Salvador Jover
s.jover@wanadoo.es

Reflexiones sobre QuickReport.

Conocer a QuickReport es una pequeña aventura que no defraudará a quienes la inicien. Su incorporación en el entorno de Delphi hacen de ella el primer contacto con el diseño y la confección de múltiples informes.

El motivo por el que di comienzo la redacción de estas líneas, a las que yo considero más una pequeña reflexión que un artículo, es el de compartir, con vosotros, algunos problemas que se presentan en el diseño de determinado tipo de informes (reportes o como queráis llamarlos). Nada hay de especial en lo que pueda comentar, puesto que abriendo cualquiera de los múltiples libros que nos hablan de Delphi, podemos encontrar un pequeño apartado, dedicado al tema de la confección de informes.

El hecho real es que ya desde la segunda versión de Delphi, (Delphi2.0) se nos incluyó en el entorno, una serie de componentes que debían facilitar la creación rápida de informes. Anteriormente, Delphi 1.0 todavía no incorporaba QuickReport. Borland se apoyó en una herramienta independiente del entorno, llamada “ReportSmith v.2.5”. Con posterioridad, fue abandonada con la aparición de Delphi 2.0, el cual ya incorporaba la paleta de componentes de QuickReport. De ahí a Delphi5.0, ha llovido mucho, y conjuntamente a la evolución y adición de componentes a la VCL, nuestro punto de referencia en el artículo –QuickReport -, también evolucionó a la par.

En la "**figura 1**" podemos ver la paleta de componentes que se incluyen en el producto, y que nos da una idea de la diversidad y de la cantidad de opciones disponibles. Vamos a nombrarlos de izquierda a derecha:

QuickRep, *QRSubDetail*, *QRStringsBand*, *QRBand*, *QRChildBand*, *QRGroup*, *QRLabel*, *QRDBText*, *QRExpr*, *QRSysData*, *QrMemo*, *QrExprMemo*, *QRRichText*, *QRDBRichText*, *QRShape*, *QRImage*, *QRDBImage*, *QRCompositeRecord*, *QRPreview*, *QRTextFilter*, *QRSCVFilter*, *QRHTMLFilter*, *QRChart*. Un total de 23 componentes incorpora, al menos, Delphi 5 Profesional. Por poner un ejemplo, con Delphi2.0 Estándar se suministraban tan solo 11, existentes en la actualidad, aunque lógicamente poco o nada tengan que ver con los anteriores.

Figura 1. Vista de los distintos componentes QuickReport, integrados en la VCL.

¿Pero qué es QuickReport? En términos formales, le podemos definir como un conjunto de componentes nativos para Delphi, que nos van a permitir la creación y diseño rápido de informes, que de otra manera, nos sería francamente difícil hacerlos. Llegar a estas alturas a diseñar una ficha, un listado de clientes mediante el objeto Tprinter puede ser una tarea ardua, y en la que, si ponemos en una balanza: ahorro de tiempo en la implementación del algoritmo de la impresión, calidad gráfica y una presentación, por decir algo, digna, lógicamente optaremos por QR (vamos a llamarle así de ahora en adelante). Su uso, puesto que se integra en la VCL como un componente más, es el habitual de cualquier otro componente: arrastrar y soltar sobre la ficha. Nada más sencillo. El código queda integrado en nuestro ejecutable y no será preciso distribuir fichero alguno. Un mundo de razones, que nos hace pensar que hemos podido llegar al culmen de la perfección y nada más lejos de la realidad: son frecuentes las quejas en los foros de programación acerca de QR, lógicamente estoy hablando de foros de C++ y de Delphi, que son dos de los entornos que lo incorporan.

Sí, son frecuentes las quejas: empezando por la pésima documentación que acompaña al producto y que dificulta el pleno entendimientos de partes esenciales del mismo. Creo que esto es lo más grave. Un

producto además, que como el mismo Delphi, no es traducido al castellano. Se han localizado y reparado numerosos errores, y en cada nueva versión del producto, Borland ha seguido apostando por ellos.

Por poneros un ejemplo, ahora mismo, creo que estoy sufriendo uno de éstos errores:

> Estoy trabajando en el ejemplo que me ha de servir de guía para escribir estas líneas. Todo funciona correctamente. Y sin embargo, si por un momento se me pasara por la cabeza, en tiempo de diseño, buscar la opción *Preview*, (dicha opción nos aparece en el submenú al teclear sobre la ventana con el botón derecho del ratón), se produciría un error grave que daría por finalizada la aplicación de Delphi con un sugestivo pantallazo azul de nuestro querido sistema operativo. ¿Cuál es el problema?. Dicho error me ha sucedido cuando he utilizado la base de datos ACCESS, con acceso a la misma mediante el controlador nativo del BDE. Si en lugar de utilizar ésta, lo hubiera hecho con Paradox o DBase no lo hubiera sufrido y el componente se hubiera comportado con normalidad, dando el resultado esperado. He hecho la comprobación y realmente es así. (Si es que siguen funcionando con normalidad mis pequeñas neuronas :-)).

Fuera de todos estos problemas, QR es una estupenda herramienta de trabajo en eso que tanta pereza da a los programadores, y que consiste nada mas y nada menos, que en la confección de los múltiples listados presentes en las aplicaciones de gestión.

Conocer los ingredientes de un informe.

