# EXPEDIENTE 48/18 "Migración del sistema de Gestión de Tasas y Precios Públicos de la CARM (QUESTOR) al entorno Angular/NodeJS"

Se ha recibido la siguiente consulta:

Estamos interesados en presentar una propuesta al concurso de migración a ANGULAR, pero tenemos un par de cuestiones que plantearos.

"La primera es si es posible disponer de una muestra de código origen para que probemos alguna herramienta que pensamos puede acelerar la migración, pero que para estar completamente seguros y ajustar la valoración queremos comprobar. Para ello os pedimos que nos hagáis llegar por el medio que consideréis más adecuado un par de aplicaciones/funcionalidades representativas de entre las que hay que migrar.

La segunda cuestión es que en la lista de Funcionalidades actuales de QUESTOR (página 15 del PPT), y en la relación de componentes de cada elemento funcional, termina en muchos de ellos con un "etc" o puntos suspensivos. ¿Es posible que nos deis una estimación del esfuerzo o al menos un listado de los otros elementos no están especificados y sin embargo habría que mirar y que están detrás de los "etc"? La cuestión es que esos puntos suspensivos provocan una incertidumbre en la valoración ya que desconocemos a qué se refiere."

Con respecto a la primera cuestión planteada:

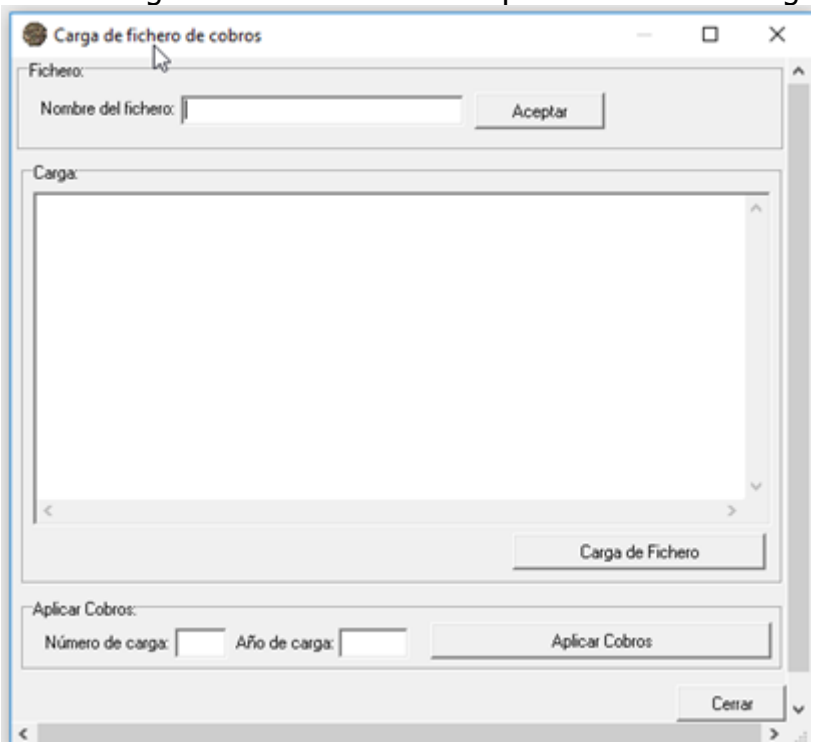
"La primera es si es posible disponer de una muestra de código origen para que probemos alguna herramienta que pensamos puede acelerar la migración, pero que para estar completamente seguros y ajustar la valoración queremos comprobar. Para ello os pedimos que nos hagáis llegar por el medio que consideréis más adecuado un par de aplicaciones/funcionalidades representativas de entre las que hay que migrar."

Se facilita el código de las funcionalidades que cuelgan del menú:

- Apremios -> Aplicar Ingresos de la Agencia -> Carga Fichero
- Listados -> Histórico de operaciones

Para ambas funcionalidades se indica el código de dos acciones de cada una de ellas.

