TABLA DE CONTENIDO

ANEXO II
CÓDIGO FUENTE DE LA APLICACIÓN

Este anexo presenta el código desarrollado en la implementación de la aplicación, en concordancia Capítulo IV "Creación de la Aplicación" de la Monografía.

1 Módulo: PASARELA.DLL.

Este componente fue desarrollado en Visual C++, debido fundamentalmente a las amplias facilidades que este lenguaje de programación brinda para la creación de Librerías de Enlace Dinámico.

1.1 Componente: PASARELA.CPP

```
//*****************************************************************************
//** PROPÓSITO: CONTIENE LA IMPLEMENTACIÓN DE LAS  ***
//**              CLASES Y LAS FUNCIONES EXPORTABLES   ***
//*****************************************************************************
#define DBNTWIN32
#include <stdio.h>
#include <string.h>
#include <windows.h>
#include <sqlfront.h>
#include <sqldb.h>
#include <iostream.h>
#include "pasarela.h"
#include "ExpPasarela.h"


//************************************************
//**  PUNTO DE ENTRADA A LA DLL     ***
//************************************************
BOOL WINAPI DllEntryPoint(HINSTANCE hDLL, DWORD dwReason, LPVOID Reserved)
{     switch (dwReason)
      {case DLL_PROCESS_ATTACH:
                  {break;
                  }
                  case DLL_PROCESS_DETACH:
                  {break;
                  }
      }
      return TRUE; }
```

```
//*************************************************************************
//** DECLARACION DE LAS VARIABLES GLOBALES       ***
//*************************************************************************
char Errores[512];                    //Errores de la conexión
GestionConexion Conexion;
GestionConsulta Consulta;
GestionTransaccionSQL Transaccion;
GestionModificacion Modificador;

//*************************************************************************
//**  CONSTRUCTOR DE LA CLASE    GestionConexion    ***
//*************************************************************************
GestionConexion::GestionConexion()
{   strcpy(Login,"Anonimo");
      strcpy(Password,"Anonomo");
      strcpy(Servidor,"Anonimo");
}

//*****************************************************************************
//**  FUNCION CONECTAR                                                       ***
//**  CLASE RELACIONADA: GESTIONCONEXION                                     ***
//**  PROPOSITO: ESTABLECER UNA CONEXION CON EL                              ***
//**                          SQL SERVER.                                    ***
//*****************************************************************************
char*    GestionConexion::conectar(char*    unLogin,char*    unPassword,char*    unServidor,    char*
unBuffer_Datos)

{     strcpy(Errores,"");
      dberrhandle (err_handler);
      dbmsghandle (msg_handler);
      strcpy(Estado," ");
      strcpy(Login,unLogin);
      strcpy(Password,unPassword);
      strcpy(Servidor,unServidor);
      strcpy(Buffer_Datos,unBuffer_Datos);

   dbinit ();
   Usuario = dblogin ();
   DBSETLUSER (Usuario, unLogin);
   DBSETLPWD (Usuario, unPassword);
   DBSETLAPP (Usuario, "Arcview");
      DBSETLTIME (Usuario,TIMEOUT_IGNORE);//Ignora el tiempo out
      dbsetlogintime(15); //Espera 15 segundos para dar time out
      strcpy(Estado,"Error en la conexion !!\n");

      if (!((MiConexion = dbopen (Usuario, unServidor)) == NULL))//conexion exitosa
      {           strcpy(Estado,"Conexion Exitosa!!\n");

                  //nombre de la base de datos enlazada
                  strcpy(Base_Datos,dbname(MiConexion));
                  strcat(Estado,Base_Datos);
                  strcat(Estado,"\n");
   }
      File_Buffer_Datos = fopen(Buffer_Datos, "w");
   fprintf(File_Buffer_Datos,Estado);
      fclose(File_Buffer_Datos);
      return Errores; // exit program

}
```

```
//*************************************************************************
//**  FUNCION DESCONECTAR                                                ***
//**  CLASE RELACIONADA: GESTIONCONEXION                                 ***
//**  PROPOSITO: FINALIZAR UNA CONEXION CON EL                           ***
//**                          SQL SERVER.                                ***
//*************************************************************************
char* GestionConexion::desconectar(void)

{ strcpy(Errores,"");
  dberrhandle (err_handler);
  dbmsghandle (msg_handler);
  dbexit ();
  strcat(Errores,"Desconectado del servidor");
  return Errores;

}

//*************************************************************************
//**  FUNCION ACCESO_BD                                                  ***
//**  CLASE RELACIONADA: GESTIONCONEXION                                 ***
//**  PROPOSITO: ESTABLECER LAS BASES DE DATOS                           ***
//**                          A LAS QUE SE TIENE ACCESO                  ***
//*************************************************************************
char* GestionConexion::Acceso_BD(void)

{ char Bases_Datos[512];
  int contador=0;
  strcpy(Errores,"");
  dberrhandle (err_handler);
  dbmsghandle (msg_handler);

  //Se cambia de contexto a la base de datos master
  dbuse(MiConexion,"master");
  dbcmd(MiConexion, (char *)"select name from sysdatabases where has_dbaccess(name)=1");

   //Ejecuta la consulta
  dbsqlexec (MiConexion);
  if (dbresults (MiConexion)== SUCCEED)
    {     // Une columnas a variables del programa.
                dbbind (MiConexion, 1, NTBSTRINGBIND, (DBINT) 0, (unsigned char *)Bases_Datos);
                File_Buffer_Datos = fopen(Buffer_Datos, "w");   /* Abrir archivo para escritudfra */
                // Retribuye  los resultados
       while (dbnextrow (MiConexion) != NO_MORE_ROWS)
       {          contador++;
                  //El acesso debe ser diferente a las bases de datos predeterminadas.
       if         (strcmp(Bases_Datos,"master")          &&          strcmp(Bases_Datos,"tempdb")          &&
strcmp(Bases_Datos,"msdb"))
                          {
                     fprintf(File_Buffer_Datos, Bases_Datos);
                          fprintf(File_Buffer_Datos,"\n");
                          }//if
                }
                     fclose(File_Buffer_Datos);    //Cerrar el archivo antes de terminar el programa

  } //del if

    dbuse(MiConexion,Base_Datos);
      return Errores;
}
```

```
//**********************************************************************
//**  FUNCION GET_TIPO_USUARIO                                        ***
//**  CLASE RELACIONADA: GESTIONCONEXION                              ***
//**  PROPOSITO: ESTABLECER EL TIPO DE USUARIO                        ***
//**                          CONECTADO    .                          ***
//**********************************************************************
char* GestionConexion::get_tipo_usuario(void)

{
      char Rol[512],LoginM[512];
      char Miembro[512],MiembroM[512];
      //Inicialzacion Variables
      strcpy(Errores,"");
      dberrhandle (err_handler);
      dbmsghandle (msg_handler);


   // Primero coloca el comando dentro del bufer.
   dbcmd(MiConexion,(char*)"EXEC sp_helprolemember");
   // Envia el comando a SQL Server inicia la ejecucion.
   dbsqlexec (MiConexion);

   // Procesa los resultados
   if (dbresults (MiConexion) == SUCCEED)
   {
       // Une columnas a variables del programa.
       dbbind (MiConexion,        1, NTBSTRINGBIND, (DBINT) 0, (unsigned char *)Rol);
                dbbind (MiConexion,         2, NTBSTRINGBIND, (DBINT) 0, (unsigned char *)Miembro);


                File_Buffer_Datos = fopen(Conexion.GetBuffer_Datos(), "w");

                        while (dbnextrow (MiConexion) != NO_MORE_ROWS)
                        {   //ESTABLECE LAS AMBAS CADENAS EN MAYUSCULAS
                                strcpy(MiembroM,strlwr(Miembro));
                                strcpy(LoginM,strlwr(Login));

                                //EXAMINA LOS PRIVILEGIOS
                                if (!strcmp(Miembro,Login))
                                {
                                  if(!strcmp(Rol,"db_owner"))
fprintf(File_Buffer_Datos,"ADMINISTRADOR");
                                  if(!strcmp(Rol,"db_datareader"))      fprintf(File_Buffer_Datos,"MONITOR
SIG");
                                }
                        }
                   fclose(File_Buffer_Datos);    /* Cerrar el archivo antes de terminar el programa */
                        return "Consulta exitosa";
      }
      return Errores;
}
```

```
//*************************************************************************
//**  FUNCION CAMBIAR_BD                                                  ***
//**  CLASE RELACIONADA: GESTIONCONEXION                                  ***
//**  PROPOSITO: CAMBIAR LA BASES DE DATOS                               ***
//**                          CONECTADA                                   ***
//*************************************************************************
char* GestionConexion::cambiar_BD(char* data_base)

{ strcpy(Errores,"");
  dberrhandle (err_handler);
  dbmsghandle (msg_handler);
  dbuse(MiConexion,data_base);
  strcpy(Base_Datos,data_base);
  strcat(Errores,"Conexion Exitosa");
  return Errores;
 }

//*************************************************************************
//**  FUNCIONES DE MANEJO DE ERROR EN LA CONEXION   ***
//**  CLASE RELACIONADA: GESTION CONEXION                                 ***
//*************************************************************************
int err_handler (PDBPROCESS MiConexion, INT severity,
    INT dberr, INT oserr, LPCSTR dberrstr, LPCSTR oserrstr)
{               strcat(Errores,"Error de conexion: ");
                strcat(Errores,dberrstr);
                strcat(Errores,"\n");
                return(0);
}

int msg_handler (PDBPROCESS MiConexion, DBINT msgno, INT msgstate,
      INT severity, LPCSTR msgtext, LPCSTR server,
    LPCSTR procedure, DBUSMALLINT line)
{       strcat(Errores,"Mensaje de Sql Server: ");
                strcat(Errores,msgtext);
                strcat(Errores,"\n");
                return(0);
}


//*************************************************************************
//**              CONSTRUCTOR          GestionConsulta                 ***
//*************************************************************************
GestionConsulta::GestionConsulta()
{
      strcpy(Tabla,"");
      NoCampos=0;
      strcpy(Campos_Elegidos,"");
      strcpy(Datos,"");

}
```

```
//*************************************************************************
//**  FUNCION CONSULTA                                                   ***
//**  CLASE RELACIONADA: GESTIONCONSULTA                                 ***
//**  PROPOSITO: REALIZAR UNA CONSULTA DE DATOS AL***
//**                        SQL SERVER.                                  ***
//*************************************************************************
char*  GestionConsulta::consulta (char *unaTabla, int unNoCampos,char* unosCampos_Elegidos)

{   int contador;

      //Inicializacion variables
      strcpy(Tabla,unaTabla);
      NoCampos=unNoCampos;
      strcpy(Campos_Elegidos,unosCampos_Elegidos);
      strcpy(Errores,"");
      strcpy(Datos,"");

      //Errores
      dberrhandle (err_handler);
   dbmsghandle (msg_handler);

      // Primero coloca el comando dentro del bufer.
   dbcmd (Conexion.GetMiConexion(), "SELECT ");
      dbcmd (Conexion.GetMiConexion(), (char *)Campos_Elegidos);
      dbcmd (Conexion.GetMiConexion(), " FROM ");
   dbcmd (Conexion.GetMiConexion(), (char *)Tabla);

   // Envia el comando a SQL Server inicia la ejecucion.
   dbsqlexec (Conexion.GetMiConexion());

   // Procesa los resultados
   if (dbresults (Conexion.GetMiConexion())== SUCCEED)
   {
                //Limpia la variable de los nombres delos campos
                strcpy(Campos_Elegidos,"");

       // Une columnas a variables del programa.
                for (contador=1;contador < NoCampos;contador++)
                {   dbbind (Conexion.GetMiConexion(), (DBINT)contador, NTBSTRINGBIND,  0, (unsigned
char *)Campos[contador]);

                        //NOMBRE DE LA COLUMNA
                        strcpy(Nombre_Columna,dbcolname(Conexion.GetMiConexion(), contador));
                        strcat(Campos_Elegidos,Nombre_Columna);
                   strcat(Campos_Elegidos,"®");

          //examina los tipos de datos no soportados
                   strcpy(Tipo_Dato,dbprtype(dbcoltype(Conexion.GetMiConexion(), contador)));

                   if (!(strcmp(Tipo_Dato,"image")) || !(strcmp(Tipo_Dato,"text")))
                        {          dbcancel(Conexion.GetMiConexion());
                                   strcat(Nombre_Columna," tiene un tipo de dato ");
                              strcat(Nombre_Columna,Tipo_Dato);
                                   strcat(Nombre_Columna," no soportado.");
                                   return Nombre_Columna;
                        }
                }


                dbbind  (Conexion.GetMiConexion(),  contador,  NTBSTRINGBIND,  0,  (unsigned
char*)Campos[NoCampos]);
```

```
                strcpy(Nombre_Columna,dbcolname(Conexion.GetMiConexion(), NoCampos));

        strcat(Campos_Elegidos,Nombre_Columna);

                //Para evitar la coma
