```
                         foreach (string piece in pieces) {
Parts.Add(piece); }
                         int Count_max = Parts.Count - 1;
                         if (Parts[Count_max].ToString() == "")
                         {
                             Parts.RemoveAt(Count_max);
                         }

                         if (i == 2) { POQ =
Convert.ToInt32(Parts[1].ToString()); }
                         if (i == 3) { Lead_time =
Convert.ToInt32(Parts[1].ToString()); }
                     }
                 }

                 if (POQ > MRP_Table.Columns.Count - 2) //REVISAR QUE EL
POQ NO SEA MAYOR A LA CANTIDAD DE PERIODOS EXISTENTE
                 {
                     Lead_time = MRP_Table.Columns.Count - 2;
                 }
                 else
                 {
                     Lead_time = POQ;
                 }
                 //MessageBox.Show("POQ= " + POQ + "Lead time= " +
Lead_time);

                 for (int j = 1; j < MRP_Table.Columns.Count - 1; j++)
//DETERMINAR EN QUE POSICIONES HAY QUE SUMAR LOS REQUERIMIENTOS
                 {
                     ArrayList Positions = new ArrayList();
                     for (int i = 0; i < MRP_Table.Columns.Count - 1;
i++)
                     {
                         int Pos = i * Lead_time;
                         Pos = Pos + 1;
                         Positions.Add(Pos);
                     }

                     int Locate_here = 0;

                     foreach (object Pos in Positions)
                     {
                         int Pos_int = Convert.ToInt32(Pos);
                         if (j == Pos_int)
                         {
                             Locate_here = 1;
```

```
                    }
                }

                if (Locate_here == 1) //SI LA COLUMNA ES UNA
POSICION PARA SUMAR LOS REQUERIMIENTOS, REALIZAR LOS CALCULOS Y AÑADIR
LOS RESULTADOS
                {
                    int Sum = 0;
                    for (int k = 0; k < Lead_time; k++)
                    {
                        if (j + k <= MRP_Table.Columns.Count - 1)
                        {

                            string Value =
MRP_Table.Rows[Actual_row - 1][j + k].ToString();
                            if (Value == "") { Value = "0"; }

//MessageBox.Show(MRP_Table.Rows[Actual_row - 1][0].ToString() + Value);
                            Sum = Sum + Convert.ToInt32(Value);
                        }
                    }

                    MRP_Table.Rows[Actual_row - 1][j] =
Sum.ToString();
                    //
      MessageBox.Show(Sum.ToString());
                }
                else
                {
                    MRP_Table.Rows[Actual_row - 1][j] = "0";
                }

                string Val = MRP_Table.Rows[Actual_row -
2][j].ToString(); if (Val == "") { Val = "0"; }
                double Rec_program = Convert.ToDouble(Val);

                Val = MRP_Table.Rows[Actual_row - 1][j].ToString();
if (Val == "") { Val = "0"; }
                double Req_prod = Convert.ToDouble(Val);

                Val = MRP_Table.Rows[Actual_row - 5][j].ToString();
if (Val == "") { Val = "0"; }
                double Req_net = Convert.ToDouble(Val);

                double Inv_init_01 = Req_prod + Rec_program -
Req_net;

                if (j < MRP_Table.Columns.Count - 1)
```

```
                    {
                        MRP_Table.Rows[Actual_row - 6][j + 1] =
Inv_init_01.ToString();  //COLOCA EL INVENTARIO FINAL EN EL ACTUAL
PERIODO
                    }

                    if (j <= MRP_Table.Columns.Count - 2)
                    {
                        MRP_Table.Rows[Actual_row - 3][j + 1] =
Inv_init_01.ToString();

                        Val = MRP_Table.Rows[Actual_row - 4][j +
1].ToString(); if (Val == "") { Val = "0"; }
                        double Rec_prog_01 = Convert.ToDouble(Val);

                        double Rec_program_01 = Rec_prog_01 +
Inv_init_01;
                        MRP_Table.Rows[Actual_row - 2][j + 1] =