- Carga de ficheros de cobros procedentes de la Agencia




1. El botón de 'Aceptar' valida el nombre del fichero *.dat que se ha introducido por teclado que corresponda con el nombre tipificado a un fichero de cobros procedente de la Agencia, y si la validación es en positivo procedente a conectarse al FTP para recuperar los datos de dicho fichero que mostrará en el bloque 'Carga'

```
procedure TFrmCargarCobros.btnAceptarClick(Sender: TObject);
var
   Str: TMemoryStream;
   NomFich, error: String;
begin
   Str := TMemoryStream.Create;
  NomFich := edFichero.Text;
   OrgSig := Copy(NomFich, 4, 4);

  if (copy(Nomfich, 1, 3) = 'CCQ') OR (copy(NomFich, 1, 3) =
'CMQ')then
        tipoCobro := copy(NomFich, 2, 1)
  else begin
       minfo('El fichero no corresponde a un fichero de cobros',
'');
       exit;
  end;
```

```
    {$IFDEF LOCAL}
      if BajarFicheroLoc(Str, NomFich, error) then
      begin
         memfic.Lines.LoadFromStream(Str);
         pnlCarga.Enabled := true;
      end
      else
      begin
         merror (error, 'Carga');
      end;
    {$ENDIF}

    {$IFNDEF LOCAL}
    if BajarFichero(Str, NomFich, error) then
    begin
       memfic.Lines.LoadFromStream(Str);
       pnlCarga.Enabled := true;
    end
    else
    begin
       merror (error, 'Carga');
    end;
    {$ENDIF}
end;
```

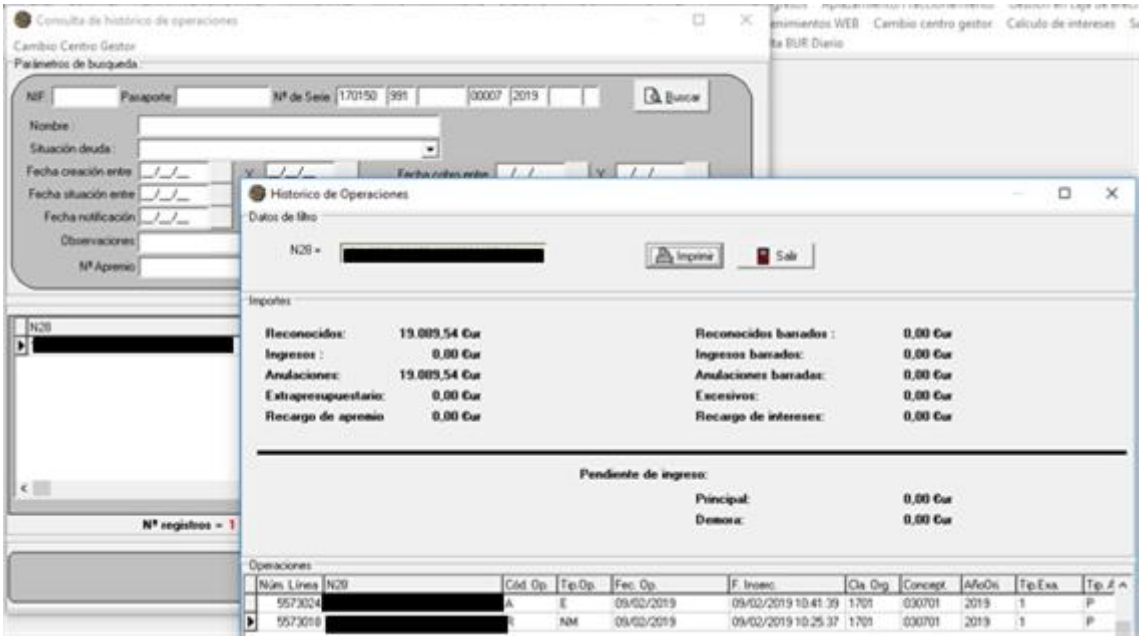
- El botón 'Carga de Fichero' almacenará en base de datos de Questor los datos incluidos en el fichero *.dat

```
procedure TFrmCargarCobros.btnCargaClick(Sender: TObject);
var
   err: String;
begin
   if CargarFichero(err) then
   begin
      minfo('El Fichero se ha cargado correctamente: Número
carga - ' + NumCar + '/' + AnoCar, 'Carga');
      pnlAplicar.Enabled := true;
      edNumCar.Text := NumCar;
      edAnoCar.Text := AnoCar;
   end
   else
   begin
      merror(err, 'Carga')
   end;
end;

function TFrmCargarCobros.CargarFichero(Var error: String):
Boolean;
var
   Ind: Integer;
```

```
    q: TQuery;
    sql: String;
begin
    if SocDFi <> TFuncBd.SociedadByOrsSig(Copy(edFichero.Text, 4,
4)) then
    begin
       error := 'No se puede cargar el archivo de otro
organismo';
       Result := false;
       Exit;
    end;

    dm1.DBFisca.StartTransaction;
    If not InsertarCabecera(error) then
    begin
       dm1.DBFisca.Rollback;
       Result := false;
       Exit;
    end;

    ImpTotalAux := 0;
    TotLinAux   := 0;
    ImpCom      := 0;
    ImpNor      := 0;

    For Ind := 1 to memfic.Lines.Count - 1 do
    begin
       Result := InsertarLinea(error, Ind);
       If not Result then
       begin
          dm1.DBFisca.Rollback;
          Exit;
       end;
    end;

    If (ImpTotalAux <> ImpTotal) or (TotLineas <> TotLinaux) then
    begin
          dm1.DBFisca.Rollback;
          Result := false;
          error  := 'No coinciden el total de lineas o el importe
total';
          Exit;
    end;

    try
       Sql := 'UPDATE FSCRECEJE SET IMPCOM = ' +
FloatSql(ImpCom/100) +
              ', IMPNOR = ' + FloatSql(ImpNor/100) + ' ' +
              'WHERE NUMCAR = ' + NumCar + ' AND ANOCAR = ' +
AnoCar;

       TBdatos.ExeQ(qUpd, Sql);
    except
       dm1.DBFisca.Rollback;
```

```
        Raise;
      end;

      dm1.DBFisca.Commit;
    end;
```