                strcpy(Tipo_Dato,dbprtype(dbcoltype(Conexion.GetMiConexion(), NoCampos)));

                if (!(strcmp(Tipo_Dato,"image")) || !(strcmp(Tipo_Dato,"text")))
                        {       dbcancel(Conexion.GetMiConexion());
                                strcat(Nombre_Columna," tiene un tipo de dato ");
                           strcat(Nombre_Columna,Tipo_Dato);
                                strcat(Nombre_Columna," no soportado.");
                                return Nombre_Columna;
                        }

                File_Buffer_Datos = fopen(Conexion.GetBuffer_Datos(), "w");      /* Abrir archivo para
escritudfra */
           fprintf(File_Buffer_Datos, Campos_Elegidos);
                fprintf(File_Buffer_Datos,"\n");

                // Retribuye  los resultados
       while (dbnextrow (Conexion.GetMiConexion()) != NO_MORE_ROWS)
       {
                        for (contador=1;contador < NoCampos;contador++)
                        { strcat(Datos,Campos[contador]);
                          strcat(Datos,"®");
                        }
                        strcat(Datos,Campos[NoCampos]);
                        strcat(Datos,"\n");
                        fprintf(File_Buffer_Datos,Datos);
                        strcpy(Datos,"");
                        }
                   fclose(File_Buffer_Datos);    //Cerrar el archivo antes de terminar el programa
                        return "Consulta exitosa";
      }
      return Errores;
}


//*************************************************************************
//**  FUNCION CONSULTA_TABLAS                                        ***
//**  CLASE RELACIONADA: GESTIONCONSULTA                             ***
//**  PROPOSITO: REALIZAR UNA CONSULTA DE TABLAS ***
//**                        DISPONBILES EN UNA B.D.                  ***
//*************************************************************************
char*  GestionConsulta::Consulta_tablas (void)

{   char Tablas[512];

      strcpy(Datos,"");
      strcpy(Errores,"");
      dberrhandle (err_handler);
    dbmsghandle (msg_handler);


    // Primero coloca el comando dentro del bufer.
    dbcmd (Conexion.GetMiConexion(), (char *)"EXEC sp_tables @table_type = \"\'Table\'\"");
    // Envia el comando a SQL Server inicia la ejecucion.
    dbsqlexec (Conexion.GetMiConexion());

    // Procesa los resultados
```

```
   if (dbresults (Conexion.GetMiConexion()) == SUCCEED)
   {
      // Une columnas a variables del programa.
      dbbind (Conexion.GetMiConexion(),  3,  NTBSTRINGBIND,  (DBINT)  0,  (unsigned  char
*)Tablas);

            File_Buffer_Datos = fopen(Conexion.GetBuffer_Datos(), "w");

            while (dbnextrow (Conexion.GetMiConexion()) != NO_MORE_ROWS)
                  {  strcat(Datos,Tablas);
                        strcat(Datos,"\n");
                        fprintf(File_Buffer_Datos,Datos);
                        strcpy(Datos,"");
                        }
                        fclose(File_Buffer_Datos);    /* Cerrar el archivo antes de terminar el pr*/
                        return "Consulta exitosa";
                        }
   // Cierra la conexion a SQL Server.
     return Errores;
}


//*************************************************************************
//**  FUNCION CONSULTA_CAMPOS                                            ***
//**  CLASE RELACIONADA: GESTIONCONSULTA                                 ***
//**  PROPOSITO: REALIZAR UNA CONSULTA DE CAMPOS                         ***
//**                DISPONBILES EN UNA TABLA.                            ***
//*************************************************************************
char*  GestionConsulta::Consulta_campos(char* unaTabla)

{  char Campos_Disponibles[512];
     //Inicialzacion Variables
     strcpy(Tabla,unaTabla);
     strcpy(Datos,"");
     strcpy(Errores,"");
     dberrhandle (err_handler);
   dbmsghandle (msg_handler);


   // Primero coloca el comando dentro del bufer.
   dbcmd(Conexion.GetMiConexion(),(char*)"EXEC sp_columns @table_name = \'");
     dbcmd(Conexion.GetMiConexion(),(char*)Tabla);
     dbcmd(Conexion.GetMiConexion(),(char*)"\'");
   // Envia el comando a SQL Server inicia la ejecucion.
   dbsqlexec (Conexion.GetMiConexion());

   // Procesa los resultados
   if (dbresults (Conexion.GetMiConexion()) == SUCCEED)
   {
      // Une columnas a variables del programa.
      dbbind (Conexion.GetMiConexion(),  4,  NTBSTRINGBIND,  (DBINT)  0,  (unsigned  char
*)Campos_Disponibles);

            File_Buffer_Datos = fopen(Conexion.GetBuffer_Datos(), "w");

                  while (dbnextrow (Conexion.GetMiConexion()) != NO_MORE_ROWS)
                  {  strcat(Datos,Campos_Disponibles);
                        strcat(Datos,"\n");
                        fprintf(File_Buffer_Datos,Datos);
                        strcpy(Datos,"");
                        }
```

```
                                fclose(File_Buffer_Datos);    /* Cerrar el archivo antes de terminar el pr*/
                                return "Consulta exitosa";
                                }
   // Cierra la conexion a SQL Server.
      return Errores;
}


//*************************************************************************
//**              CONSTRUCTOR            GestionTransaccion     ***
//*************************************************************************
GestionTransaccionSQL::GestionTransaccionSQL()
{
      strcpy(Datos,"");
      strcpy(Sentencia_Sql,"");
      strcpy(Campos_Elegidos,"");
   NoCampos_Retornados=0;
}

//*****************************************************************************
//**  FUNCION TRANSACCION_SQL                                                ***
//**  CLASE RELACIONADA: GESTIONTRANSACCIONSQL                               ***
//**  PROPOSITO: REALIZAR UNA CONSULTA SQL                                   ***
//**                          AL SGBD.                                       ***
//*****************************************************************************
char*  GestionTransaccionSQL::Transaccion_Sql (char* query)

{   int contador;
      //Variables iniciales
      strcpy(Datos,"");
      strcpy(Campos_Elegidos,"");
      strcpy(Sentencia_Sql,query);
      strcpy(Errores,"");

      dberrhandle (err_handler);
   dbmsghandle (msg_handler);

   // Primero coloca el comando dentro del bufer.
   dbcmd (Conexion.GetMiConexion(), (char *)Sentencia_Sql);

   // Envia el comando a SQL Server inicia la ejecucion.
   dbsqlexec (Conexion.GetMiConexion());

   // Procesa los resultados
   if (dbresults (Conexion.GetMiConexion()) == SUCCEED)
   {   // Numero de columnas retribuidas
                NoCampos_Retornados = dbnumcols(Conexion.GetMiConexion());//Si estamos haciendo
una actualizacion
                if (NoCampos_Retornados==0)
                { return "Transaccion exitosa";}

                // Une columnas a variables del programa.
      for (contador=1;contador < NoCampos_Retornados;contador++)
                { dbbind (Conexion.GetMiConexion(), contador, NTBSTRINGBIND,     0, (unsigned
char*)Campos[contador]);
                        strcpy(Nombre_Columna,dbcolname(Conexion.GetMiConexion(), contador));
                        strcat(Campos_Elegidos,Nombre_Columna);
                   strcat(Campos_Elegidos,"®");

          //examina los tipos de datos no soportados
```

```
                strcpy(Tipo_Dato,dbprtype(dbcoltype(Conexion.GetMiConexion(), contador)));

                if (!(strcmp(Tipo_Dato,"image")) || !(strcmp(Tipo_Dato,"text")))
                        {       dbcancel(Conexion.GetMiConexion());
                                strcat(Nombre_Columna," tiene un tipo de dato ");
                           strcat(Nombre_Columna,Tipo_Dato);
                                strcat(Nombre_Columna," no soportado.");
                                return Nombre_Columna;
                        }


            }

            dbbind (Conexion.GetMiConexion(), contador, NTBSTRINGBIND, 0, (unsigned
char*)Campos[NoCampos_Retornados]);
            strcpy(Nombre_Columna,dbcolname(Conexion.GetMiConexion(),
NoCampos_Retornados));
      strcat(Campos_Elegidos,Nombre_Columna);

            //Para evitar la coma
            strcpy(Tipo_Dato,dbprtype(dbcoltype(Conexion.GetMiConexion(),
NoCampos_Retornados)));
            if (!(strcmp(Tipo_Dato,"image")) || !(strcmp(Tipo_Dato,"text")))
                        {       dbcancel(Conexion.GetMiConexion());
                                strcat(Nombre_Columna," tiene un tipo de dato ");
                           strcat(Nombre_Columna,Tipo_Dato);
                                strcat(Nombre_Columna," no soportado.");
                                return Nombre_Columna;
            }

            File_Buffer_Datos = fopen(Conexion.GetBuffer_Datos(), "w");
      fprintf(File_Buffer_Datos, Campos_Elegidos);
            fprintf(File_Buffer_Datos,"\n");
            // Retribuye  los resultados
      while (dbnextrow (Conexion.GetMiConexion()) != NO_MORE_ROWS)
      {
                        for (contador=1;contador < NoCampos_Retornados;contador++)
                        { strcat(Datos,Campos[contador]);
                          strcat(Datos,"®");
                        }
                        strcat(Datos,Campos[NoCampos_Retornados]);
                        strcat(Datos,"\n");
                        fprintf(File_Buffer_Datos,Datos);
                        strcpy(Datos,"");
                        }
               fclose(File_Buffer_Datos);    /* Cerrar el archivo antes de terminar el programa */
                        return "Consulta exitosa";
      }
      return Errores;
}
```

```
//*************************************************************************
//**              CONSTRUCTOR  GestionModificacion          ***
//*************************************************************************
GestionModificacion::GestionModificacion()
{
      strcpy(Nombre_Tabla,"");
    strcpy(Campo_Actualizar,"");
      strcpy(Campo_de_Referencia,"");
      strcpy(Valor_Campo_de_Referencia,"");
      strcpy(Valor_Actualizar,"");
}

//*****************************************************************************
//**  FUNCION ACTUALIZAR                                                     ***
//**  CLASE RELACIONADA: GESTIONMODIFICACION                                 ***
//**  PROPOSITO:REALIZAR ACTUALIZACION DE LAS TABLAS                         ***
//**                          DISPONBILES EN UNA B.D.                        ***
//*****************************************************************************
char*       GestionModificacion::actualizar    (char*    tabla,char*    campo_Act,char*    campo_Ref,char*
Val_Cam_Ref,char* Val_Act)

{   strcpy(Nombre_Tabla,tabla);
    strcpy(Campo_Actualizar,campo_Act);
      strcpy(Campo_de_Referencia,campo_Ref);
      strcpy(Valor_Campo_de_Referencia,Val_Cam_Ref);
      strcpy(Valor_Actualizar,Val_Act);
      strcpy(Errores," ");
      dberrhandle (err_handler);
    dbmsghandle (msg_handler);

    // Primero coloca el comando dentro del bufer.
      dbcmd (Conexion.GetMiConexion(), " UPDATE ");
      dbcmd (Conexion.GetMiConexion(), Nombre_Tabla) ;
      dbcmd (Conexion.GetMiConexion(), " SET ");
      dbcmd (Conexion.GetMiConexion(), Campo_Actualizar);
      dbcmd (Conexion.GetMiConexion(), " ='");
      dbcmd (Conexion.GetMiConexion(), Valor_Actualizar);
      dbcmd (Conexion.GetMiConexion(), "' WHERE ");
      dbcmd (Conexion.GetMiConexion(), Campo_de_Referencia);
      dbcmd (Conexion.GetMiConexion(), " ='");
      dbcmd (Conexion.GetMiConexion(), Valor_Campo_de_Referencia);
      dbcmd (Conexion.GetMiConexion(), "'");


    // Envia el comando a SQL Server inicia la ejecucion.
    dbsqlexec (Conexion.GetMiConexion());
        // Cierra la conexion con SQL Server.
    if (dbresults (Conexion.GetMiConexion()) == SUCCEED)
                return "Actualizacion Exitosa";
      return Errores;
}
```

```
//*****************************************************************************
//**                                                                        ****
//**  IMPLEMENTACION DE LAS FUNCIONES EXPORTABLES                            ***
//**                                                                         ***
//*****************************************************************************
char *CONECTAR(char* unLogin,char* unPassword,char* unServidor, char* unBuffer_Datos)
{   return Conexion.conectar(unLogin,unPassword,unServidor,unBuffer_Datos);
}
char *DESCONECTAR(void)
{     return Conexion.desconectar();
}
char* CONSULTA(char* tabla, int NoCampos, char* camposelegidos)
{   return Consulta.consulta(tabla, NoCampos, camposelegidos);
}
char* CONSULTA_TABLAS (void)
{     return Consulta.Consulta_tablas();
}
char *CONSULTA_CAMPOS(char* unaTabla)
{     return Consulta.Consulta_campos(unaTabla);
}
char *TRANSACCION_SQL (char* query)
{     return Transaccion.Transaccion_Sql(query);
}
char* ACTUALIZAR(char* tabla,char* campo_Act,char* campo_Ref,char* Val_Cam_Ref,char* Val_Act)
{
      return Modificador.actualizar(tabla,campo_Act,campo_Ref,Val_Cam_Ref,Val_Act);
}
char* ACCESO_BD(void)
{
      return Conexion.Acceso_BD();
}
char* CAMBIAR_BD(char* data_base)
{
      return Conexion.cambiar_BD(data_base);
}
char* GET_TIPO_USUARIO(void)
{
      return Conexion.get_tipo_usuario();
}
```