El principal componente de todos cuantos figuran en la lista de la figura 1 es el primero: *TQuickRep*. Si procedemos a agregarlo a nuestro *Tform*, nos proporcionará una plantilla, una pizarra, sobre la que ir situando el resto de componentes, de igual manera que lo podemos hacer, con nuestras ventanas (*Tform*) al configurar la interfaz gráfica de una aplicación, e ir determinando la posición de cada uno de los controles y componentes que la integrarán. La idea, por decirlo de alguna manera, viene a ser la misma.

Un producto cuyo uso es sencillo

Su uso, puesto que se integra en la VCL como un componente más, es el habitual de cualquier otro componente: arrastrar y soltar sobre la ficha. Nada más sencillo. El código queda integrado en nuestro ejecutable y no será preciso distribuir fichero alguno.

A diferencia de ésta última, el método de trabajo con QR, es la utilización de distintas bandas que se van acomodando automáticamente sobre nuestra plantilla. Y sobre ellas, procederemos a acoplar los distintos componentes imprimibles, tales como por ejemplo el *TQRLabel* o el eficiente *TQRDBTex*t.

Por otro lado es importante destacar que cada una de nuestras bandas puede ser configurada para realizar una misión distinta. El componente *TQRBand* nos muestra en una de sus propiedades más importantes, *BandType* once tipos distintos de comportamiento o valores, a saber:

1.- RbTitle, sobre el que se situará el título del informe

2.-RbPageHeader, cabecera de la página y que nos aparecerá en cada una de las paginas que diseñemos.

3.-RbDetail, es la banda que representa el cuerpo del informe y tiene conexión con la fuente de datos principal, mediante la propiedad *TdataSe*t. Aquí tenemos que tener en cuenta, que este tipo de banda suele ser utilizado cuando queremos listar una serie de registros de una tabla cualquiera de forma secuencial. Cualquier listado de clientes que podamos tener en nuestras manos es un buen ejemplo de su uso. Durante el proceso interno de composición del informe, nuestro QR, accederá progresivamente a cada uno de los registros hasta llegar hasta el último y de no haber, por debajo de éste, otra banda distinta con valor *rbPageFooter*, ira usando cuanto espacio quede libre hasta el final de la página (según las dimensiones establecidas).

4.-RbSubDetail, esta banda representa, en aquellos informes en los que la fuente de datos sea el resultante de una relación maestro-detalle, a la sección de detalle. Su proceder será similar al visto en el anterior. Estas bandas pueden ser combinadas para obtener distintos tipos de informes, de tal manera, que la distinta relación establecida entre las mismas determinará el tipo de presentación.

5.-RbPageFooter, representa al pie de página y estará presente en cada una de las distintas páginas que puedan componer nuestro listado o informe. Mucha atención a ésta, que va a tener importancia en el ejemplo de informe que posteriormente intentaremos crear.

6.-RbSummary, no tiene demasiada dificultad y su uso se limita al denominado Sumario de un escrito

y que siempre figura al pie de la ultima página del mismo. Tanto esta opción como la segunda, la cabecera, pueden ser, desde el mismo QuickRep, desactivadas, para poder evitar su presencia en el informe.

7.-RbGroupHeader, su utilidad se hace patente cuando es enlazada a una banda rbSubDetail y representará la cabecera que anteceda a dicha banda.

8.-RbGroupFooter, por el contrario, y siguiendo el mismo razonamiento, representará el pie de página de dicha banda *rbSubDetail* Estas dos anteriores también nos van a ayudar a construir nuestro ejemplo.

9.-RbColumnHeader, es utilizada en aquellos listados en los que se haga necesaria la presentación por columnas.

10.-RbOverlay, nos da la posibilidad de incorporar a nuestro informe una serie de elementos que serán mostrado en cada una de las páginas del informe superpuestos al contenido del mismo. La idea que más se acerca a su uso es la de servir como fondo de un documento, tal como la imagen del logotipo de la empresa o cualquier otro elemento textual o gráfico que se nos pueda ocurrir.

11.-RbChild, siempre dependiente de otra banda, que será "madre" de la presente y a la que quedará enlazada como subsección de la misma e impresa con posterioridad.

Hemos acabado de ver los distintos once valores por los que podemos optar al insertar una banda dentro de nuestra pizarra. Podríamos seguir, de aquí en adelante, enumerando todas y cada una de las propiedades y métodos de los 23 componentes, pero no encuentro motivo para hacerlo, puesto que nos desviaríamos de la línea central que da razón al artículo. Quizá esta empresa podría ser valida, para la redacción de próximos números, en donde se podrían tratar con profundidad, al menos los componentes más importantes. Hay mucho de lo que hablar y comentar: de crear los reportes dinámicamente –ahora lo vamos a hacer de forma estática -, de la fusión de múltiples informes mediante el componente *TcompositeRecord*, y mucho más. Pienso yo, que nos puede ser más provechoso, apartar de momento dichas definiciones y centrarnos en un ejemplo práctico y sencillo, que nos pueda dar una ligera idea de cómo diseñar un informe cualquiera. Tenemos todos los ingredientes para poder hacerlo: disponemos de un TQuickRep, que nos servirá de pizarra, de varios componentes TQRBand que nos servirán como soporte sobre los que iremos depositando los componentes imprimibles. Y para imprimir podemos optar por la incorporación de algunos *TQRDBText* y *TQRLabel*, de uso mas sencillo. Tened en cuenta que podemos complicarlo tanto como queramos, pero para empezar creo que nos es suficiente.