- Histórico de Operaciones:




1. El botón 'Buscar' buscará las tasas que cumplen con los parámetros introducidos en el filtro de selección

```
procedure TFrmConsulAut.btnBusClick(Sender: TObject);
var
  s, Sql1, Aux, Aux2, OrdenSql, s2, Sql2, AuxApre: string;
  s3, s4, Aux3, Aux4: string;
  sqlauxmain, sql3, sql4: string;
  registros, practicado, ingresado: Double;
  insertpos: integer;
begin
  grdCon.TitleFont.Color := clWindowText;
  grdCon.TitleFont.Style := [];

  if cdsCon.IndexName <> EmptyStr then
cdsCon.DeleteIndex(cdsCon.IndexName);

  if ValidaNif then
  begin
    imp_total := 0;
```

```
imp_ingresado := 0;
case tipo of
  tdRevisable: begin
      ugi_actual_estricta := true;

      tipo := auto;
      s := formarSql(true);

      tipo := liqui;
      s2 := formarSql;

      tipo := recibo;
      s3 := formarSql;

      tipo := tiquet;
      s4 := formarSql;

      tipo := tdRevisable;
    end;
  tdautoweb: begin
      ugi_actual_estricta := false;
      tipo := auto;
      s := formarSql(true);
      tipo := tdautoweb;
      s2 := formarSql;
    end;
else begin
    ugi_actual_estricta := false;
    s := formarSql;
  end;
end;

if tipo = webAuto then
  s2 := formarSql2;

if tipo = historico then
begin
  tipo := liqui;
  s := formarSql;
  tipo := recibo;
  s2 := formarSql;
  tipo := historico;
end;

case tipo of
  historico: begin
      sql1 := formarSQlContador(liqui, s);
      sql2 := formarSQlContador(recibo, s2);
    end;
  webAuto: begin
      sql1 := formarSQlContador(tipo, s);
      sql2 := formarSQlContador(liqui, s2);
    end;
```

```
    tdRevisable: begin
        sql1 := formarSQlContador(auto, s, true);
        sql2 := formarSQlContador(liqui, s2);
        sql3 := formarSQlContador(recibo, s3);
        sql4 := formarSQlContador(tiquet, s4);
        sqlauxmain := formarSQlContador(tipo, '');
      end;
    tdautoweb: begin
        sql1 := formarSQlContador(auto, s, true);
        sql2 := formarSQlContador(tdautoweb, s2, true);
        sqlauxmain := formarSQlContador(tdRevisable, '');
      end;
  else
    sql1 := formarSQlContador(tipo, s);
  end;

  if s <> '' then
  begin
    if TGestora.SoportaCambio then
      case tipo of
        historico: begin
            aux := TGestora.RangoAutliqAuto(s);
            aux2 := TGestora.RangoReciboAuto(s);
          end;
        tdRevisable: begin
            aux := TGestora.RangoAutliqAuto(s);
            aux2 := TGestora.RangoAutliqAuto(s2);
            aux3 := TGestora.RangoReciboAuto(s3);
            aux4 := TGestora.RangoAutliqAuto(s4);
          end;
        tdAutoWeb: begin
            aux := TGestora.RangoAutliqAuto(s);
            aux2 := TGestora.RangoAutliqAuto(s2);
          end;
        recibo: begin
            aux := TGestora.RangoReciboAuto(s);
          end;
      else begin
          aux := TGestora.RangoAutliqAuto(s);
        end;
      end;
    OrdenSql := ' ORDER BY N28 ' + orden;
    if orden = 'ASC' then
      orden := 'DESC'
    else
      orden := 'ASC';

    case tipo of
      historico: begin
          s := s + aux + cQUERY_UNION + s2 + aux2 + OrdenSql;
          sql1 := sql1 + aux + cQUERY_UNION + sql2 + aux2;
        end;
      webAuto: begin
          s := s + aux + cQUERY_UNION + s2 + aux + OrdenSql;
```

```
            sql1 := sql1 + aux + cQUERY_UNION + sql2 + aux;
          end;
        tdRevisable: begin
            s := s + aux;
            s2 := s2 + aux2;
            s3 := s3 + aux3;
            s4 := s4 + aux4;
            sql1 := sql1 + aux;
            sql2 := sql2 + aux2;
            sql3 := sql3 + aux3;
            sql4 := sql4 + aux4;
          end;
        tdautoweb: begin