1.2 Componente: EXPPASARELA.H

```
//*****************************************************************************
//**  DECLARACION DE LAS FUNCIONES EXPORTABLES     ***
//*****************************************************************************
#define EXPORTS extern "C" __declspec(dllexport)
EXPORTS char* CONECTAR(char* unLogin,char* unPassword,char* unServidor, char*
unBuffer_Datos);
EXPORTS char* DESCONECTAR(void);
EXPORTS char* CONSULTA(char* tabla, int NoCampos, char* camposelegidos);
EXPORTS char* CONSULTA_TABLAS(void);
EXPORTS char* CONSULTA_CAMPOS(char* unaTabla);
EXPORTS char* TRANSACCION_SQL (char* query);
EXPORTS char* ACTUALIZAR(char* tabla,char* campo_Act,char* campo_Ref,char*
Val_Cam_Ref,char* Val_Act);
EXPORTS char* ACCESO_BD(void);
EXPORTS char* CAMBIAR_BD(char* data_base);
EXPORTS char* GET_TIPO_USUARIO(void);
```

1.3 Componente: PASARELA.H

```
//*********************************************************************
//**  PROPOSITO :DECLARACION DE LAS CLASES       ***
//*********************************************************************
#include <windows.h>
#include <sqlfront.h>
#include <sqldb.h>


//*****************************************************
//**  CLASE: GESTIONCONEXION              ***
//*****************************************************
class GestionConexion
{
private:
   char Login[512];
      char Servidor[512];
      char Password[512];
      PLOGINREC   Usuario; //Contiene la informaciòn del usuario que se conecta
      char Estado[512];
      FILE       *File_Buffer_Datos;
      char Buffer_Datos[512];
      PDBPROCESS  MiConexion;
      char Base_Datos[512];


public:

      GestionConexion();//Constructor

      // Envia mensajes de error al manejador de mensajes.
      friend int err_handler(PDBPROCESS, INT, INT, INT, LPCSTR, LPCSTR);
      friend int msg_handler(PDBPROCESS, DBINT, INT, INT, LPCSTR, LPCSTR,
            LPCSTR, DBUSMALLINT);
      friend class GestionConsulta;

      char* Acceso_BD(void);
      char* cambiar_BD(char* data_base);
      char* desconectar(void);
      char* conectar(char* unLogin,char* unPassword,char* unServidor, char* unBuffer_Datos);
      char* get_tipo_usuario(void);
};

//*****************************************************
//**  CLASE: GESTIONCONSULTA              ***
//*****************************************************
class GestionConsulta
{
private:
      FILE*     File_Buffer_Datos;
      char      Campos[100][512];
      char      Datos[512];
      char      Tabla[512];
      int                  NoCampos;
      char      Campos_Elegidos[512];
      char Tipo_Dato[512];
      char Nombre_Columna[512];
```

```
public:
      GestionConsulta();//Constructor;
      char* Consulta_tablas (void);
      char* Consulta_campos (char* unaTabla);
      char* consulta (char *unaTabla, int unNoCampos,char* unosCampos_Elegidos);
      friend int err_handler(PDBPROCESS, INT, INT, INT, LPCSTR, LPCSTR);
      friend int msg_handler(PDBPROCESS, DBINT, INT, INT, LPCSTR, LPCSTR,
            LPCSTR, DBUSMALLINT);
};

//*******************************************************
//**  CLASE: GESTIONTRANSACCIONSQL   ***
//*******************************************************
class GestionTransaccionSQL
{
private:
      FILE*     File_Buffer_Datos;
      char Sentencia_Sql[512];
      char Campos[100][512];
      char Campos_Elegidos[512];
   int  NoCampos_Retornados;
      char Datos[512];
      char Tipo_Dato[512];
      char Nombre_Columna[512];

public:
      GestionTransaccionSQL(); //Constructor
      char* Transaccion_Sql (char* query);
      friend int err_handler(PDBPROCESS, INT, INT, INT, LPCSTR, LPCSTR);
      friend int msg_handler(PDBPROCESS, DBINT, INT, INT, LPCSTR, LPCSTR,
            LPCSTR, DBUSMALLINT);

};

//*******************************************************
//**  CLASE: GESTIONCONMODIFICACION   ***
//*******************************************************
class GestionModificacion
{

private:
      char Nombre_Tabla[512];
   char Campo_Actualizar[512];
      char Campo_de_Referencia[512];
      char Valor_Campo_de_Referencia[512];
      char Valor_Actualizar[512];

public:
      GestionModificacion();//Constructor
      char* actualizar (char* tabla,char* campo_Act,char* campo_Ref,char* Val_Cam_Ref,char* Val_Act);
      friend int err_handler(PDBPROCESS, INT, INT, INT, LPCSTR, LPCSTR);
      friend int msg_handler(PDBPROCESS, DBINT, INT, INT, LPCSTR, LPCSTR,
            LPCSTR, DBUSMALLINT);
};
```

2 Módulo: PASARELA.AVX

PASARELA.AVX es una extensión del SIG Arcview GIS 3.2, desarrollada en Avenue que es el lenguaje nativo de dicho Sistema. Se puede contar con sus funcionalidades adicionándola a un proyecto Arcview, lo cual la hace utilizable en cualquier aplicación SIG.

Debido a que Avenue, aunque es un lenguaje orientado a objetos, no permite la creación de nuevas clases, éstas se implementan en Scripts independientes.

2.1 SCRIPTS:

2.1.1 Script: GestionConexion.ave

```
'*************************************************************************************************
'*** PROPOSITO: DECLARACIÓN DE LA CLASE GESTION_CONEXION     **
'***           ENCARGADA DE TODO LO REFERENTE A LA CONEXION        **
'*************************************************************************************************

'**TOMA EL METODO QUE SE SOLICITA
Metodo = self.getName

'*************************************************************
'***                  METODO CONECTAR                  **
'*************************************************************
if (Metodo="btn_Conectar") then
'**ENLAZA LAS VARIABLES DE LA INTERFAZ DE CONEXION
Dialogo_Conexion = av.GetProject.FindDialog("IS_Conexion")
user = Dialogo_Conexion.FindByName("txl_Login")
passwd = Dialogo_Conexion.FindByName("txl_Password")
serv = Dialogo_Conexion.FindByName("txl_Servidor")

'**VARIABLES DE LA CONEXION
_Login= user.GetText
Password= passwd.GetText
_Servidor=serv.GetText

'**ESTABLECE LA RUTA DONDE SE PONDRA EL BUFER_DATIOS
'**ESTA VARIABLE SERA ENVIADA A LA DLL COMO _Ruta
Ruta = System.GetEnvVar("AVHOME")
_Ruta=Ruta+"\Bin32\Buffer_Datos.txt" '**Direccion del Buffer_Datos Buffer_Datos

'**ELAZA LAS DLL
_DLL = FileName.FindinSystemSearchPath("Pasarela.dll")
'**REVISA LA EXISTENCIA DE LA DLL
if (_DLL =nil) then
 MsgBox.Error("No existen las DLL requeridas. Revise la instalacion del Script.","Error")
 Exit
end

'**EJECUTA LA CONEXION
_PasarelaDLL = DLL.Make(_DLL)
```

```
Conectar                                                                 =
DLLProc.Make(_PasarelaDLL,"CONECTAR",#DLLPROC_TYPE_STR,{#DLLPROC_TYPE_STR,#DLLP
ROC_TYPE_STR,#DLLPROC_TYPE_STR,#DLLPROC_TYPE_STR})
Resultado = Conectar.Call({_Login,Password,_Servidor,_Ruta})
MsgBox.info (Resultado.asString,"")
'**TRES OPORTUNIDADES DE CONEXION
'**HAY QUE ANALIZAR CUANTAS VECES LO PERMITE
Buffer_Datos = Filename.Make(_Ruta)
_Intentos=_Intentos+1
theLineFile = LineFile.Make(Buffer_Datos, #FILE_PERM_READ)
Datos = theLineFile.ReadELT

if (_Intentos = 3) then
  Dialogo_Conexion.Close
  Exit
end

'**SE REALIZA LA CONEXION Y GUARDAMOS EL NOMBRE DE LA BASE DE DATOS.
if (Datos= "Conexion Exitosa!!") then
 rec = 2
 '**FIJA LA BASE DE DATOS
 _BaseDatosActual = theLineFile.ReadELT

 '**ESTABLECEMOS LA HORA INICAL DE CONEXION
 Date.SetDefFormat ("d, h:m:s AMPM")
 _Hora_InicioConexion = Date.Now


 '**CONSULTA EL TIPO DE USUARIO
  GET_TIPO_USUARIO                                                       =
DLLProc.Make(_PasarelaDLL,"GET_TIPO_USUARIO",#DLLPROC_TYPE_STR,{})
  Resultado = GET_TIPO_USUARIO.Call({})
  Buffer_Datos = Filename.Make(_Ruta)
  theLineFile = LineFile.Make(Buffer_Datos, #FILE_PERM_READ)
  _Tipo_Usuario = theLineFile.ReadELT


 '**HABILITA LOS BOTONES
 av.Run("Gestion_Conexion.Habilitar_Botones",nil)

 '**CONSULTA LAS TABLAS DISPONIBLES EN LA BASE DE DATOS
  av.Run("GestionConsultas.Consulta_tablas",nil)

 Dialogo_Conexion.Close

else
   Dialogo_Conexion.Close
   Dialogo_Conexion.Open
   Exit
end

end '**DEL METODO
```

```
'*************************************************************************
'***        METODO CERRAR DIALOGO ISCONEXION           **
'*************************************************************************
if (Metodo="btn_Cancelar") then
  Dialogo_Conexion = av.GetProject.FindDialog("IS_Conexion")
  Dialogo_Conexion.Close
end


'*************************************************************************
'***        METODO MINIMIZAR DIALOGO ISCONEXION         **
'*************************************************************************
if (Metodo="btn_Minimizar") then
  Dialogo_Conexion = av.GetProject.FindDialog("IS_Conexion")
  Dialogo_Conexion.Minimize
end



'*****************************************************************************************
'***    METODO CERRAR DIALOGO ISINFORMACION_CONEXION      **
'*****************************************************************************************
if (Metodo="btn_AceptarIC") then
  Dialogo = av.GetProject.FindDialog("IS_Información_Conexión")
  Dialogo.Close
end

'*****************************************************************************************
'***    METODO MINIMIZAR DIALOGO ISINFORMACION_CONEXION      **
'*****************************************************************************************
if (Metodo="btn_MinimizarIC") then
  Dialogo = av.GetProject.FindDialog("IS_Información_Conexión")
  Dialogo.Minimize
end

'************************************************************
'***                METODO CAMBIAR_BD                **
'************************************************************
'** CAMBIA LA BASE DE DATOS A LA CUAL STAMOS CONECTADOS
if (Metodo = "btn_ConectarBD") then
  Dialogo = av.GetProject.FindDialog("IS_BasesdeDatosDisponibles")

  '**CAPTURA LA BASE DE DATOS.
  BD_Disponibles= Dialogo.FindByName("lbx_Bases_de_Datos_Disponibles")
  Base_Datos = BD_Disponibles.GetCurrentValue

 '**EJECUTA EL CAMBIO DE BASE DE DATOS
  Cambiar_BD                                                                    =
DLLProc.Make(_PasarelaDLL,"CAMBIAR_BD",#DLLPROC_TYPE_STR,{#DLLPROC_TYPE_STR})
  Resultado = Cambiar_BD.Call({Base_Datos})

  '**Si la consulta no es exitosa se sale

  '**MENSAJE DE CONEXION EXITOSA
  msgBox.Info(Resultado.AsString,"Mensaje de Conexion")

 '**CONSULTA EL TIPO DE USUARIO
  GET_TIPO_USUARIO=
DLLProc.Make(_PasarelaDLL,"GET_TIPO_USUARIO",#DLLPROC_TYPE_STR,{})
  Resultado = GET_TIPO_USUARIO.Call({})
  Buffer_Datos = Filename.Make(_Ruta)
```

```
  theLineFile = LineFile.Make(Buffer_Datos, #FILE_PERM_READ)
  _Tipo_Usuario = theLineFile.ReadELT

  '**HABILITA EL BOTON DE ACTUALIZACION SI EL USUARIO ES ADMINISTRADOR
  IUG         = av.FindGUI("Project")
  theBar      = IUG.GetButtonBar
  theControls = theBar.GetControls
  contador    = 0
  for each theControl in theControls
      contador =  contador+1
     '**Obviamos los espacios
      if (contador>4) then
        Click_Script   = theControl.GetClick

        if (Click_Script = "GestiónModificaciónInformación.Inicializar_IS_Modificar_Información")then
        theControl.SetEnabled(False)
        end
        if   ((Click_Script   =   "GestiónModificaciónInformación.Inicializar_IS_Modificar_Información")   and
(_Tipo_Usuario="Administrador"))then
        theControl.SetEnabled(not(theControl.IsEnabled))
        end
      end
  end

  '**CONSULTA LAS TABLAS DISPONIBLES EN LA BASE DE DATOS
  av.Run("GestionConsultas.Consulta_tablas",nil)

  '**ESTABLECEMOS LA NUEVA BASE DE DATOS CONECTADA
  _BaseDatosActual= Base_Datos
   DataBase = Dialogo.FindByName("txl_Base_de_Datos_Actual")
   DataBase.SetText(_BaseDatosActual)
   Dialogo.close
  Exit
end

'*************************************************************************************
'***   METODO CERRAR DIALOGO IS_BASESDEDATOSDISPONIBLES    **
'*************************************************************************************
if (Metodo="btn_CerrarCBD") then
  Dialogo_Conexion = av.GetProject.FindDialog("IS_BasesdeDatosDisponibles")
  Dialogo_Conexion.Close
end

'*************************************************************************************
'***   METODO MINIMIZAR DIALOGO IS_BASESDEDATOSDISPONIBLES    ***
'*************************************************************************************
if (Metodo="btn_MinimizarCBD") then
  Dialogo_Conexion = av.GetProject.FindDialog("IS_BasesdeDatosDisponibles")
  Dialogo_Conexion.Minimize
end
```