Metodo de trabajo: el TqrBand.

El método de trabajo con QuickReport, es la utilización de distintas bandas que se van acomodando automáticamente sobre nuestra plantilla. Y sobre ellas, procederemos a acoplar los distintos componentes imprimibles, tales como por ejemplo el *TQRLabel* o el eficiente *TQRDBTex*t

Respecto a estos dos últimos componentes nombrados, el primero nos servirá para enlazar con cualquiera de los campos de las distintas tablas creadas, mientras que el segundo, será utilizado para manipular el texto explicativo que acompaña a los datos extraídos de la base de datos. Son dos componentes imprimibles que iremos repartiendo a nuestro antojo sobre las distintas bandas según diseño. Podemos establecer cierta semejanza entre el componente *TDBEdit* de los controles de acceso a las bases de datos y el componente *TQRDBText*. Asimismo la haremos entre el *TLabel* y el *TQRLabel*.

Si repasamos la ayuda que nos suministra QR, acerca del componente *TQRDBText* observaremos que nos anuncia que es un descendiente de la clase *TQRCustomLabel*, de la que desciende también *TQRLabel*. Las principales propiedades del primero: básicamente *DataSet* y *DataField* nos permitirán enlazar o conectar con los campos de cada tabla. Del segundo, nos basaremos principalmente en la propiedad *Caption*. Existen otras propiedades comunes a los dos que nos ayudarán a mejorar la presentación del informe: *Alignment* (alineamiento del texto), *AlignToBand* (alineamiento respecto a la banda sobre la que están depositados), A*utosize* (ajuste automático del componente a la longitud del texto, disminuyendo o aumentando según dicho valor) y los típicos *Color* o *Font*, sobre los que no es necesario hacer comentario.

Vamos a poner la mesa.

Imaginemos una situación real: tenemos en nuestras manos una pequeña aplicación cuyo objetivo final es listar múltiples fichas de un producto imaginario, que se compone de también de imaginarias

Reflexiones sobre QuickReport.

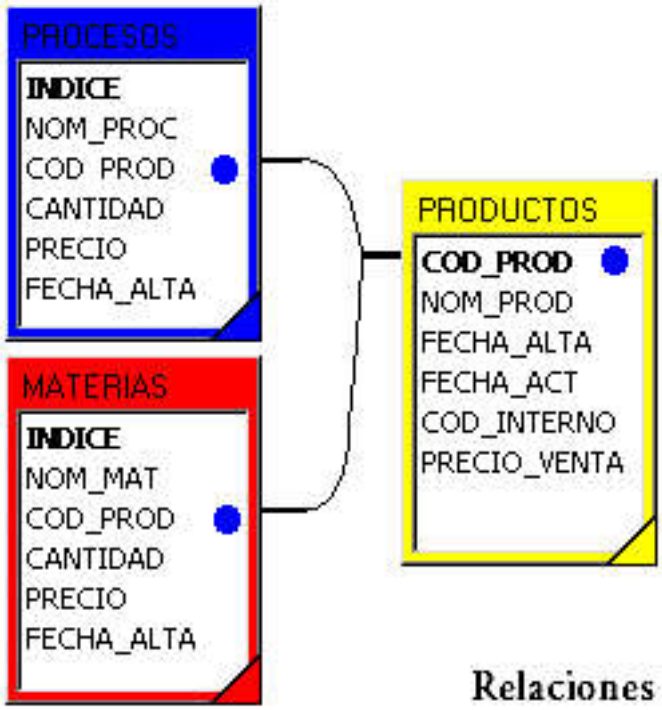

Figura 2. Relaciones establecidas entre las tres tablas

materias primas y de imaginarios procesos productivos. La suma de las materias asignadas a cada producto generará un subtotal. De igual manera, la suma de los procesos asignados a cada producto, generará otro subtotal. Y por supuesto, ambos subtotales nos comunicaran el precio de coste del producto imaginario.

Hablando en términos de programación se establecerá una doble relación maestro-detalle entre la ficticia tabla de productos con la tabla de materias y con la de procesos. Esto, nos queda reflejado en la "**figura 2**". En dicha figura queda patente la relacion de "uno a varios" que se establece entre la tabla maestra y nuestras dos tablas detalles. Podemos ver en la "**Tabla A**" una relación de campos asignados a cada una de ellas, que nos pueda servir para nuestro ejemplo.