Rec_program_01.ToString();
                        //
      MessageBox.Show(Rec_program_01.ToString());
                    }
                }
            }
            catch (Exception Error)
            {
                F.Show_Error_Message(Error.Message, "Adecuate_to_POQ");
            }
        }

        private void EOQ_Storage_cost(DataTable Products_or_MP_table,
string Table_name, string MP_Product_name, bool Is_product, string
Row_name)
        {
            try
            {
                DataRow row = MRP_Table.NewRow();
                row[0] = Row_name;
                MRP_Table.Rows.Add(row);

                int Actual_row = MRP_Table.Rows.Count;


                //OBTENER EL COSTO DE PEDIR U ORDENAR
                if (Is_product == true)
                {
                    Table_name = "Productos";
                }
```

```
                  else
                  {
                      Table_name = "MP";
                  }

                  int Index_on_table = 0;
                  for (int l = 0; l < Products_or_MP_table.Rows.Count;
l++)
                  {
                      string Name_on_table =
Products_or_MP_table.Rows[l][1].ToString();
                      if (Name_on_table == MP_Product_name)
                      {
                          Index_on_table = l;
                      }
                  }

                  double Costo_de_pedir = 0;
                  double Costo_de_almacenar = 0;
                  double Demanda_anual = 0;
                  double Tamaño_predeterminado_lote = 0;

                  if (Is_product == true)
                  {
                      Costo_de_pedir =
Convert.ToDouble(Products_or_MP_table.Rows[Index_on_table][7].ToString()
);
                      Costo_de_almacenar =
Convert.ToDouble(Products_or_MP_table.Rows[Index_on_table][8].ToString()
);
                      Demanda_anual =
Convert.ToDouble(Products_or_MP_table.Rows[Index_on_table][10].ToString(
));
                      Tamaño_predeterminado_lote =
Convert.ToDouble(Products_or_MP_table.Rows[Index_on_table][11].ToString(
));
                  }
                  else
                  {
                      Costo_de_pedir =
Convert.ToDouble(Products_or_MP_table.Rows[Index_on_table][7].ToString()
);
                      Costo_de_almacenar =
Convert.ToDouble(Products_or_MP_table.Rows[Index_on_table][8].ToString()
);
                      Demanda_anual =
Convert.ToDouble(Products_or_MP_table.Rows[Index_on_table][9].ToString()
);
```

```
                     Tamaño_predeterminado_lote =
Convert.ToDouble(Products_or_MP_table.Rows[Index_on_table][10].ToString(
));
                 }

                 for (int j = 1; j < Numero_periodos; j++)
                 {
                     string Inv_end = MRP_Table.Rows[Actual_row -
9][j].ToString();
                     if (Inv_end == "") { Inv_end = "0"; }
                     int Inventario_inicial = Convert.ToInt32(Inv_end);

                     //$4/52 * 215 unidades

                     int Numero_de_periodos =
Convert.ToInt32(textBox6.Text);

                     double Storage_cost =
(Convert.ToDouble(Costo_de_almacenar) /
Convert.ToDouble(Numero_de_periodos)) *
Convert.ToDouble(Inventario_inicial);
                     Storage_cost = Math.Round(Storage_cost, 4);


                     MRP_Table.Rows[Actual_row - 1][j] = Storage_cost;
                     //MessageBox.Show("ST_cost: " +
Storage_cost.ToString() + "\r\n" +
                     // "costo almacenar: " + Costo_de_almacenar + "
Numero_de_periodos: " + Numero_de_periodos + " Inventario_inicial: " +
Inventario_inicial);

                 }
             }
             catch (Exception Error)
             {
                 F.Show_Error_Message(Error.Message,
"EOQ_Storage_cost");
             }
         }

         private void EOQ_Order_cost(DataTable Products_or_MP_table,
string Table_name, string MP_Product_name, bool Is_product, string
Row_name)
         {