            s := s + aux;
            s2 := s2 + aux2;
            sql1 := sql1 + aux;
            sql2 := sql2 + aux2;
          end;
      else begin
          s := s + aux + OrdenSql;
          sql1 := sql1 + aux;
        end;
      end;
      case tipo of
        tdRevisable: begin
            qrycon.close;
            qrycon.sql.clear;
            qrycon.sql.add(s);
            qrycon.sql.add(cQUERY_UNION);
            qrycon.sql.add(s2);
            qrycon.sql.add(cQUERY_UNION);
            qrycon.sql.add(s3);
            qrycon.sql.add(cQUERY_UNION);
            qrycon.sql.add(s4);
            qrycon.sql.add(ordenSQL);
            qrycon.Prepare;
            qrycon.Open;
          end;
        tdautoweb: begin
            qrycon.close;
            qrycon.sql.clear;
            qrycon.sql.add(s);
            qrycon.sql.add(cQUERY_UNION);
            qrycon.sql.add(s2);
            qrycon.sql.add(ordenSQL);
            qrycon.Prepare;
            qrycon.Open;
          end;
      else begin
          TBdatos.abreq(qrycon, s);
        end;
      end;
      case tipo of
        tdRevisable: begin
```

```
                qagr.close;
                qagr.sql.clear;
                insertpos := pos(subquery, sqlauxmain);
                qagr.SQL.add(copy(sqlauxmain, 1, insertpos - 1));
                qagr.sql.add(sql1);
                qagr.sql.add(cQUERY_UNION);
                qagr.sql.add(sql2);
                qagr.sql.add(cQUERY_UNION);
                qagr.sql.add(sql3);
                qagr.sql.add(cQUERY_UNION);
                qagr.sql.add(sql4);
                qagr.SQL.add(copy(sqlauxmain, insertpos +
length(subquery), length(sqlauxmain) - insertpos -
length(subquery) + 1));
                qagr.Prepare;
                qagr.Open;
              end;
            tdautoweb: begin
                qagr.close;
                qagr.sql.clear;
                insertpos := pos(subquery, sqlauxmain);
                qagr.SQL.add(copy(sqlauxmain, 1, insertpos - 1));
                qagr.sql.add(sql1);
                qagr.sql.add(cQUERY_UNION);
                qagr.sql.add(sql2);
                qagr.SQL.add(copy(sqlauxmain, insertpos +
length(subquery), length(sqlauxmain) - insertpos -
length(subquery) + 1));
                qagr.Prepare;
                qagr.Open;
              end;
            else begin
                TBdatos.abreq(qagr, sql1);
              end;
            end;
            if (tipo = webAuto) or (tipo = historico)
              or (tipo = tdRevisable) or (tipo = tdAutoWeb) then
            begin
              registros := 0;
              practicado := 0;
              ingresado := 0;

              while not qagr.Eof do
              begin
                practicado := practicado + qagr.fields[0].AsFloat;
                registros := registros + qagr.fields[1].AsInteger;
                ingresado := ingresado + qagr.fields[2].AsFloat;
                qagr.Next;
              end;

              limp_total.Caption := FormatFloat('#,##0.00',
practicado);
              lcount.caption := FloatToStr(registros);
```

```
        limp_ingresado.caption := FormatFloat('#,##0.00',
ingresado);
      end
      else
      begin
        limp_total.Caption := FormatFloat('#,##0.00',
qagr.fields[0].asFloat);
        lcount.caption := qagr.fields[1].asString;
        if tipo <> auto then
          limp_ingresado.caption := FormatFloat('#,##0.00',
qagr.fields[2].asFloat)
        else
          limp_ingresado.caption := '0';
      end;
      cdscon.Active := false;
      cdscon.Active := true;
      TBDatos.Abreq(qrycon, s);
    end;
  end;
end;
```