Es importante que destaque, para no dar pie a errar, que en la **tabla A** he añadido en color verde aquellos campos calculados creados en nuestro Ttable. Las relaciones que establecen entre ellos son las siguientes:

TABLA DE MATERIAS

TOTAL_MAT = CANTIDAD x PRECIO

TABLA DE PROCESOS

TOTAL_PROC= CANTIDAD x PRECIO

TABLA DE PRODUCTOS

SUB_TOTAL_MAT = Suma Total de Materias sobre el campo TOTAL_MAT
SUB_TOTAL_PROC = Suma Total de Procesos sobre el campo TOTAL_PROC
TOTAL_PROD = SUB_TOTAL_MAT + SUB_TOTAL_PROC

TABLA DE PRODUCTOS	TABLA DE MATERIAS	TABLA DE PROCESOS
(*) **COD_PROD**	(*) INDICE	(*) INDICE
NOM_PROD	NOM_MAT	NOM_PROC
FECHA_ALTA	**COD_PROD**	**COD_PROC**
FECHA_ACT	CANTIDAD	CANTIDAD
COD_INTERNO	PRECIO	PRECIO
PRECIO_VENTA	FECHA_ALTA	FECHA_ALTA
SUB_TOTAL_MAT	***TOTAL_MAT***	***TOTAL_PROC***
SUB_TOTAL_PROC		
TOTAL_PROD		

Tabla A. Descripción de los distintos campos de las tres tablas.

(*) - Indice primario.
XX - Campo sobre el que se realiza la doble relación maestro-detalle
XX - Campos calculados

Reflexiones sobre QuickReport.

Si deseáis obtener más detalle como se ha implementado esto último podéis consultar el "**Listado 1**", en donde podéis encontrar la implementación de las relaciones establecidas en el párrafo anterior.

```
...

implementation

uses prn_fichas, quick_r;

{$R *.DFM}

procedure Tfrm_tablas.bib_imprimirClick(Sender: TObject);
begin
{Si es pulsado el boton nos aparecerá la ficha para que
podamos filtrar el listado desde que producto hasta que
producto. En este caso, la llamada mediante el método
ShowModal, provocará que nuestra ficha principal no pueda
recobrar el foco hasta que no sea cerrada ésta}
frm_principal.showmodal;
end;

procedure Tfrm_tablas.tab_materiasCalcFields(DataSet: TDataSet);
begin
{Establecemos un campo total calculado sobre la cantidad de
materias por su precio, para cada uno de los registros. }
tab_materiasTOTAL_MAT.asInteger:= tab_materiasCANTIDAD.asInteger *
                                        tab_materiasPRECIO.asInteger;
end;

procedure Tfrm_tablas.tab_procesosCalcFields(DataSet: TDataSet);
begin
{Establecemos un campo total, calculado sobre la cantidad de
procesos por su precio, para cada uno de los registros. }
tab_procesosTOTAL_PROC.asInteger:= tab_procesosCANTIDAD.asInteger *
                                        tab_procesosPRECIO.asInteger;
end;

procedure Tfrm_tablas.tab_productosCalcFields(DataSet: TDataSet);
begin
{Vamos primero a sumar mediante una sentencia SQL, que ejecutaremos
mediante el componente TQuery que insertamos en nuestra ficha, el total
de materias asignadas al producto}
que_suma_parcial.sql.clear;
que_suma_parcial.sql.Add(
'SELECT SUM(MATERIAS.CANTIDAD*MATERIAS.PRECIO) '+
'FROM MATERIAS, PRODUCTOS ' +
'WHERE (MATERIAS.COD_PROD = PRODUCTOS.COD_PROD) ' +
'AND (MATERIAS.COD_PROD = ' + IntToStr(tab_productosCOD_PROD.asInteger) + ')');
que_suma_parcial.open;
{Obtenido dicho resultado lo procedemos a asignar a nuestro campo calculado
que pertenece a la tabla maestra de productos.}
tab_productosSUB_TOTAL_MAT.asInteger:= que_suma_parcial.fields[0].asInteger;

{Volvemos a repetir la misma acción pero esta vez con la tabla de procesos}
que_suma_parcial.sql.clear;
que_suma_parcial.sql.Add(
'SELECT SUM(PROCESOS.CANTIDAD*PROCESOS.PRECIO) '+
'FROM PROCESOS, PRODUCTOS ' +
'WHERE (PROCESOS.COD_PROD = PRODUCTOS.COD_PROD) ' +
'AND (PROCESOS.COD_PROD = ' + IntToStr(tab_productosCOD_PROD.asInteger) + ')');
que_suma_parcial.open;
{Obtenemos el total que nos devuelve el componente TQuery y lo asignamos
al campo calculado creado a tal efecto en la tabla maestra de productos.}
tab_productosSUB_TOTAL_PROC.asInteger:= que_suma_parcial.fields[0].asInteger;

{Por fin, podemos a proceder a sumar ambos subtotales y a asignarlo para
obtener el coste del producto.}
tab_productosTOTAL_PROD.asInteger:= tab_productosSUB_TOTAL_MAT.asInteger +
                                    tab_productosSUB_TOTAL_PROC.asInteger;

// ya no necesitamos el componente TQuery y lo podemos cerrar
que_suma_Parcial.Close;
end;
```

Reflexiones sobre QuickReport.

```
procedure Tfrm_tablas.spb_conectarClick(Sender: TObject);
begin
// si pulsamos el botón y esta permanece pulsado
if spb_conectar.down then
  begin
  // abrimos y conectamos todas las tablas
  Database1.Connected:= true;
  tab_productos.Active:= true;
  rejilla_productos.Repaint;
  tab_materias.Active:= true;
  rejillas_materias.repaint;
  tab_procesos.Active:= true;
  rejilla_procesos.Repaint;
  bib_imprimir.enabled:= true; // activamos el botón para imprimir
  end
else  // en el caso contrario
  begin
  // Cerramos todas las tablas
  Database1.Connected:= false;
  tab_productos.Active:= false;
  tab_materias.Active:= false;
  tab_procesos.Active:= false;
  bib_imprimir.enabled:= false; // desactivamos el botón para imprimir
  end;
end;

end.
```

Listado 1. Implementación de los campos calculados.