2.1.2 Script: Gestion_Conexion.Habilitar_Botones.ave

```
'*************************************************************************************************
'**METODO PUBLICO PERTENECIENTE A LA CLASE GESTION_CONEXION          **
'**SE ENCARGA DE HABILITAR O DESABILITAR LAS OPCIONES DEL MENU       **
*************************************************************************************************
IUG        = av.FindGUI("Project")
 theBar     = IUG.GetButtonBar
 theControls = theBar.GetControls
 contador    = 0

 for each theControl in theControls
    contador =  contador+1
   '**Obviamos los espacios
    if (contador>4) then
      Click_Script   = theControl.GetClick
      if ((Click_Script = "GestionConsultas.Inicializar_IS_SelecciónDatos") or (Click_Script =
"Gestion_Conexion.Desconectar")) then
      theControl.SetEnabled(not(theControl.IsEnabled))
      end
      if ((Click_Script = "GestiónTransaccionSQL.Inicializar_IS_TransacciónSQL")) then
      theControl.SetEnabled(not(theControl.IsEnabled))
      end
      if ((Click_Script = "GestiónModificaciónInformación.Inicializar_IS_Modificar_Información") and
(_Tipo_Usuario="Administrador") )then
      theControl.SetEnabled(not(theControl.IsEnabled))
      end
      if ((Click_Script = "Gestion_Conexion.Inicializar_ISBasesdeDatosDisponibles")) then
      theControl.SetEnabled(not(theControl.IsEnabled))
      end
      if ((Click_Script = "Gestion_Conexion.Inicializar_ISConexion")) then
      theControl.SetEnabled(not(theControl.IsEnabled))
      end
      if ((Click_Script = "GestionConsultas.Refrescar_Tablas")) then
      theControl.SetEnabled(not(theControl.IsEnabled))
      end
      if ((Click_Script = "Gestion_Conexion.Inicializar_ISInformación_Conexión")) then
      theControl.SetEnabled(not(theControl.IsEnabled))
      end
    end
 end
```

2.1.3 Script: Gestion_Conexion.Habilitar_Botones.ave

```
'*************************************************************************************************
'**METODO PUBLICO PERTENECIENTE A LA CLASE GESTION_CONEXION          **
'**SE ENCARGA DE INICIALIZAR LA INTERFAZ BASESDEDATOSDIPONIBLES      **
'*************************************************************************************************

'**LLAMA LA INTERFAZ ISBasesdeDatosDisponibles
ISBasesdeDatosDisponibles = av.GetProject.FindDialog("IS_BasesdeDatosDisponibles")


'**CENTRA LA INTERFAZ ISBasesdeDatosDisponibles
ptCenter = (av.ReturnOrigin + (av.ReturnExtent * (2@2))) -
(ISBasesdeDatosDisponibles.ReturnExtent.ReturnSize * (2@2))
ISBasesdeDatosDisponibles.MoveTo(ptCenter.GetX, ptCenter.GetY)
```

```
'**DESPLIEGA LAS BASES DE DATOS A LAS CUALES SE TIENE ACCESO
'**EJECUTA LA CONSULTA
ACCESO_BD = DLLProc.Make(_PasarelaDLL,"ACCESO_BD",#DLLPROC_TYPE_STR,{})
Resultado = ACCESO_BD.Call({})

'**MsgBox.info (Resultado.asString,"")
ISBasesdeDatosDisponibles.Open

'**LIMPIA LAS CAJAS DE TEXTO
BD_Disponibles= ISBasesdeDatosDisponibles.FindByName("lbx_Bases_de_Datos_Disponibles")
BD_Disponibles.Empty
BD_Disponibles.GoColumn(0)
BD_Disponibles.SetColumnWidth(3)

'**LA BASE DE DATOS ACTUAL
DataBase = ISBasesdeDatosDisponibles.FindByName("txl_Base_de_Datos_Actual")
DataBase.SetText(_BaseDatosActual)

'**DESPLIEGA LOS DATOS EN EL CAMPO PERTINENTE
Buffer_Datos = Filename.Make(_Ruta)
theLineFile = LineFile.Make(Buffer_Datos, #FILE_PERM_READ)
theLineFileSize = theLineFile.GetSize
for each rec in 1..theLineFileSize
  linea = theLineFile.readELT
  BD_Disponibles.GoRow(BD_Disponibles.GetRowCount)
  BD_Disponibles.InsertRows(1)
  BD_Disponibles.SetCurrentValue (linea)
end
```

2.1.4 Script: Gestion_Conexion.Inicializar_ISConexion.ave

```
'*****************************************************************************************************
'**METODO PUBLICO PERTENECIENTE A LA CLASE GESTION_CONEXION          **
'**SE ENCARGA DE INICIALIZAR LA INTERFAZ IS_CONEXION                 **
'*****************************************************************************************************

'**LLAMA AL DIALOGO CONEXION
Dialogo_Conexion = av.GetProject.FindDialog("IS_Conexion")

'**CENTRA EL DIALOGO
ptCenter = (av.ReturnOrigin + (av.ReturnExtent * (2@2))) - (Dialogo_Conexion.ReturnExtent.ReturnSize
* (2@2))
Dialogo_Conexion.MoveTo(ptCenter.GetX, ptCenter.GetY)

'**INICIALIZA LAS VARIABLES
Login = Dialogo_Conexion.FindByName("txl_Login")
Password = Dialogo_Conexion.FindByName("txl_Password")
Servidor = Dialogo_Conexion.FindByName("txl_Servidor")
Login.SetText("")
Password.SetText("")
Servidor.SetText("")
_Intentos=0 '**De conexion

'**ABRE LA INTERFAZ
Dialogo_Conexion.Open
```

2.1.5 Script:Gestion_Conexion.Inicializar_ISInformación_Conexión.ave

```
'*****************************************************************************************************
'**METODO PUBLICO PERTENECIENTE A LA CLASE GESTION_CONEXION          **
'**SE ENCARGA DE INICIALIZAR LA INTERFAZ IS_INFORMACION_CONEXION     **
'*****************************************************************************************************

'**LLAMA AL DIALOGO DE INFORMACION DE CONEXION
ISInformacion_Conexion = av.GetProject.FindDialog("IS_Información_Conexión")

'**CENTRA EL DIALOGO
ptCenter = (av.ReturnOrigin + (av.ReturnExtent * (2@2))) -
(ISInformacion_Conexion.ReturnExtent.ReturnSize * (2@2))
ISInformacion_Conexion.MoveTo(ptCenter.GetX, ptCenter.GetY)

'**CARGA LAS VARIABLES
Usuario = ISInformacion_Conexion.FindByName("txl_Usuario")
Servidor = ISInformacion_Conexion.FindByName("txl_Servidor")
Base_Datos = ISInformacion_Conexion.FindByName("txl_Base_Datos")
Tipo_Usuario = ISInformacion_Conexion.FindByName("txl_Tipo_Usuario")
Tiempo_Conexion = ISInformacion_Conexion.FindByName("txl_Tiempo_Conexion")

'**CALCULA EL TIEMPO TRANSCURRIDO DE CONEXION
Duracion = (Date.Now - _Hora_InicioConexion).AsSeconds
if (Duracion >= 86400) then
  Dias = (Duracion * 86400).Truncate
  Horas = ((Duracion  - (Dias*86400)) * 3600) .Truncate
  Minutos = ( (Duracion -  ((Dias*86400)+(Horas*3600) ) ) * 60).Truncate
  Segundos = Duracion - ((Dias*86400)+(Horas*3600)+(Minutos*60))
  Duracion_Total =
Dias.AsString++"Day(s)"++Horas.AsString++"Hrs"++Minutos.AsString++"Mins"+NL+Segundos.AsString
++"Segs"
elseif (Duracion >= 3600) then
  Horas = (Duracion * 3600).Truncate
  Minutos = ((Duracion - (Horas*3600)) * 60).Truncate
  Segundos = Duracion - ((Horas*3600)+(Minutos*60))
  Duracion_Total = Horas.AsString++"Hrs"++Minutos.AsString++"Mins"++Segundos.AsString++"Segs"
elseif ((Duracion >= 60) and (Duracion < 3600)) then
  Minutos = (Duracion * 60).Truncate
  Segundos = Duracion - (Minutos*60)
  Duracion_Total = Minutos.AsString++"Mins"++Segundos.AsString++"Segs"
else
  Segundos = Duracion
  Duracion_Total = Segundos.AsString++"Segs"
end


'**FIJA LAS VARIBALES
Usuario.SetLabel(_Login)
Servidor.SetLabel(_Servidor)
Base_Datos.SetLabel(_BaseDatosActual)
Tiempo_Conexion.SetLabel(Duracion_Total)
Tipo_Usuario.SetLabel(_Tipo_Usuario)

'**ABRE EL DIALOGO
ISInformacion_Conexion.Open
```

