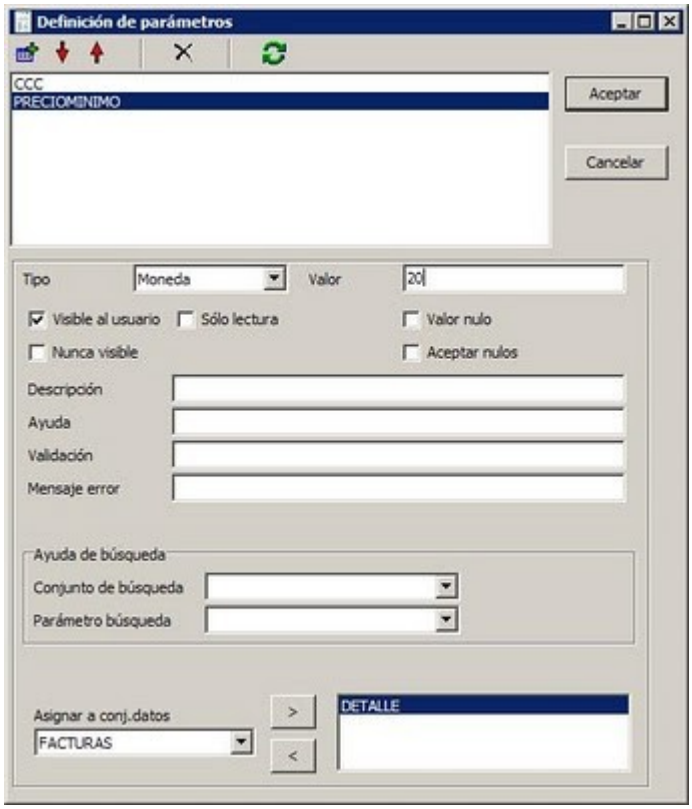


Ahora modificamos la SQL del detalle de este modo:

```
SELECT DETALLE.*,ARTICULOS.NUMERO,
ARTICULOS.NOMBRE FROM DETALLE
LEFT JOIN ARTICULOS ON ARTICULOS.ID=DETALLE.IDARTICULO
WHERE PRECIO >= :PRECIOMINIMO
```

Si intentamos hacer la vista previa en el informe nos dirá esto:

Para probar que vaya bien, entramos de nuevo al parámetro, lo ponemos de tipo moneda y con el valor 20:



Al hacer la vista previa solo mostrará las líneas de detalle cuyo precio es mayor de 20:

Precio	% IVA	Importe IVA	Total línea
78,45	16,00	12,55	91,00
140,99	16,00	22,56	163,55
1.099,00	16,00	175,84	1.274,84

En Delphi le pasamos el parámetro como hemos visto antes:

```
VCLReport.Report.Params.ParamByName('CCC').AsString := '1234-5678-44-
1234567890';
VCLReport.Report.Params.ParamByName('PRECIOMINIMO').Value := 20;
VCLReport.Report.PrepareParamsBeforeOpen;
VCLReport.Execute;
```

Aunque Report Manager pueda parecer un editor de informes algo simple al principio, conforme vamos profundizando en sus detalles vemos que podemos hacer cosas bastante potentes. El próximo artículo veremos como realizar informes a varias bandas y como sumar el contenido de varias de ellas.