Ahora bien, y aquí puede estar una de las claves del modelo de informe elegido (que no ha sido al azar), el listado debe de ser múltiple, donde podamos elegir desde que producto hasta que producto y las condiciones adicionales son que: Un producto podrá necesitar cuantas páginas sean necesarias para cubrir el total de registros detalles y su sucesor lo hará en pagina independiente iniciando una nueva impresión.

Condiciones para nuestro informe o ficha.

El listado debe de ser múltiple, donde podamos elegir desde que producto hasta que producto y las condiciones adicionales son que: Un producto podrá necesitar cuantas páginas sean necesarias para cubrir el total de registros detalles y su sucesor lo hará en página independiente iniciando una nueva impresión.

Además, lógicamente, puesto que hablamos de una ficha de un producto la pagina impresa ha de quedar dividida en tres secciones fijas: '*una cabecera*' en donde se detallarán los principales campos o características del producto, '*un cuerpo central*' en el que se expandirán todos los registros de detalles siguiendo un orden y ofreciendo un subtotal parcial para las materias otro para los procesos, *y 'un pie de pagina'* en el que habremos de detallar los subtotales y totales del producto.

Aunque la idea pueda parecer rebuscada, nada mas lejos de la realidad. Para ver claro nuestro ejemplo podríamos tener el pensamiento puesto en la impresión de múltiples facturas, donde se siguen las reglas antedichas. Y no son nada raras, preguntas sobre lo dicho en los foros de programación.

Así que, puestos manos a la obra, ya disponemos sobre la mesa de trabajo nuestros tres primeros componentes, que los vamos a describir de la siguiente forma:

Tfrm_tablas es la ficha principal. Va a ser la que nos suministre los datos a nuestra "pizarra". En ella están los tres componentes *Ttable*, acompañados de un trío de componentes *TDBGrig* con sus respectivos TdataSource, un TQuery, para el cálculo de las sumas parciales y poco más. Ah, me olvidaba: también he dispuesto de un pequeño *TbitBtn* (un simple botón) para poder hacer aparecer la segunda de las ventanas, con una simple llamada a *ShowModal.*

Tfrm_principal es la segunda forma o ventana, que aparecerá a la llamada anterior y nos servirá para filtrar los códigos de los productos que deseamos listar. Nos basta un: <desde... hasta>, mediante dos componentes *TDBLookComboBox*, para hacer dicha selección. Pongamos un botón que nos permita lanzar la impresión de la tercera y ultima ventana y simplifiquemos al máximo nuestro código:

Reflexiones sobre QuickReport.

```
    Procedure Tfrm_principal.bib_imprimir(Sender: Tobject);
    Begin
    ...
    ...
    {Solo en el caso de que el codigo del primer producto elegido sea menor o igual que
el segundo producto elegido}
    if StrToInt(db_cbx_desde.text) <= StrToInt(db_cbx_hasta.text) then
       Begin
       // Preparamos el filtro a nuestra tabla de productos
       Frm_Tablas.tab_productos.filter:= 'COD_PROD >= ' +
                                         Chr(39) + db_cbx_desde.text + Chr(39) +
                                        'AND COD_PROD <= ' +
                                         Chr(39) + db_cbx_hasta.text + Chr(39);
       // Lo activamos
       Frm_Tablas.tab_productos.filtered:= true;
           Frm_Impresion.QuickRep1.ShowProgress:= true;
       {Visualizamos la preview del informe generado en espera de dar el visto bueno
para su impresión. Si hubieramos querido imprimir directamente basta tan solo con la
sustitución de Preview por Print}
       Frm_Impresion.QuickRep1.Preview;
       End;
    End;
```

Más condiciones.

La pagina impresa ha de quedar dividida en tres secciones fijas: '*una cabecera*' en donde se detallarán los principales campos o características del producto, '*un cuerpo central*' en el que se expandirán todos los registros de detalles siguiendo un orden y ofreciendo un subtotal parcial para las materias otro para los procesos, *y* '*un pie de pagina*' en el que habremos de detallar los subtotales y totales del producto.

TFrm_Impresión es la ventana (*Tform*) en la que insertaremos nuestro *TQuickRep* para empezar a trabajar en el diseño del informe. Por el principio, su apariencia es la de ser un lienzo cuadriculado, que nos informa sobre cual va a ser el área útil de impresión por defecto. Todas estas características, tal como el tamaño del papel, la orientación, las dimensiones, etc. podrán ser modificadas, tanto en tiempo de diseño como de ejecución.

Situados ya en este punto, ya podemos empezar a cocinar. ;-)

De la mesa a la cocina: ingredientes a la olla.

La "**Figura 3**" nos va a dar una idea exacta del esquema de impresión de nuestra ficha de productos.

Pero nos queda lo más importante, el porqué se ha hecho así, o por lo menos, el porqué yo lo he hecho así:

Nos podemos hacer un esquema mental de los requisitos demandados por el informe: sabemos que nos están demandando que dividamos cada una nuestras páginas en una composición fija en la que debe aparecer una cabecera, un cuerpo y un pie de pagina. Luego vamos a insertar sobre nuestro lienzo los componentes *TQRBand* que nos permitan diferenciar las tres secciones.