2.1.6 Script: Gestion_Consultas.ave

```
'*********************************************************************************************
'*** PROPOSITO: DECLARACIÓN DE LA CLASE GESTION_CONSULTAS     **
'*********************************************************************************************

'**TOMA EL METODO QUE SE SOLICITA
Metodo = self.getName

'**INICIALIZA EL DIALOGO
Dialogo = av.GetProject.FindDialog("IS_SelecciónDatos")


'*****************************************************************************
'***       METODO CONSULTAR CAMPOS SELECCIONADOS          **
'*****************************************************************************
'** CONSULTA LOS CAMPOS SELECCIONADOS

if (Metodo = "btn_Consulta") then

'**ENLAZA LA FUNCION CONSULTA DE PASARELA.DLL
CONSULTA =
DLLProc.Make(_PasarelaDLL,"CONSULTA",#DLLPROC_TYPE_STR,{#DLLPROC_TYPE_STR,#DLLP
ROC_TYPE_INT32,#DLLPROC_TYPE_STR})

'**CHEQUEA SI EXISTEN CAMPOS SELECCIONADOS
Campos_Seleccionados = self.GetDialog.FindByName("lbx_Campos_Seleccionados")
RowCount = Campos_Seleccionados.GetRowCount -2
if (RowCount < -1) then
  MsgBox.Error("No existen campos seleccionados.","Error")
  Exit
end

'**GENERA UNA CADENA DE CAMPOS SELECCIONADOS PARA LA CONSULTA
_Campos=List.Make
Cadena_Campos=""
if (RowCount > -1) then
 for each RowNumber in 0..RowCount
   Campos_Seleccionados.GoRow(RowNumber)
   CurrentRow = Campos_Seleccionados.GetCurrentValue
   Cadena_Campos=Cadena_Campos+CurrentRow+","
 end
end
RowNumber=RowCount+1 'Ultima variable
Campos_Seleccionados.GoRow(RowNumber)
CurrentRow = Campos_Seleccionados.GetCurrentValue
Cadena_Campos=Cadena_Campos+CurrentRow

'**TOMAMOS EL NUMERO DE CAMPOS A CONSULTAR
Numero_campos=Campos_Seleccionados.GetRowCount

'**TOMAMOS LA TABLA A CONSULTAR
Dialogo = av.GetProject.FindDialog("IS_SelecciónDatos")
Etiqueta_Tabla= Dialogo.FindByName("txl_Tabla")
Tabla = Etiqueta_Tabla.GetLabel

'**Realizamos la consulta
Resultado = CONSULTA.Call({Tabla,Numero_campos,Cadena_Campos})

'**SI LA CONSULTA NO ES EXITOSA TERMINA LA FUNCION
```

```
if(not(Resultado = "Consulta exitosa")) then
  '**Habilitamos los Metodoes
  'av.Run("Habilitar_Metodoes",nil)
  MsgBox.info (Resultado.asString,"")
  'Dialogo.close
  Exit
end

'**HACEMOS EL TRATAMIENTO DE LOS DATOS
av.Run("GestionConsultas.Tratamiento_Datos",nil)

end '** FINAL DEL METODO

'*************************************************************************
'***              METODO CONSULTAR CAMPOS                **
'*************************************************************************
'** CONSULTA LOS CAMPOS DISPONIBLES EN LA TABLA SELECCIONADA
if (Metodo = "btn_Consultar_Campos") then

'**INICIALIZA EL DIALOGO
Dialogo = av.GetProject.FindDialog("IS_SelecciónDatos")

'**TOMA LA TABLA QUE SE DESEA CONSULTAR
Etiqueta_Tabla = Dialogo.FindByName("txl_Tabla")
Tablas_Disponibles = Dialogo.FindByName("lbx_Tablas_Disponibles")
Tabla = Tablas_Disponibles.GetCurrentValue

'**EJECUTA LA CONSULTA
Consulta_Campos =
DLLProc.Make(_PasarelaDLL,"CONSULTA_CAMPOS",#DLLPROC_TYPE_STR,{#DLLPROC_TYPE_S
TR})
Resultado = Consulta_Campos.Call({Tabla})

'**SI LA CONSULTA NO ES EXITOSA SE SALE
if(not(Resultado = "Consulta exitosa")) then
  '**Habilitamos los Metodoes
  av.Run("Habilitar_Metodoes",nil)
  MsgBox.info (Resultado.asString,"")
  Dialogo.close
  Exit
end

'**DESPLIEGA EL NOMBRE DE LA TABLA CONSULTADA
Etiqueta_Tabla.SetLabel(Tabla)

'**ACTUALIZA LOS CAMPOS DESPLEGADOS EN LA INTERFAZ
Dialogo.Close
av.Run("GestionConsultas.Inicializar_IS_SelecciónDatos",nil)

'**DESPLIEGA LOS RESULTADOS EN EL CAMPO PERTINENTE
Buffer_Datos = Filename.Make(_Ruta)
theLineFile = LineFile.Make(Buffer_Datos, #FILE_PERM_READ)
theLineFileSize = theLineFile.GetSize
Lista_Campos= Dialogo.FindByName("lbx_Lista_Campos")
for each rec in 1..theLineFileSize
  Datos = theLineFile.readELT
  Lista_Campos.GoRow(Lista_Campos.GetRowCount)
  Lista_Campos.InsertRows(1)
  Lista_Campos.SetCurrentValue (Datos)
end
```

```
'**HABILITA EL Metodo DE SELECCION DE CAMPOS
self.GetDialog.FindByName("btn_Seleccionar_Todo").SetEnabled(True)

end '** FINAL DEL METODO


'*****************************************************************
'***              METODO CERRAR DIALOGO                **
'*****************************************************************
'** CIERRA EL DIALOGO
if (Metodo = "btn_Cancel") then
  Dialogo = av.GetProject.FindDialog("IS_SelecciónDatos")
  Dialogo.close
  Exit
end

'*********************************************************************
'***              METODO MINIMIZAR DIALOGO             **
'*********************************************************************
'** MINIMIZA EL DIALOGO
if (Metodo = "btn_Minimizar") then
  Dialogo = av.GetProject.FindDialog("IS_SelecciónDatos")
  Dialogo.Minimize
  Exit
end

'*********************************************************************
'***              METODO SELECCIONAR TODO              **
'*********************************************************************
'** SELECCIONA UNA SERIE DE CAMPOS PARA CONSULTA
if (Metodo = "btn_Des_Seleccionar_Todo") then
  '**Realiza la des_seleccion de todos los campos
  Campos_Seleccionados = Dialogo.FindByName("lbx_Campos_Seleccionados")
   Lista_Campos = Dialogo.FindByName("lbx_Lista_Campos")

   RowCount = Campos_Seleccionados.GetRowCount -1

  '**Pasamos todos los campos de la tabla para consulta
  for each RowNumber in 0..RowCount
  Campos_Seleccionados.GoRow(0)
  CurrentRow = Campos_Seleccionados.GetCurrentValue
  Campos_Seleccionados.DeleteRows(1)
  Lista_Campos.GoRow(Lista_Campos.GetRowCount)
  Lista_Campos.InsertRows(1)
  Lista_Campos.SetCurrentValue (CurrentRow)
  end

  '**Habilita los Metodoes pertinentes
  Dialogo.FindByName("btn_Des_Seleccionar").SetEnabled(false)
  Dialogo.FindByName("btn_Des_Seleccionar_Todo").SetEnabled(False)
  Dialogo.FindByName("btn_Seleccionar_Todo").SetEnabled(True)

  Exit
end
```

```
'*************************************************************
'***              METODO SELECCIONAR                  **
'*************************************************************
'** SELECCIONA UN CAMPO PARA CONSULTA
if (Metodo = "btn_Seleccionar") then
  '**Trasnfiere los campos para la consulta
  Lista_Campos = Dialogo.FindByName("lbx_Lista_Campos")

  '**Toma el campo actual y lo trasnfiere
  Lista_Campos.SelectCurrent(true)
  valor = Lista_Campos.GetCurrentValue
  Lista_Campos.DeleteRows(1)
  Campos_Seleccionados = Dialogo.FindByName("lbx_Campos_Seleccionados")
  Campos_Seleccionados.GoRow(Campos_Seleccionados.GetRowCount)
  Campos_Seleccionados.InsertRows(1)
  Campos_Seleccionados.SetCurrentValue (valor)
  self.GetDialog.FindByName("btn_Seleccionar").SetEnabled(false)
  Exit

end


'*************************************************************
'***             METODO DES_SELECCIONAR          **
'*************************************************************
'** DESSELECCIONA UN CAMPO EN LA CONSULTA

if (Metodo = "btn_Des_Seleccionar") then
  '** Realiza la des_seleccion de los campos

  Campos_Seleccionados = Dialogo.FindByName("lbx_Campos_Seleccionados")
  Campos_Seleccionados.SelectCurrent(true)
  valor = Campos_Seleccionados.GetCurrentValue
  Campos_Seleccionados.DeleteRows(1)
  Lista_Campos = Dialogo.FindByName("lbx_Lista_Campos")
  Lista_Campos.GoRow(Lista_Campos.GetRowCount)
  Lista_Campos.InsertRows(1)
  Lista_Campos.SetCurrentValue (valor)
  self.GetDialog.FindByName("btn_Des_Seleccionar").SetEnabled(false)
  Exit

end


'*************************************************************************
'***             METODO DES_SELECCIONAR TODO           **
'*************************************************************************
'** DESSELECCIONA LOS CAMPOS EN LA CONSULTA
if (Metodo = "btn_Seleccionar_Todo") then
  '**Pasamos todos los campos de la tabla para consulta

  Campos_Seleccionados = Dialogo.FindByName("lbx_Lista_Campos")
  Lista_Campos = Dialogo.FindByName("lbx_Campos_Seleccionados")

  RowCount = Campos_Seleccionados.GetRowCount -1

  '**Pasamos todos los campos de la tabla para consulta
  for each RowNumber in 0..RowCount
    Campos_Seleccionados.GoRow(0)
    CurrentRow = Campos_Seleccionados.GetCurrentValue
    Campos_Seleccionados.DeleteRows(1)
```

```
    Lista_Campos.GoRow(Lista_Campos.GetRowCount)
    Lista_Campos.InsertRows(1)
    Lista_Campos.SetCurrentValue (CurrentRow)
  end

  '**Habilita los Metodoes pertinentes
  self.GetDialog.FindByName("btn_Seleccionar").SetEnabled(false)
  self.GetDialog.FindByName("btn_Seleccionar_Todo").SetEnabled(False)
  self.GetDialog.FindByName("btn_Des_Seleccionar_Todo").SetEnabled(True)

  '**Se sale del bucle
  Exit
end
```

2.1.7 Script: GestionConsultas.Consulta_tablas.ave

```
'*************************************************************************************************
'**METODO PUBLICO PERTENECIENTE A LA CLASE GESTION_CONSULTAS        **
'**SE ENCARGA DE CONSULTAR LAS TABLAS DISPONIBLES EN LA B.D.        **
'*************************************************************************************************
'**ENCARGADO DE CONSULTAR LAS TABAS DISPONIBLES EN LA BASE DE DATOS
'**ENLAZADA

'**ENLAZA EL DIALOGO
Dialogo = av.GetProject.FindDialog("IS_SelecciónDatos")

'**EJECUTA LA CONSULTA DE LAS TABLAS
Consutla_Tablas = DLLProc.Make(_PasarelaDLL,"CONSULTA_TABLAS",#DLLPROC_TYPE_STR,{})
Resultado = Consutla_Tablas.Call({})

'**SI LA CONSULTA ES EXITOSA...
if (Resultado = "Consulta exitosa") then '** Consulta exitosa
  Archivo_Origen = Filename.Make(_Ruta)
  theLineFile_Origen = LineFile.Make(Archivo_Origen, #FILE_PERM_READ)
  theLineFileSize = theLineFile_Origen.GetSize

  '**GUARDA LAS TABLAS DISPONIBLES EN UNA VARIABLE GLOBAL
  '**PARA SU POTSERIOR USO
  _Tablas = List.Make
  for each rec in 1..theLineFileSize
    linea = theLineFile_Origen.readELT
    _Tablas.add(linea)
    'theLineFile_Destino.WriteELT(linea)
  end


else '**HAY PROBLEMAS EN LA CONEXION
  '**Habilitamos los botones
  av.Run("Gestion_Conexion.Habilitar_Botones",nil)
  MsgBox.info (Resultado.asString,"")
  Dialogo.close
End
```