2. Al hacer doble click en un registro recuperado de la búsqueda se navega al detalle de operaciones de la tasa

```
procedure TFrmConsulAut.grdCon2DblClick(Sender: TObject);
var
  f: TFrmAutLiq;
  fn: TFrmNotDi;
  fnr: TFrmNotDi_Reb;
  ff: TFrmFracci;
  ffr: TFrmFracciReb;
  fhis: TFrmHistoricoOperacion;
  n28, tabla, tabla2: string;
  q: TQuery;
  s, s2, ug, ee, va, ni, ej, cl, co: string;
  N28F: string;
  EsMigrado: Boolean;
begin
  if cdscon.IsEmpty then exit;

  if brevisa
    then
    if tipo = tdRevisable
      then begin
      self.visible := false;
      try
        TFrmRevisa.CrearModal(stringreplace(cdscon.FieldByName('
n28').AsString, #32, '', [rfReplaceAll]));
      finally
        self.visible := true;
      end;
      exit;
    end;
```

```
ug := copy(cdscon.FieldByName('n28').AsString, 1, 6);
ee := copy(cdscon.FieldByName('n28').AsString, 8, 3);
va := copy(cdscon.FieldByName('n28').AsString, 12, 6);
ni := copy(cdscon.FieldByName('n28').AsString, 19, 5);
ej := copy(cdscon.FieldByName('n28').AsString, 25, 4);
cl := copy(cdscon.FieldByName('n28').AsString, 30, 3);
co := copy(cdscon.FieldByName('n28').AsString, 34, 1);
n28 := ug + ee + va + ni + ej + cl + co;

if tipo = tdRevisable then
  esMigrado := false
else
  EsMigrado := TFuncBd.EsN28Migrado(n28, n28F, true);
if bNotifiDirecta or bFracci or IngPrevio or bSelecc then
  if EsMigrado then Exit;


if IngPrevio then
begin
  sl.add(cdscon.FieldByName('N28').AsString);
      sl.add(cdscon.FieldByName('importe').AsString);
  sl.add(copy(cdscon.FieldByName('ESTADO').AsString, 1, 2));
  close;
  exit;
end;

if tipo = historico then
begin
  try
    fhis := TFrmHistoricoOperacion.create(self, n28);
    fhis.ShowModal;
  finally
    fhis.Free;
  end;
  exit;
end;


if (((StrToInt(cl) > 540) and (strtoint(cl) < 590))
  or ((StrToInt(cl) > 640) and (strtoint(cl) < 690)))
  and not devol then
begin
  if not (bNotifiDirecta or bOrigenRevisa) then
  begin
    minfo('Fraccionamiento sin detalles', '');
    exit;
  end;
end;

if bNotifiDirecta then
begin
  if tipo = liqui then
  begin
```

```
        fn := TFrmNotDi.createRev(self, n28);
        fn.ShowModal;
        fn.free;
      end
      else begin
        fnr := TFrmNotDi_Reb.createRev(self, n28);
        fnr.ShowModal;
        fnr.free;
      end;
      exit;
    end;

    if bFracci then
    begin

      if (proceso = untfrmfracci.fraccionar) and
(cdscon.FieldByname('Flag_Frac').AsString = 'A') then
      begin
        minfo('Esta deuda esta aplazada, entre por aplazamiento',
'');
        exit;
      end;
      if (proceso = untfrmfracci.aplazar) and
(cdscon.FieldByname('Flag_Frac').AsString = 'F') then
      begin
        minfo('Esta deuda esta fraccionada, entre por
fraccionamiento', '');
        exit;
      end;
      if (cdscon.FieldByName('F_NOTIFICACIÓN').asString = '') then
      begin
        minfo('La deuda debe estar notificada ', '');
        exit;
      end;
      if (Copy(cl, 1, 1) = '9') then
      begin
        minfo('No se puede aplazar o fraccionar una deuda
Extrapresupuestaria', '');
        exit;
      end;

      if tipo = recibo then
      begin
        ffr := TFrmfraccireb.create(self, n28, proceso);
        ffr.ShowModal;
        ffr.Free;
      end
      else
      begin
        ff := TFrmfracci.create(self, n28, proceso);
        ff.ShowModal;
        ff.Free;