El primero de ellos representará a la cabecera de la página, (en la figura 3, zona cabecera). Necesitamos asignar la función que desempeñará en nuestro informe y para eso vamos a asignar la propiedad *BandType* a *RbPageHeader*. Yo recomendaría asignar el nombre del componente con palabras fáciles de reconocer como "Cabecera_de_Ficha" en el momento de su adición. Posteriormente nos facilitará establecer las relaciones entre las distintas bandas que aparezcan en el informe.

En esta banda vamos a depositar todos los datos de la tabla maestra que creamos conveniente e incluso adornar el listado mediante un *TQrShape* (podemos imaginarlo como un componente *Tbevel*), agrupando de esa manera visualmente aquellos campos que deseemos resaltar. Para remarcar mas la división por zonas he asignado a la propiedad '*Frame.DrawBottom*' el valor de True y en '*Frame.Color*' el de *clGreen*, dibujando así, a pie de banda una línea horizontal de color verde.

Dejaremos para el final la zona de cuerpo o de detalles y comentaremos a continuación, que para

Reflexiones sobre QuickReport.

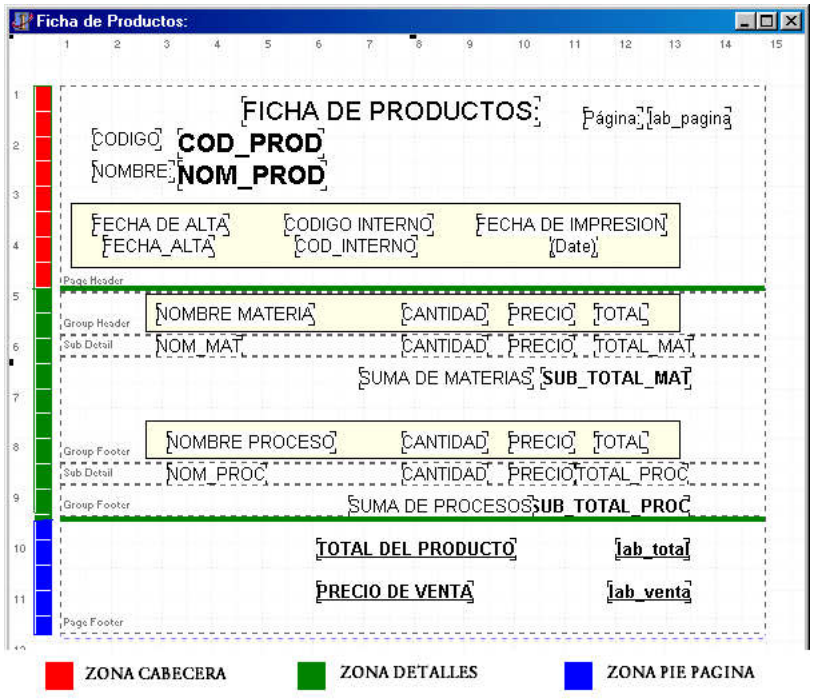

Figura 3. Nuestro diseño del informe.

obtener la zona de pie de página, volveremos a insertar un nuevo *TQRBand* cuya propiedad BandType será asignada a *RBPageFooter*. Está nos servirá para reflejar aquellos valores de los campos dependientes de la zona de detalles, como los totales parciales y el total de producto.

Con la inserción de estas dos bandas hemos conseguido dividir en tres zonas diferenciadas nuestro informe puesto que el ancho de las dos va a ser respetado. Una se situara en la cabecera de la página y la otra en el pie de la misma. En el centro vamos a crear el entramado que configurará todos los detalles de las tablas secundarias.

Necesitaremos dos componentes *TqrBand* con la función de *SubDetail*. Uno para conectar con la tabla de materias mediante la propiedad *DataSet*, y el segundo para conectar con la tabla de procesos mediante la misma propiedad. No he comentado que cuando insertamos en un principio el componente TQuickRep asignamos su propiedad *DataSet* conectándolo con la tabla maestra.

Está claro que nuestra primera banda de detalles necesitará una cabecera y un pie de banda. Y lo mismo podemos decir de nuestra segunda banda subdetail. Luego, puesto que van a continuación una de otra podemos prescindir del pie del primero y aprovechar la cabecera del segundo para realizar dicha función de unión o viceversa.

Veámoslo esquemáticamente:

GroupHeader- SubDetail(Materias)- GroupFooter- SubDetail(Procesos)- GroupFooter.

Para poder efectuar el encadenamiento entre estas cinco bandas contamos con un par de propiedades del componente TQRSubDetail que son respectivamente *HeaderBand* y *FooterBand*, a las que les daremos los valores apropiados según la distribución. Resumiendo, le vamos a decir a cada uno de las bandas *Subdetail* cual será la banda que efectuará dichas funciones.

Para finalizar esta parte, deberemos enlazar nuestro primer componente TGroupHeader de la segunda sección, a nuestra cabecera de página, y para eso existe una propiedad cuyo nombre es LinkBand y que le viene a informar a Qr que existe un enlace entre ambas y que una debe imprimirse a continuación de la otra.