2.1.8 Script: GestionConsultas.Guardar/Eliminar_Consulta.ave

```
'*************************************************************************************************
'**METODO PUBLICO PERTENECIENTE A LA CLASE GESTION_CONSULTAS        **
'**SE ENCARGA DE GUARDAR LA INFORMACION CONSULTADA                  **
'*************************************************************************************************

'** INICIALIZA EL NOMBRE POR DEFECTO DE LA CONSULTA
Tabla_Defecto = System.GetEnvVar("AVHOME")+"\bin32\consulta.dbf"
Nombre_Defecto = "Consulta.dbf".AsFileName

'** PRESENTA EL DIALOGO DONDE SE ELIGE LA RUTA Y EL NOMBRE DEL ARCHIVO.
Tabla_Destino = FileDialog.Put(Nombre_Defecto, "*.dbf", "Salvar tabla como:")

'**OBTIENE EL NOMBRE
Lista = Tabla_Destino.AsString.AsTokens("\")
NombreTabla = Lista.Get(Lista.Count-1)

'** OBIENTE LA TABLA FUENTE
Documento = av.GetActiveDoc
Tabla_Fuente =Documento.AsString.AsFileName

'** CRREA LA NUEVA TABLA Y ELIMINA LA ANTERIOR
if (nil <> Tabla_Destino) then

  '**SI DESEA SOBREESCRIBIR
  '**ACTUALIZAMOS LA TABLA
  if (Tabla_Fuente=Tabla_Destino) then
    Exit
  end


  '**SI LA TABLA EXISTE Y ESTA ABIERTA
  if (av.FindDoc(NombreTabla).AsString = NombreTabla) then
  av.GetProject.RemoveDoc(av.FindDoc(NombreTabla))
  end

  '**ELIMINA LA ANTERIOR SI LA TABLA ESTA ABIERTA
  if (av.FindDoc(Documento.AsString).AsString = Documento.AsString) then
    Documento.GetWin.Close
    av.GetProject.RemoveDoc(Documento)
    'File.Delete(Tabla_Fuente)
  end

  '**CREA LA TABLA
  File.Copy(Tabla_Fuente,Tabla_Destino)
  TheVtab= Vtab.Make(Tabla_Destino,false,false)
  Tabla= Table.Make(TheVtab)
  Tabla.SetName(NombreTabla)
  Tabla.GetWin.Open

end
```

2.1.9 Script: GestionConsultas.Inicializar_IS_SelecciónDatos.ave

```
'*****************************************************************************************************
'**METODO PUBLICO PERTENECIENTE A LA CLASE GESTION_CONSULTAS        **
'**SE ENCARGA DE INICIALIZAR LA INTERFAZ IS_SELECCIONDATOS          **
'*****************************************************************************************************

'**IDENTIFICA EL Dialogo_SeleccionDatos
Dialogo_SeleccionDatos = av.GetProject.FindDialog("IS_SelecciónDatos")

'**CENTRA EL DIALOGO
ptCenter = (av.ReturnOrigin + (av.ReturnExtent * (2@2))) -
(Dialogo_SeleccionDatos.ReturnExtent.ReturnSize * (2@2))
Dialogo_SeleccionDatos.MoveTo(ptCenter.GetX, ptCenter.GetY)

'LIMPIA LAS CAJAS DE TEXTO
Tablas_Disponibles = Dialogo_SeleccionDatos.FindByName("lbx_Tablas_Disponibles")
Tablas_Disponibles.Empty
Tablas_Disponibles.GoColumn(0)
Tablas_Disponibles.SetColumnWidth(3)
Dialogo_SeleccionDatos.FindByName("btn_Seleccionar").SetEnabled(false)
Dialogo_SeleccionDatos.FindByName("btn_Des_Seleccionar_Todo").SetEnabled(False)
Dialogo_SeleccionDatos.FindByName("btn_Seleccionar_Todo").SetEnabled(False)
Dialogo_SeleccionDatos.FindByName("btn_Consultar_Campos").SetEnabled(False)
Dialogo_SeleccionDatos.SetTitle("Base de datos:"++_BaseDatosActual)

Lista_Campos = Dialogo_SeleccionDatos.FindByName("lbx_Lista_Campos")
Lista_Campos.Empty
Lista_Campos.GoColumn(0)
Lista_Campos.SetColumnWidth(3)
Dialogo_SeleccionDatos.FindByName("btn_Des_Seleccionar").SetEnabled(false)

Campos_Seleccionados = Dialogo_SeleccionDatos.FindByName("lbx_Campos_Seleccionados")
Campos_Seleccionados.Empty
Campos_Seleccionados.GoColumn(0)
Campos_Seleccionados.SetColumnWidth(3)
Dialogo_SeleccionDatos.FindByName("btn_Des_Seleccionar").SetEnabled(false)

'**DESPLIEGA LAS TABLAS DISPONIBLES
theLineFileSize = _Tablas.Count-1
for each rec in 0..theLineFileSize
  Tablas_Disponibles.GoRow(Tablas_Disponibles.GetRowCount)
  Tablas_Disponibles.InsertRows(1)
  Tablas_Disponibles.SetCurrentValue(_Tablas.Get(rec))
end

'**ABRE EL DIALOGO
Dialogo_SeleccionDatos.Open
```

2.1.10 Script: GestionConsultas.Refrescar_Tablas.ave

```
'**********************************************************************
'**METODO PUBLICO PERTENECIENTE A LA CLASE GESTION_CONSULTAS         **
'**SE ENCARGA DE ACTUALIZAR LAS TABLAS RELACIONADAS CON EL PROYCTO   **
'**DISPONIBLES EN LA BASE DE DATOS                                   **
'**********************************************************************

'SE ESCOJEN LAS TABLAS DISPONIBLES EN EL PROYECTO QUE SE DESEAN ACTUALIZAR
Tablas = av.GetProject.GetDocswithGui(av.FindGui("Table"))

if (Tablas=nil)  then
  MsgBox.Info("No hay tablas para seleccionadas","")
  Exit
end

NoTablas = Tablas.Count
NTablas=_Tablas.Count

if (NoTablas= 0)  then
  MsgBox.Info("No hay tablas seleccionadas.","")
  Exit
end

'**AVERIGUA EN EL SERVIDOR QUE TABLAS HAY PARA ACTUALIZAR
TablasDisponibles = List.Make
for each cont in 1..NoTablas
  for each conta in 1..NTablas
    if(_Tablas.Get(conta-1).AsString+".dbf"=Tablas.Get(cont-1).AsString)then
      LaVtab=Vtab.Make(_Tablas.Get(conta-1).AsFileName,false,false)
      TablasDisponibles.Add(Tablas.Get(cont-1))
    end
  end
end


'PREGUNTA LAS TABLAS QUE DESEA ACTUALIZAR
TablasElegidas = MsgBox.Multilist(TablasDisponibles, "Escoja las tabla que desea actualizar:",
"Actualizar Tablas")
if (TablasElegidas= nil)  then
  Exit
end

NoTablas = TablasElegidas.Count
if (NoTablas= 0)  then
  MsgBox.Info("No hay tablas seleccionadas","")
  Exit
end

'ACTUALIZA CADA TABLA SELECCIONADA
for each cont in 1..NoTablas
  TablaActual=TablasElegidas.Get(cont-1).GetVtab
  Tabla_Destino=TablasElegidas.Get(cont-1).GetVtab

  Lista=TablasElegidas.Get(cont-1).AsString.AsTokens(".")
  NombreTabla=Lista.Get(0).AsString
  RutaTabla=TablaActual.GetBaseTableFileName.AsString

  Campos = TablaActual.GetFields
  NoCampos = Campos.Count
```

```
  NoCampos = NoCampos-1

  if (NoCampos= 0)  then
    Exit
  end

  CamposSeleccionados =""
  for each cont in 1..NoCampos
    CamposSeleccionados =CamposSeleccionados+Campos.Get(cont-1).AsString+","
  end
  CamposSeleccionados =CamposSeleccionados+Campos.Get(NoCampos).AsString

  '**CONSULTA LA TABLA
  '**ENLAZA LA FUNCION CONSULTA DE PASARELA.DLL
  CONSULTA =
DLLProc.Make(_PasarelaDLL,"CONSULTA",#DLLPROC_TYPE_STR,{#DLLPROC_TYPE_STR,#DLLP
ROC_TYPE_INT32,#DLLPROC_TYPE_STR})

  '**REALIZA LA CONSULTA
  NoCampos = NoCampos+1
  Resultado = CONSULTA.Call({NombreTabla,NoCampos,CamposSeleccionados})

  '**SI LA CONSULTA NO ES EXITOSA TERMINA LA FUNCION
  if(not(Resultado = "Consulta exitosa")) then
    '**Habilitamos los Botones
    'av.Run("Habilitar_Botones",nil)
    MsgBox.info (Resultado.asString,"")
    'Dialogo.close
    Exit
  end

  '**HACEMOS EL TRATAMIENTO DE LOS DATOS
  av.Run("GestionConsultas.Tratamiento_Datos",nil)


  '**CREA LA TABLA FUENTE
  fileString = System.GetEnvVar("AVHOME")+"\bin32\Consulta"+".dbf"
  theDBFName = fileString.AsFileName
  Tabla_Fuente = VTab.Make(theDBFName, false,false)

  "**EDITA LAS TABLAS
  numRecs = Tabla_Destino.GetNumRecords
  Tabla_Destino.StartEditingWithRecovery
  Tabla_Destino.BeginTransaction
  numRecsTF = Tabla_Fuente.GetNumRecords
  Tabla_Fuente.StartEditingWithRecovery
  Tabla_Fuente.BeginTransaction

  '**LIMPIA LA TABLA DE DESTINO
  for each rec in 1..numRecs
    Tabla_Destino.RemoveRecord(rec-1)
  end

  '**LISTA DE CAMPOS
  lstFldPairs = {}
  for each f in Tabla_Destino.GetFields
    g = Tabla_Fuente.FindField(f.GetAlias)
    if (g <> NIL) then
      lstFldPairs.Add({f,g})
    end
  end
```

```
  '**COPIA LA TABLA
  for each rec in 1..numRecsTF
    newRec = Tabla_Destino.AddRecord
    for each fg in lstFldPairs
      objValue = Tabla_Fuente.ReturnValue( fg.Get(1), rec-1 )
      Tabla_Destino.SetValue( fg.Get(0), newRec, objValue )
    end
  end

  '**DESEDITA LAS TABLAS
  Tabla_Fuente.EndTransaction
  Tabla_Fuente.StopEditingWithRecovery(true)
  Tabla_Destino.EndTransaction
  Tabla_Destino.StopEditingWithRecovery(true)

  '**CIERRA LA VENTANA
  Documento = av.GetActiveDoc
  Documento.GetWin.Close

end


'**INFORMA EL FIN DE LA ACTULIZACION
MsgBox.Info("Las tablas fueron actualizadas exitosamente","Información")
```

2.1.11 Script: GestionConsultas.Tratamiento_Datos.ave

```
'*************************************************************************************************************
'**METODO PUBLICO PERTENECIENTE A LA CLASE GESTION_CONSULTAS                **
'**SE ENCARGA DE LEER EL BUFFER DE DATOS Y CONVERTIR LA                     **
'**INFORMACION EN UNA BASE DE DATOS ASOCIADA AL PROYECTO EN CURSO          **
'*************************************************************************************************************

'**DETERMINA EL BUFFER_DATOS A CONVERTIR Y CREA UNA BASE DE DATOS VACIA.

Buffer_Datos = Filename.Make(_Ruta)

if (Not(File.Exists(Buffer_Datos))) then
 MsgBox.Info("El archivo de texto no existe","Error")
 Exit
end

if (File.Exists(Buffer_Datos)) then
 '**REVISA LAS CONSULTAS HECHAS
    fileString = System.GetEnvVar("AVHOME")+"\bin32\Consulta"+".dbf"


 if (av.FindDoc(fileString).AsString = fileString.AsString) then
 Documento = av.FindDoc(fileString)
 av.GetProject.RemoveDoc(Documento)
 end

 theDBFName = fileString.AsFileName
 theVTab = VTab.MakeNew(theDBFName, dbase)
 theTable = Table.Make(theVTab)
 theTable.SetName(theDBFName.AsString)
end
```

```
'**LEE LA PRIMERA LINEA PARA DETEMINAR LOS NOMBRES DE LOS CAMPOS.

av.ShowMsg("Reading Source File....")
theLineFile = LineFile.Make(Buffer_Datos, #FILE_PERM_READ)
fieldNmString = theLineFile.ReadELT
fieldNmList = fieldNmString.AsTokens("®".quote)

'**DETERMINA COMO ESTA DELIMITADO EL BUFFER_DATOS.

theComma = "*®*".AsPattern
'theSpace = "* *".AsPattern
delimit = "Comma"

'**DETERMINA EL ANCHO DE LOS CAMPOS
fldCount = fieldNmList.Count
theWidthList = List.Make

'**INICIALIZA LA LISTA
for each fld in 1..fldCount
 theWidthList.Add(1)
end

'**LEE LAS LINEAS DE TEXTO Y OBTIENE EL MAXIMO ANCHO PARA CADA CAMPO
theLineFileSize = theLineFile.GetSize

for each rec in 1..theLineFileSize
 theLine = theLineFile.readELT

 if (theLine = Nil) then
  continue
 end

 '**CHEQUEA POR ANCHO Y LOS CAMBIA SI ES NECESARIO
 if (theLine.Contains("""")) then
  theLine = theLine.Substitute("""", "")
 end

 thePosition = theLineFile.GetPos
 theLineLength = theLine.Count

 if (thePosition > 0) then

   for each fld in 1..fldCount

   if (delimit = "Comma") then
    theOffset = theLine.IndexOf("®")
   elseif (delimit = "Space") then
    theOffset = theLine.IndexOf(" ")
   end

   if (theOffset = -1) then
    theOffset = theLine.Count
   end

   theValue = theLine.Left(theOffset)
   theValueWidth = theValue.Count+2
   theExistingWidth = theWidthList.Get(fld-1)

   if (theValueWidth > theExistingWidth) then
    theWidthList.Set((fld-1), theValueWidth)
```

```
    theExistingWidth = theValueWidth
   end

   theLine = theLine.Right(theLineLength-(theOffset+1))
   theLineLength = theLine.Count
  end
 end
end

'**CREA LOS CAMPOS
for each fld in 1..fldCount
 theField = Field.Make(fieldNmList.Get(fld-1).AsString,#FIELD_CHAR, theWidthList.Get(fld-
1).AsString.AsNumber, 0)
 theVTab.AddFields({theField})
end

theVtab.SetEditable(True)
theFieldList = theVTab.GetFields
newLineFile = LineFile.Make(Buffer_Datos, #FILE_PERM_READ)

'**BUCLE A TRAVES DEL ARCHIVO DE TEXTO
for each line in 1..theLineFileSize
 thePosition = newLineFile.GetPos
 theLine = newLineFile.ReadELT

 if (theLine.Contains("""")) then
   theLine = theLine.Substitute("""", "")
 end

 theLineLength = theLine.Count
 if (thePosition > 0) then
  newRec = theVtab.AddRecord

  '**BUCLE A TRAVES DEL CUAL LOS CAMPOS SON ADERIDOS A LA BD
  for each fld in theFieldList

   if (delimit = "Comma") then
    theOffset = theLine.IndexOf("®")
   elseif (delimit = "Space") then
    theOffset = theLine.IndexOf(" ")
   end

   if (theOffset = -1) then
    theOffset = theLine.Count
   end

   theValue = theLine.Left(theOffset)
   theVTab.SetValue(fld,newRec,theValue)
   theLine = theLine.Right(theLineLength-(theOffset+1))
   theLineLength = theLine.Count
  end
 end
end
theVTab.SetEditable(False)

'**ABRE LA NUEVA TABLA
TheTable.GetWin.Open
```