      end;
      exit;
```

```
    end;

    if bselecc then
    begin
      close;
      exit;
    end;
    if tipo = recibo then
    begin
      Minfo('La pantalla Ver detalle recibo no está disponible',
'');
      exit;
    end;

    if tipo = liqui then
      tabla := 'FSLINEAS'
    else if tipo = auto then
      tabla := 'FSLINAUT'
    else if tipo = tiquet then
      tabla := 'FSLINTIQ'
    else if tipo = webLiqui then
      tabla := 'FSLINEAS'
    else if tipo = webAuto then
    begin
      tabla := 'FSLINAUTWEB';
      tabla2 := 'FSLINEAS';
    end else
      if tipo = tdAutoWeb then begin
        tabla := 'FSLINAUT';
        tabla2 := 'FSLINAUTWEB';
      end;

    TBDatos.CrearQ(q, nil);

    try
      if ((tipo = webAuto) or (tipo = tdAutoWeb)) then
      begin
        s := 'select cod_concepto, cod_hi, f_ini_hi from ' + tabla
+ ' where u_gest=' + TBdatos.com(ug) + ' and e_emis=' +
TBdatos.com(ee) +
          ' and valor=' + TBdatos.com(va) + ' and num_impreso=' +
TBdatos.com(ni) +
          ' and ejercicio=' + TBdatos.com(ej) + ' and
cla_est_deuda=' + TBdatos.com(cl) +
          ' and control=' + TBdatos.com(co);

        s2 := 'select cod_concepto, cod_hi, f_ini_hi from ' +
tabla2 + ' where u_gest=' + TBdatos.com(ug) + ' and e_emis=' +
TBdatos.com(ee) +
          ' and valor=' + TBdatos.com(va) + ' and num_impreso=' +
TBdatos.com(ni) +
          ' and ejercicio=' + TBdatos.com(ej) + ' and
cla_est_deuda=' + TBdatos.com(cl) +
          ' and control=' + TBdatos.com(co);
```

```
        s := s + cQUERY_UNION + s2;
        TBDatos.Abreq(q, s);
        n28Web := n28;

        if (not q.IsEmpty) then
        begin
          concepto := q.FieldByName('cod_concepto').AsString;
          hi := q.FieldByName('cod_hi').AsString;
          fecIni := q.FieldByName('F_INI_HI').AsDateTime;

          if tipo = tdAutoWeb then begin
            f := TFrmAutLiq.creaconsul(self, consulta, tipo, n28,
concepto);
            try
              f.showmodal;
            finally
              f.free;
            end;
          end;
        end
        else
        begin
          n28Web := '';
          Merror('Deuda sin líneas de detalle', '');
        end;
      end
      else
      begin
        s := 'select * from ' + tabla + ' where u_gest=' +
TBdatos.com(ug) + ' and e_emis=' + TBdatos.com(ee) +
          ' and valor=' + TBdatos.com(va) + ' and num_impreso=' +
TBdatos.com(ni) +
          ' and ejercicio=' + TBdatos.com(ej) + ' and
cla_est_deuda=' + TBdatos.com(cl) +
          ' and control=' + TBdatos.com(co);
        TBDatos.Abreq(q, s);

        if q.IsEmpty then
          Merror('Deuda sin líneas de detalle.', '')
        else
        begin
          if tipo = webLiqui then
          begin
            n28Web := n28;
            concepto := q.FieldByName('cod_concepto').AsString;
            hi := q.FieldByName('cod_hi').AsString;
            fecIni := q.FieldByName('F_INI_HI').AsDateTime;
          end
          else
          begin
            f := TFrmAutLiq.creaconsul(self, consulta, tipo,
n28,     q.FieldByName('cod_concepto').AsString);
            try
```

```
            f.showmodal;
          finally
            f.free;
          end;
        end;
      end;
    end;
  finally
    q.free;
    if ((tipo = webAuto) or (tipo = webLiqui)) then
      BitBtn1.Click;
  end;
end;
```