A partir de este momento, comenzaremos a distribuir en cada una de las bandas, tantos componentes TQRLabel y TQRDBText como nos sean necesarios, para completar la información que deseamos recoja nuestro informe o listado.

Y los problemas...

Los problemas no tardan en aparecer. El más habitual, es el que se produce cuando en el pie de página (la zona de pie de página de la figura 3), es insertado algún componente TQRDBText conectado con la tabla maestra. Si el usuario de la aplicación listara una ficha solamente, no se apercibiría del mismo, no existiría el problema. Cuando el usuario decida listar mas de un producto, se encontrará con la desagradable sorpresa de que los totales y subtotales que figuran en el pie de pagina no corresponden a los productos, en el momento en el que un producto se componga de mas de una pagina.

Reflexiones sobre QuickReport.

El porqué es muy fácil de comprender, aunque a mí me llevó un doloroso tiempo descubrirlo. Cuando nuestro Qr elabora el informe, inmediatamente que ha leido el ultimo de los registros detalle de un determinado producto, dará por supuesto que no va a encontrar referencias al mismo en nuestro pie de página y buscará el siguiente producto que deba listar. Si existiera algún TQRDBText en dicho pie de página leerá el valor del registro del siguiente producto, por lo que dichos totales, subtotales o cualquier campo que haga referencia a la tabla maestra estará equivocada.

Para darse cuenta, fácilmente, de lo que os comento, no teneis más que añadir, en el ejemplo con el que se acompaña el artículo, un componente TQrDbText (apuntando a un campo de la tabla de productos) en la zona de pie de pagina y ejecutarlo.

Encontrar un problema...

Cuando el usuario decida listar mas de un producto, se encontrará con la desagradable sorpresa de que los totales y subtotales que figuran en el pie de pagina no corresponden a los productos, en el momento en el que un producto se componga de mas de una pagina. El porqué es muy fácil de comprender, aunque a mí me llevó un doloroso tiempo descubrirlo. Cuando nuestro Qr elabora el informe, inmediatamente que ha leido el ultimo de los registros detalle de un determinado producto dará por supuesto que no va a encontrar referencias al mismo en nuestro pie de página y buscará el siguiente producto que deba listar.

Si os dais cuenta, en todos los listados de ejemplo que figuran en la ayuda del producto, cuentan con pies de pagina pero tan solo se les asignan componentes como el TQRSysData, con valores como página o Fecha.

Asimismo, me he visto obligado a no utilizar el componente TQRSysData para numerar las páginas, ya que de haberlo hecho, su numeración no tendría en cuenta el cambio de impresión al registro de un producto posterior.

¿Solución?

La más fácil a estos problemas es utilizar los eventos BeforePrint. En el **Listado 2**, podréis encontrar la implementación que he utilizado para alcanzar los fines propuestos. En lugar de utilizar TQRDBText en los pies de página del informe he utilizado el *TQRLabel* y he asignado los valores de los campos que me han interesado a través de dichos eventos. Antes de imprimir la cabecera de la página el procedimiento (TFrm_impresión.Cabecera_GeneralBeforePrint) borra cualquier referencia a los datos asignando sus Caption a [‘’]: por ejemplo, *lab_total_prod.caption:= ‘’;* o incremento la variable global página en una unidad.

Elijo el momento que me interesa para visualizar los datos del pie de página y para ello lo hago en el evento *TFrm_impresión.Pie_de_ProcesoBeforePrint* y al mismo tiempo inicializo dicha variable global para que empiece a numerar la ficha de un nuevo producto desde el primer numero.

```
unit prn_fichas;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  ExtCtrls, QuickRpt, Qrctrls, Db, DBTables;

type
  Tfrm_impresion = class(TForm)
    QuickRep1: TQuickRep;
    Cabecera_materias: TQRBand;
    Cabecera_General: TQRBand;
    lab_titulo: TQRLabel;
    qr_codigo: TQRDBText;
    lab_nombre: TQRLabel;
    lab_codigo: TQRLabel;
    qr_nombre: TQRDBText;
    cuadro_adorno: TQRShape;
    lab_fecha_alta: TQRLabel;
    qr_fecha_alta: TQRDBText;
    lab_codigo_interno: TQRLabel;
    qr_cod_interno: TQRDBText;
```