2.1.12 Script: GestiónModificaciónInformación.ave

```
'*****************************************************************************
'***  CLASE ENCARGADA DE LA OPERACION DE ACTUALIZACION    **
'*****************************************************************************

'**TOMAMOS EL METODO QUE SE SOLICITA
Metodo = self.getName

'**INICIALIZA LAS VARIABLES
Dialogo = av.GetProject.FindDialog("IS_Modificar_Información")
Metodo = self.getName
Tablas=Dialogo.FindByName("cbx_Tablas_Disponibles")
ctheListBox= Dialogo.FindByName("lbx_Lista_Campos")
Tabla_Elegida= Tablas.GetCurrentValue
Campo = ctheListBox.GetCurrentValue
Etiqueta= Dialogo.FindByName("txl_Campo_Actualizar")
Campos_Selec=  Dialogo.FindByName("lbx_Valores_Campo_Referencia")

'*****************************************************************
'***              METODO CERRAR DIALOGO              **
'*****************************************************************
'**CIERRA EL DIALOGO
if (Metodo = "btn_Cerrar") then
  Dialogo.close
  Exit
end

'*****************************************************************
'***              METODO MINIMIZAR DIALOGO           **
'*****************************************************************
'**CIERRA EL DIALOGO
if (Metodo = "btn_Minimizar") then
  Dialogo.Minimize
  Exit
end

'*****************************************************************
'***              METODO CONSULTAR TABLA             **
'*****************************************************************
'**SE ENCARGA DE RECUPERAR LOS CAMPOS DE LA TABLA SELECCIONADA.
if (Metodo = "btn_Ver_tabla") then

  '**CREA LA CONSULTA
  CbxTabla=Dialogo.FindByName("cbx_Tablas_Disponibles")
  Tabla = CbxTabla.GetCurrentValue
  consulta= "select * from "+Tabla

  '**EJECUTA LA CONSULTA
  add =
DLLProc.Make(_PasarelaDLL,"TRANSACCION_SQL",#DLLPROC_TYPE_STR,{#DLLPROC_TYPE_ST
R})
  Resultado = add.Call({consulta})

  '** CONSULTA NO EXITOSA
    if (not(Resultado = "Consulta exitosa"))  then
   '**Problemas en la conexion
   '**Habilitamos los Metodoes
   '**av.Run("GestionConsultas.GestionConsultas.Tratamiento_Datos",nil)
   MsgBox.Report(Resultado,"Error")
   '**MsgBox.info (Resultado.asString,"")
```

```
  end

  '**HACE EL TRATAMIENTO DE LOS DATOS
  av.Run("GestionConsultas.Tratamiento_Datos",nil)

    Exit
end

'*************************************************************************
'***            METODO CONSULTAR_CLAVE_PRINCIPAL         **
'*************************************************************************
'** EXTRAE LA CLAVE PRINCIPAL DE LA TABLA SELECCIONADA
if (Metodo = "btn_Traer_PK") then

 Consulta_Sql= "SELECT COLUMN_NAME from INFORMATION_SCHEMA.TABLE_CONSTRAINTS
JOIN INFORMATION_SCHEMA.KEY_COLUMN_USAGE"++
                "ON (INFORMATION_SCHEMA.TABLE_CONSTRAINTS.CONSTRAINT_NAME =
INFORMATION_SCHEMA.KEY_COLUMN_USAGE.CONSTRAINT_NAME)"++
            "WHERE
INFORMATION_SCHEMA.TABLE_CONSTRAINTS.TABLE_NAME='"+Tabla_Elegida+"' AND
CONSTRAINT_TYPE = 'PRIMARY KEY'"

'**EJECUTA LA CONSULTA
  add =
DLLProc.Make(_PasarelaDLL,"TRANSACCION_SQL",#DLLPROC_TYPE_STR,{#DLLPROC_TYPE_ST
R})
  Resultado = add.Call({ Consulta_Sql})
  '**Ejecuta la consulta

  '** CONSULTA NO EXITOSA
  if (not(Resultado = "Consulta exitosa"))  then
    '**Problemas en la conexion
    '**Habilitamos los Metodoes
    '**av.Run("GestionConsultas.GestionConsultas.Tratamiento_Datos",nil)
    MsgBox.Report(Resultado,"Error")
    '**MsgBox.info (Resultado.asString,"")
    Exit
  end

  '**DESPLIEGA LOS DATOS EN EL CAMPO PERTINENTE
  Buffer_Datos = Filename.Make(_Ruta)
  theLineFile = LineFile.Make(Buffer_Datos, #FILE_PERM_READ)
  theLineFileSize = theLineFile.GetSize

  '**DESPLEGA LA COLUMNA PRINCIPAL
  Campo_Elegido= Dialogo.FindByname("txl_Campo_de_Referencia")
  linea = theLineFile.readELT
  linea = theLineFile.readELT

    '**DETECTA SI NO EXISTE CLAVE PRINCIPAL
  if (linea.AsString="nil") then
    MsgBox.Error("Esta tabla no contiene una clave primaria. Elija otro campo.","Error")
    Exit
  end

  Campo_Elegido.SetLabel(linea)

  '**LLAMA AL SCRIPT DE CONSULTA CAMPO, PASANDOLE LOS PARAMETROS ADECUADOS.
  Campo=linea
  Metodo = "btn_Otro_Campo_Ref"
end
```

```
'*****************************************************************
'***              METODO OTRO_CAMPO_REF                **
'*****************************************************************
'** RECUPERA EL CAMPO SELECCIONADO Y LO COLOCA COMO CAMPO DE REFERENCIA
if (Metodo = "btn_Otro_Campo_Ref") then

  CONSULTA =
DLLProc.Make(_PasarelaDLL,"CONSULTA",#DLLPROC_TYPE_STR,{#DLLPROC_TYPE_STR,#DLLP
ROC_TYPE_INT32,#DLLPROC_TYPE_STR})

  '**OBTIENE EL CAMPO DE CONSULTA
  Campo_Elegido= Dialogo.FindByname("txl_Campo_de_Referencia")
  Campo_Elegido.SetLabel(Campo)

  '**REALIZA LA CONSULTA
  Resultado = CONSULTA.Call({Tabla_Elegida,1,Campo})

  '**SI LA CONSULTA NO ES EXITOSA SE SALE
  if(not(Resultado = "Consulta exitosa")) then
    '**Habilitamos los Metodoes
    'av.Run("GestionConsultas.GestionConsultas.Tratamiento_Datos",nil)
    MsgBox.info (Resultado.asString,"")
    'Dialogo.close
    'Exit
  end

  '**DESPLIEGA LOS DATOS EN EL CAMPO PERTINENTE
  Buffer_Datos = Filename.Make(_Ruta)
  theLineFile = LineFile.Make(Buffer_Datos, #FILE_PERM_READ)
  theLineFileSize = theLineFile.GetSize

  ctheListBox = Dialogo.FindByName("lbx_Valores_Campo_Referencia")

  '**LIMPIA EL CAMPO
  ctheListBox.Empty
  ctheListBox.GoColumn(0)
  ctheListBox.SetColumnWidth(3)

  '\\IGNORA LA PRIMERA FILA, QUE CORRESONDE AL NOMBRE DE LA COLUMNA
  linea = theLineFile.readELT

  '\\DESPLIEGA LOS DATOS
  for each rec in 2..theLineFileSize
    linea = theLineFile.readELT
    ctheListBox.GoRow(ctheListBox.GetRowCount)
    ctheListBox.InsertRows(1)
    ctheListBox.SetCurrentValue (linea)
  end

  Exit
end
```

```
'**********************************************************************
'***              METODO SELECCIONAR_CAMPO                **
'**********************************************************************
'** SELECCIONA EL CAMPO A ACTUALIZAR
if (Metodo = "btn_Seleccionar_Campo") then
  Etiqueta.SetLabel(Campo)
  Exit
end


'**********************************************************************
'***              METODO ACTUALIZAR_CELDA                 **
'**********************************************************************
'** INTROUCE EL NUEVO CAMPO EN LA CELDA

if (Metodo = "btn_Actualizar") then

  '**CREA LA CONSULTA DE ACTUALIZACION
  Tmp= Dialogo.FindByName("lbx_Valor_Actualizar")
  Actualizar=Tmp.GetText

  '**WHERE
  Tmp=  Dialogo.FindByName("lbx_Valores_Campo_Referencia")
  Campo_Sel = Tmp.GetCurrentValue
  Update = Etiqueta.GetLabel

  '**OBTIENE EL VALOR DEL CAMPO A ACTUALIZAR
  Tmp= Dialogo.FindByName("txl_Valor_Campo_de_Referencia")
  Campo = Tmp.GetText

  '**HAY CAMPOS SELECCIONADOS?
  if  (Campo="") then
  MsgBox.Error("Debe seleccionar un campo para actualizar","Error")
  Exit
  end
  Etiqueta= Dialogo.FindByName("txl_Campo_Actualizar")
  Campo = Etiqueta.GetLabel
  if  (Campo="") then
  MsgBox.Error("Debe seleccionar un campo para actualizar","Error")
  Exit
  end

  Etiqueta= Dialogo.FindByName("txl_Campo_de_Referencia")
  Campo = Etiqueta.GetLabel

   '**ENLAZA LA DLL
  ACTUALIZA =
DLLProc.Make(_PasarelaDLL,"ACTUALIZAR",#DLLPROC_TYPE_STR,{#DLLPROC_TYPE_STR,#DLL
PROC_TYPE_STR,#DLLPROC_TYPE_STR,#DLLPROC_TYPE_STR,#DLLPROC_TYPE_STR})
  Resultado = ACTUALIZA.Call({Tabla_Elegida,Update,Campo,Campo_Sel,Actualizar})

    '** CONSULTA NO EXITOSA
  if (not(Resultado = "Actualizacion Exitosa"))  then
    '**Problemas en la conexion
    '**Habilitamos los Metodoes
    '**av.Run("Habilitar_Metodoes",nil)
    MsgBox.Report(Resultado,"Error")
    '**MsgBox.info (Resultado.asString,"")
    Exit
  end
```

```
  '**SIGUIENTE CAMPO
  '**Aqui va el codigo para que sea automatica la actualizacion
  Exit
end

'************************************************************
'***                 METODO BORRAR                        **
'************************************************************
'** BORRAMOS EL VALOR DE ACUTALIZACION
 if (Metodo = "btn_Borrar") then
    Tmp= Dialogo.FindByName("lbx_valor_actualizar")
    Tmp.SetText("")
 end
```