Con respecto a la segunda cuestión planteada:

"La segunda cuestión es que en la lista de Funcionalidades actuales de QUESTOR (página 15 del PPT), y en la relación de componentes de cada elemento funcional, termina en muchos de ellos con un "etc" o puntos suspensivos. ¿Es posible que nos deis una estimación del esfuerzo o al menos un listado de los otros elementos no están especificados y sin embargo habría que mirar y que están detrás de los "etc"? La cuestión es que esos puntos suspensivos provocan una incertidumbre en la valoración ya que desconocemos a qué se refiere."

Funcionalidades actuales de QUESTOR a migrar al nuevo entorno ANGULAR 5/NODE-JS/PLSQL ORACLE, Servicios PI Y NUEVOS DESARROLLOS.

*En letra cursiva las opciones de menú que sustituyen a los etc.*

- **<u>Mantenimiento de tablas maestras</u>** Formularios para la parametrización donde se basa la gestión y fórmulas para el cálculo de los importes de los conceptos de ingreso, así como el mantenimiento de la parametrización de la emisión remota de tasas vía API3.0. Como por ejemplo:
  - contadores,
  - beneficios fiscales,
  - conceptos de ingresos:
    - *conceptos de ingreso,* tablas de valores, tarifas, atributos, formulas, *actualizar conceptos ingreso.*
  - conceptos presupuestarios:
    - *mantenimiento conceptos presupuestarios, c. presup. por U.G.I., actualización de c. presupuestarios.*
  - *fijación de deuda*
  - interés de demora,
  - *modalidad de gestión*
  - tipos de IVA,
  - *estados de la deuda*
  - días festivos,
  - procedimientos (API 3.0):

- **<u>Mantenimiento de Entidades</u>**: Formularios para mantenimiento de tablas maestras para la parametrización, registro y administración de:

- contribuyentes,
- consejerías,
- centros directivos,
- organismos autónomos,
- unidades gestoras,
- entidades emisoras,
- *conceptos de ingreso por UGI*
- cajas de efectivos,
- entidades bancarias,
- *administraciones hacienda*
- *administración de usuarios:*
    - *grupos, menús, permisos, asignación masiva de grupos a usuarios, usuario – centro gestor.*

- mantenimiento cartas de pago
- *mantenimiento de ugi:*
    - *carga inicial, proceso inicio de ejercicio, tabla de equivalencias, proceso cambio de UGI.*

- **<u>Gestión de Padrones</u>**: Formularios para mantenimiento de tablas generales y formularios para la gestión y generación de padrones mediante:
    - tipos y mantenimiento de objetos,
    - mantenimiento de padrones,
    - domiciliación de recibos,
    - *generar listas de cobro*
    - *numeración de recibos*
    - *expedición listas de cobro*
    - impresión,
    - consulta de recibos,
    - notificación recibos
    - *carga de datos externos (I.V.)*
    - *carga de datos externos (Caza)*
    - *informe de ingreso*
    - *escuelas infantiles*

- *cotos de caza*
- *fraccionar/aplazar recibos*

En cuanto a la domiciliación de recibos, se deben abordar los desarrollos necesarios para que la aplicación QUESTOR haga uso de servicios ofrecidos por la aplicación ARECA para la tramitación completa de las domiciliaciones bancarias según la norma SEPA.