Reflexiones sobre QuickReport.

```
    qr_fecha_actual: TQRSysData;
    lab_fecha_impresion: TQRLabel;
    Detalle_Materias: TQRSubDetail;
    Detallle_Procesos: TQRSubDetail;
    Pie_de_Enlace: TQRBand;
    qr_nom_proc: TQRDBText;
    qr_cantidad_proceso: TQRDBText;
    qr_precio_proceso: TQRDBText;
    qr_total_proceso: TQRDBText;
    qr_nom_mat: TQRDBText;
    qr_cantidad_materia: TQRDBText;
    qr_precio_materia: TQRDBText;
    qr_total_materia: TQRDBText;
    Pie_de_Proceso: TQRBand;
    qr_sub_total_procesos: TQRDBText;
    lab_nombre_materia: TQRLabel;
    lab_precio_materia: TQRLabel;
    lab_total_materia: TQRLabel;
    qr_cuadro_1: TQRShape;
    qr_sub_total_materias: TQRDBText;
    lab_suma_materias: TQRLabel;
    lab_nombre_proceso: TQRLabel;
    lab_cantidad_proceso: TQRLabel;
    lab_precio_proceso: TQRLabel;
    lab_total_proceso: TQRLabel;
    qr_cuadro_2: TQRShape;
    lab_suma_procesos: TQRLabel;
    Pie_de_Pagina: TQRBand;
    lab_precio_venta: TQRLabel;
    lab_total_prod: TQRLabel;
    lab_total: TQRLabel;
    lab_venta: TQRLabel;
    lab_pagina: TQRLabel;
    lab_Pagina_titulo: TQRLabel;
    lab_cantidad_materia: TQRLabel;
    QRImage1: TQRImage;
    procedure QuickRep1AfterPrint(Sender: TObject);
    procedure QuickRep1AfterPreview(Sender: TObject);
    procedure Pie_de_ProcesoBeforePrint(Sender: TQRCustomBand;
      var PrintBand: Boolean);
    procedure Cabecera_GeneralBeforePrint(Sender: TQRCustomBand;
      var PrintBand: Boolean);
    procedure QuickRep1BeforePrint(Sender: TCustomQuickRep;
      var PrintReport: Boolean);
  private
    { Private declarations }
  pagina: integer;
  public
    { Public declarations }
  end;

var
  frm_impresion: Tfrm_impresion;

implementation

uses base;

{$R *.DFM}

procedure Tfrm_impresion.QuickRep1AfterPrint(Sender: TObject);
begin
{No hay que olvidar que antes de devolver el control a la ficha
 principal hemos de cancelar el filtro. Lo podemos hacer al cerrar
 esta ventana.}
frm_Tablas.tab_productos.filtered:= false;
end;
```

Reflexiones sobre QuickReport.

```
    procedure Tfrm_impresion.QuickRep1AfterPreview(Sender: TObject);
    begin
    {Ver el comentario anterior.}
    frm_Tablas.tab_productos.filtered:= false;
    end;

    procedure Tfrm_impresion.Pie_de_ProcesoBeforePrint(Sender: TQRCustomBand;
      var PrintBand: Boolean);
    begin
    {Antes de imprimir el último de los registros detalles asociados al
    producto, procedemos a asignar los valores a las distintos componentes
    TQRLabel que nos componen el pie de página. En ese momento, todavia
    tenemos acceso al valor del registro de la tabla maestra y por eso
    se han realizado las asignaciones.}
    lab_total_prod.Caption:= 'PRECIO COSTE PRODUCTO';
    lab_total.Caption:= FormatFloat('#,####,##0',
frm_tablas.tab_productos.fields[8].asCurrency) + ' Ptas';
    lab_precio_venta.Caption:= 'PRECIO DE VENTA';
    lab_venta.Caption:= FormatFloat('#,####,##0',
frm_tablas.tab_productos.fields[5].asCurrency) + ' Ptas';
    pagina:= 1;
    end;

    procedure Tfrm_impresion.Cabecera_GeneralBeforePrint(Sender: TQRCustomBand;
      var PrintBand: Boolean);
    begin
    {En cada nueva página, que es lo que simboliza este evento, puesto que
    representa al momento de imprimir la Cabecera, y recordemos que dicha
    banda se repite en cada una de las páginas, ocultamos los detalles del
    pie de página que no queramos que se visualizen. Si en el transcurso de
    la impresión de la misma, no se dispara el evento OnBeforePrint del
    componente Pie_de_Proceso, quedara vacia la zona de pie de página de la
    misma.}
    lab_total_prod.Caption:= '';
    lab_total.Caption:= '';
    lab_precio_venta.Caption:= '';
    lab_venta.Caption:= '';
    lab_pagina.caption:= IntToStr(pagina);
    // incrementamos el contador que nos devuelve el nº de pagina de la ficha
    pagina:= pagina + 1;
    end;

    procedure Tfrm_impresion.QuickRep1BeforePrint(Sender: TCustomQuickRep;
      var PrintReport: Boolean);
    begin
    // antes de imprimir el documento, ponemos a 1 el nº de pagina inicial
    pagina:= 1;
    end;

    end.
```

Listado 2. Codigo de TFrm_Impresión.

Otro problema que he creído detectar y que vosotros podréis comprobar: Si os decidís a insertar otro de los componentes estrella de QR, como lo es le TQRExpr, cuya principal propiedad *Expression*, os permitirá efectuar operaciones entre los distintos campo de las tablas e imprimir el resultado. Si optáis por hacerlo en tiempo de diseño existe a vuestra disposición un “Wizard” con el que podréis crear dichas expresiones de manera sencilla, pero para poder acceder a los campos de una tabla cualquiera, dicha fuente de datos (Ttable o Tquery) deberá estar en la misma ventana en la que se halla el TQuickRep o bien, que es lo habitual, en el componente TdataModule que suele alojar todas las tablas y querys de la aplicación. Intentarlo desde cualquier otra ventana. El “Wizard” del componente TQRExpr no podrá establecer conexión con la fuente de datos aunque la propiedad Expression si que reconozca dichas tablas y dichos campos, de hacer manualmente dichas asignaciones.

Como resumen...

El uso de Qr no resultará tan difícil cuando juguemos, en el sentido más amplio, con él, cuando encontremos que no es tan fiero el perro como lo pintan y que con un poco de paciencia y buen humor, se aprende lo que ahora nos parece imposible.