2.1.13 Script:GestiónModificaciónInformación.Inicializar_IS_Modificar_Información.ave

```
'**************************************************************************
'**METODO PUBLICO PERTENECIENTE A LA CLASE GESTION_MODIFICACION_INFO **
'**SE ENCARGA DE INICIALIZAR LA INTERFAZ IS_MODIFICAR_INFORMACION    **
'**************************************************************************

'**IDENTIFICA EL DIALOGO
Dialogo = av.GetProject.FindDialog("IS_Modificar_Información")


'**CENTRA EL DIALOGO
ptCenter = (av.ReturnOrigin + (av.ReturnExtent * (2@2))) - (Dialogo.ReturnExtent.ReturnSize * (2@2))
Dialogo.MoveTo(ptCenter.GetX, ptCenter.GetY)

'**LIMPIA LAS CAJAS DE TEXTO
Lista_Campos = Dialogo.FindByName("lbx_Lista_Campos")
Lista_Campos.Empty
Lista_Campos.GoColumn(0)
Lista_Campos.SetColumnWidth(3)


'**PONE EL NOMBRE DE LA BASE DE DATOS CONECTADA
Dialogo.SetTitle("Base de datos:"++_BaseDatosActual)


'**LIMPIA LAS CAJAS DE TEXTO
Valores_Campo_Referencia = Dialogo.FindByName("lbx_Valores_Campo_Referencia")
Valores_Campo_Referencia.Empty
Valores_Campo_Referencia.GoColumn(0)
Valores_Campo_Referencia.SetColumnWidth(3)

Texto= Dialogo.FindByName("txl_Valor_Campo_de_Referencia")
Texto.SetText("")

Etiqueta= Dialogo.FindByName("txl_Campo_de_Referencia")
Etiqueta.SetLabel("")

Etiqueta= Dialogo.FindByName("txl_Campo_Actualizar")
Etiqueta.SetLabel("")

Texto= Dialogo.FindByName("lbx_Valor_Actualizar")
Texto.SetText("")

Tablas = Dialogo.FindByName("cbx_Tablas_Disponibles")
```

```
Tablas.DefineFromList(_Tablas)
Tablas.Update

Campo_Consulta = Dialogo.FindByName("lbx_Valor_Actualizar")
Campo_Consulta.SetText("")

'**ABRE EL DIALOGO
Dialogo.Open
```

2.1.14 Script: GestiónTransaccionSQL.ave

```
'*****************************************************************************************
'*** CLASE ENCARGADA DE COLABORAR EN LA CREACION DE UNA   **
'*** CONSULTA SQL Y EJECUTARLA                            **
'*****************************************************************************************

'**INICIALIZA LAS VARIABLES
Dialogo = av.GetProject.FindDialog("IS_TransacciónSQL")
Consulta_SQL = Dialogo.FindByName("txbox_Consulta_SQL")
Metodo = self.getName
Cadena=""

'*****************************************************************
'***            METODO EJECUTAR CONSULTA                 **
'*****************************************************************
'** ENCARGADO DE HACER LA CONSULTA
if (Metodo="btn_Ejecutar") then

  '**LEE LA CONSULTA
   consulta =  Consulta_SQL.getText

  '**SI NO EXISTE LA TRANSACCION
  if(consulta="") then
     MsgBox.Error("Debe escribir una consulta","Error")
     Exit
   end

  '**EJECUTA LA CONSULTA
  TRANSACCION_SQL=
DLLProc.Make(_PasarelaDLL,"TRANSACCION_SQL",#DLLPROC_TYPE_STR,{#DLLPROC_TYPE_ST
R})
  Resultado = TRANSACCION_SQL.Call({consulta})

  '** CONSULTA NO EXITOSA
  if (not(Resultado = "Consulta exitosa"))  then
    '**PROBLEMAS EN LA CONEXION
    '**HABILITA LOS BOTONES
    '**av.Run("Habilitar_Botones",nil)
    MsgBox.Report(Resultado,"Error")
    Exit
  end

  '**HACE EL TRATAMIENTO DE LOS DATOS
  av.Run("GestionConsultas.Tratamiento_Datos",nil)

  Exit
end '**DEL METODO
```

```
'*********************************************************************
'***            METODO CONULSTAR CAMPOS                  **
'*********************************************************************
'** ENCARGADO DE HACER LA CONSULTA DE LOS CAMPOS DISPONIBLES
'** EN LA TABLA SELECCIONADA

if (Metodo="btn_Consultar_Campos") then

  Txt_Etiqueta = Dialogo.FindByName("txl_tabla")
  Tablas_Disponibles = Dialogo.FindByName("lbx_Tablas_Disponibles")
  valor = Tablas_Disponibles.GetCurrentValue

  '**EJECUTA LA CONSULTA
  CONSULTA_CAMPOS                                                                    =
DLLProc.Make(_PasarelaDLL,"CONSULTA_CAMPOS",#DLLPROC_TYPE_STR,{#DLLPROC_TYPE_S
TR})
  Resultado = CONSULTA_CAMPOS.Call({valor})

  '**SI LA CONSULTA NO ES EXITOSA SE SALE
  if(not(Resultado = "Consulta exitosa")) then
    '**Habilitamos los botones
    av.Run("Habilitar_Botones",nil)
    MsgBox.info (Resultado.asString,"")
    Dialogo.close
    Exit
  end

  '**FIJA LA ETIQUETA
  Txt_Etiqueta.SetLabel(valor)
  _Tabla_Sel=valor

  '**DESPLIEGA LOS DATOS EN EL CAMPO PERTINENTE
  Bufer_Datos = Filename.Make(_Ruta)
  theLineFile = LineFile.Make(Bufer_Datos, #FILE_PERM_READ)
  theLineFileSize = theLineFile.GetSize

  '**LIMPIAMOS EL CAMPO
  Lista_Campos = Dialogo.FindByName("lbx_Lista_Campos")
  Lista_Campos.Empty
  Lista_Campos.GoColumn(0)
  Lista_Campos.SetColumnWidth(3)

  for each rec in 1..theLineFileSize
    linea = theLineFile.readELT
    Lista_Campos.GoRow(Lista_Campos.GetRowCount)
    Lista_Campos.InsertRows(1)
    Lista_Campos.SetCurrentValue (linea)
  end

  Exit
end '**DEL METODO

'*********************************************************************
'***            METODO CERRAR EL DIALOGO                 **
'*********************************************************************
'** CIERRA EL DIALOGO
if (Metodo="btn_Cerrar") then
   Dialogo.Close
   Exit
end
```

```
'*****************************************************************
'***          METODO MINIMIZAR CAMPOS                  **
'*****************************************************************
'** MINIMIZA EL DIALOGO
if (Metodo="btn_Minimizar") then
   Dialogo.Minimize
   Exit
end

'*****************************************************************
'***          METODO CREAR CONSULTAS                   **
'*****************************************************************
'** COLABORA EN LA CREACION DE LAS CONSULTAS

'**INDEXA LOS CONTROLES Y FUNCIOES A LA CONSULTA
if (Metodo="btn_from") then Cadena = " FROM " end
if (Metodo="btn_select") then Cadena = " SELECT " end
if (Metodo="btn_insert") then Cadena = " INSERT INTO " end
if (Metodo="btn_Asterisco") then Cadena = " * " end
if (Metodo="btn_update") then Cadena = " UPDATE " end
if (Metodo="btn_set") then Cadena = " SET " end
if (Metodo="btn_where") then Cadena = " WHERE " end
if (Metodo="btn_igual") then Cadena = " = " end
if (Metodo="btn_mayor") then Cadena = " > " end
if (Metodo="btn_menor") then Cadena = " < " end
if (Metodo="btn_and") then Cadena = " AND " end
if (Metodo="btn_or") then Cadena = " OR " end
if (Metodo="btn_not") then Cadena = " NOT " end

'**BOTON TABLAS
'**ANEXA EL NOMBRE DE LA TABLA SELECCIONADA A LA CONSULTA
if (Metodo="btn_Tablas") then
  theListBox = Dialogo.FindByName("lbx_Tablas_Disponibles")
  Cadena = theListBox.GetCurrentValue
end

'**BOTON CAMPOS
'**ANEXA EL NOMBRE DEL CAMPO SELECCIONADO A LA CONSULTA
if (Metodo="btn_Campos") then
  theListBox = Dialogo.FindByName("lbx_Lista_Campos")
  Cadena = theListBox.GetCurrentValue
end

'** CONCATENA LA CONSULTA
consulta =  Consulta_SQL.getText
consulta = consulta ++ Cadena
Consulta_SQL.SetText(consulta)

'*************************************************************************
'***          METODO BORRAR CONSULTASQL                **
'*************************************************************************
'** BORRA LA EL CAMPO DE CONSULTA
if (Metodo="btn_borrar") then
  Consulta_SQL.SetText("")
  Exit
end
```

2.1.15 Script: GestiónTransaccionSQL.Inicializar_IS_TransacciónSQL.ave

```
'************************************************************************
'**METODO PUBLICO PERTENECIENTE A LA CLASE GESTION_TRANSACCION      **
'**SE ENCARGA DE INICIALIZAR LA INTERFAZ IS TRANSACCIONSQL          **
'************************************************************************

'**LLAMA AL DIALOGO
Dialogo = av.GetProject.FindDialog("IS_TransacciónSQL")

'**CENTRA EL DIALOGO
ptCenter = (av.ReturnOrigin + (av.ReturnExtent * (2@2))) - (Dialogo.ReturnExtent.ReturnSize * (2@2))
Dialogo.MoveTo(ptCenter.GetX, ptCenter.GetY)

'**LIMPIA LAS CAJAS DE TEXTO
Tablas_Disponibles = Dialogo.FindByName("lbx_Tablas_Disponibles")
Tablas_Disponibles.Empty
Tablas_Disponibles.GoColumn(0)
Tablas_Disponibles.SetColumnWidth(3)
Lista_Campos = Dialogo.FindByName("lbx_Lista_Campos")
Lista_Campos.Empty
Lista_Campos.GoColumn(0)
Lista_Campos.SetColumnWidth(3)

'**FIJA EL NOMBRE DE LA BASE DE DATOS CONECTADA
Dialogo.SetTitle("Base de datos:"++_BaseDatosActual)

'**DESPLIEGA LAS TABLAS DISPONIBLES
theLineFileSize = _Tablas.Count-1
for each rec in 0..theLineFileSize
  Tablas_Disponibles.GoRow(Tablas_Disponibles.GetRowCount)
  Tablas_Disponibles.InsertRows(1)
  Tablas_Disponibles.SetCurrentValue (_Tablas.Get(rec))
end

'**LIMPIA EL CAMPO DE CONSULTA
Consulta_SQL = Dialogo.FindByName("txbox_Consulta_SQL")
Consulta_SQL.SetText("")

'**ABRE EL DIALOGP
Dialogo.Open
```

2.1.16 Script:GestiónTransaccionSQL.Inicializar_IS_TransacciónSQL.ave

```
'************************************************************************************************
'** INICIALIZA LA INTERFAZ DE AYUDA DE LA APLICACION                **
'************************************************************************************************

'**UBICACION DE LA AYUDA
strfile = System.GetEnvVar("AVHOME")+"\Bin32\pasarela.chm"


'**NO HAY HARCHIVO DE AYUDA
if (NOT (File.Exists(strfile.AsFileName)))then
  msgbox.error("El archivo de ayuda no se encuentra. Revise la instalación","ERROR")
  exit
end
```

```
'LLAMA A LAS API DE WINDOWS PARA ABRIR EL ARCHIVO DE AYUDA
success = true
intHandle = DLL.GetAVWindowHandle
dllName = filename.findinsystemsearchPath("shell32.dll")
u32DLL = DLL.Make(dllName)
OpenFile=DLLProc.Make(u32DLL, "ShellExecuteA", #DLLPROC_TYPE_INT32,
        {#DLLPROC_TYPE_INT32, #DLLPROC_TYPE_STR, #DLLPROC_TYPE_STR,
        #DLLPROC_TYPE_STR, #DLLPROC_TYPE_STR, #DLLPROC_TYPE_INT32})
result = OpenFile.call({intHandle, "open", strFile, "", "C:\", 1})
if (result = 0) then
  success = false
end
return success
```

2.2 INTERFACES:

Las interfaces implementadas en la aplicación se encuentran detalladas en el capítulo IV de la Monografía.