Esta tramitación actualmente se encuentra implementada de forma íntegra en la aplicación ARECA y es del ámbito de este proyecto el desarrollo de los servicios necesarios para hacerla reutilizable por la aplicación QUESTOR, de forma que el intercambio de información resulte automático, instantáneo y manteniendo la consistencia de ambos sistemas.

- **Presentación de Autoliquidaciones**: Formularios de:
  - presentación de autoliquidaciones emitidas por el sistema Questor:
    - *emitidas por sistema Questor (código de barras), emitidas por entidades emisoras (lectura PDF), emitidas por entidades emisoras (entrada manual), emitidas mediante preimpreso.*
  - consulta de autoliquidaciones presentadas

- **Aplazamiento. Fraccionamiento**. Formularios de emisión, consulta e impresión de aplazamientos y fraccionamientos:
  - gestión de fraccionamientos
  - gestión de aplazamientos
  - impresión aplazamientos/fraccionamientos
  - retrotraer aplazamientos/fraccionamientos

- **Gestión en caja de efectivo:** Formularios de generación de arqueos, regularizaciones...:
  - cobro de tickets,
  - cobro de autoliquidaciones,
  - cobros pendientes de arqueo,
  - arqueo: generar arqueo, consulta arqueos,

- libro de diario: listado de diario de caja, regularización de saldo de caja,
- *ingreso en cuenta restringida: generación de ingreso en cuenta restringida, consulta liquidaciones ingresos en cuenta*

- **Procedimiento de Revisión**: Formularios para gestionar las situaciones de una deuda, anulaciones especiales:
  - anulación por insolvencia,
  - prescripción,
  - cobro en especie.

- **Gestión de notificaciones:** Formularios para generación y gestión de notificaciones:
  - *Notificaciones por listas*
  - notificaciones directas,
  - consultas,
  - anuncio – edicto – Borm
  - grabar fecha notificación
  - gestión notificaciones: generar relaciones para notificar, anuncios, edictos, *mantenimiento de relaciones generadas, reclamación de relaciones entregadas*

- **Resúmenes Contables**: Formularios que recogen las operaciones contables, por periodos y centro gestor, de reconocidos, de ingresos e incidencias recaudatorias, agrupados por conceptos presupuestarios, para pasarlas a SIGEPAL para su contabilización:
  - gestión de resúmenes
  - tablas auxiliares
  - consulta resúmenes
  - estado de ejecución presupuesto
  - relación nominal para Sigepal
  - *centro de operaciones contables*
  - *chequeo aplicación de ingresos.*
- **Utilidades, Listados y Estadísticas**: Varios listados de carácter general para el usuario:
  - cambio de contraseña,

- cambio de Unidad Gestora,
- unidades gestoras,
- entidades emisoras,
- modalidades de gestión,
- beneficios fiscales,
- conceptos de ingreso,
- hechos imponibles,
- atributos,
- resumen económico.
- *preimpresos*
- *consulta por doc. 901*
- *nº de liquidaciones e importe recaudado por HI y por mes*
- *fondo de mejoras*
- *importe pendiente por NIF*
- *histórico de operaciones*

- **Devolución de ingresos indebidos**: Formularios para la gestión y tramitación de la devolución:
  - tramitar devolución,
  - tramitar devoluciones masivas,
  - consulta devoluciones,
  - envío a Sigepal,

- **Consultas generales**: Formularios de consulta de:
  - liquidaciones y autoliquidaciones presentadas,
  - ingresos por municipios,
  - operaciones contables.
  - BUR

- **Gestiones en Ejecutiva:** Formularios para generación y gestión de relaciones de deudas a comunicar a ARECA para su recaudación en ejecutiva:
  - generar relación de deudas: generar fichero…
  - anulación de deudas pasadas a ejecutiva,
  - cobros posteriores a la fecha fin de voluntaria,

- aplicar ingresos de ejecutiva: cargar fichero, aplicar ingresos…
- aplicar anulaciones de ejecutiva,
- aplicar retrotracciones de movimientos (anulaciones de fallidos, devoluciones de ingresos, etc...)
- *Simulación de generación de apremios.*

(Documento firmado electrónicamente al margen)

El Jefe de Unidad de Gestión Técnica

Fdo.: Francisco Javier Zapata Martínez