             try
             {
                 DataRow row = MRP_Table.NewRow();
                 row[0] = Row_name;
```

```
                MRP_Table.Rows.Add(row);

                int Actual_row = MRP_Table.Rows.Count;


                //OBTENER EL COSTO DE PEDIR U ORDENAR
                if (Is_product == true)
                {
                    Table_name = "Productos";
                }
                else
                {
                    Table_name = "MP";
                }

                int Index_on_table = 0;
                for (int l = 0; l < Products_or_MP_table.Rows.Count;
l++)
                {
                    string Name_on_table =
Products_or_MP_table.Rows[l][1].ToString();
                    if (Name_on_table == MP_Product_name)
                    {
                        Index_on_table = l;
                    }
                }

                double Costo_de_pedir = 0;
                double Costo_de_almacenar = 0;
                double Demanda_anual = 0;
                double Tamaño_predeterminado_lote = 0;

                if (Is_product == true)
                {
                    Costo_de_pedir =
Convert.ToDouble(Products_or_MP_table.Rows[Index_on_table][7].ToString()
);
                    Costo_de_almacenar =
Convert.ToDouble(Products_or_MP_table.Rows[Index_on_table][8].ToString()
);
                    Demanda_anual =
Convert.ToDouble(Products_or_MP_table.Rows[Index_on_table][10].ToString(
));
                    Tamaño_predeterminado_lote =
Convert.ToDouble(Products_or_MP_table.Rows[Index_on_table][11].ToString(
));
                }
                else
```

```
                 {
                     Costo_de_pedir =
Convert.ToDouble(Products_or_MP_table.Rows[Index_on_table][7].ToString()
);
                     Costo_de_almacenar =
Convert.ToDouble(Products_or_MP_table.Rows[Index_on_table][8].ToString()
);
                     Demanda_anual =
Convert.ToDouble(Products_or_MP_table.Rows[Index_on_table][9].ToString()
);
                     Tamaño_predeterminado_lote =
Convert.ToDouble(Products_or_MP_table.Rows[Index_on_table][10].ToString(
));
                 }

                 for (int j = 1; j < Numero_periodos; j++)
                 {
                     string A = MRP_Table.Rows[Actual_row -
2][j].ToString();
                     if (A == "") { A = "0"; }
                     int Batch = Convert.ToInt32(A);


                     if (Batch > 0)
                     {
                         MRP_Table.Rows[Actual_row - 1][j] =
Costo_de_pedir;
                     }
                 }
             }
             catch (Exception Error)
             {
                 F.Show_Error_Message(Error.Message, "EOQ_Order_cost");
             }
         }

         private void Solve_level_MRP(ProgressBar Progbar)
         {
             try
             {
                 int Kase = 0;

                 // Create the columns in the MRP table
                 int Numb_periods = Get_the_number_of_periods();

                 //MOSTRAR EL NUMERO DE PERIODOS
                 textBox4.Text = Numb_periods.ToString();
```

```
                //CREAR LAS COLUMNAS EN LA TABLA DE MRP
                Create_colums_for_MRP(Numb_periods);

                Progbar.Minimum = 1;
                Progbar.Maximum = (Products_table.Rows.Count * 9) +
(MP_table.Rows.Count * 11);
                Progbar.Value = 1;
                Progbar.Step = 1;


                //PRODUCTS
                for (int j = 0; j < Products_table.Rows.Count; j++)
                {
                    Kase =
Convert.ToInt32(Products_table.Rows[j][12].ToString());  //obtiene la
politica de lotes
                    //MessageBox.Show("KASE: " + Kase);

                    DialogResult Result = new DialogResult();

                    if (Kase == 1) //LFL
                    {
                        Result = MessageBox.Show("En los lotes de
produccion de Lote por Lote (LFL) los costos de Almacenar se toman como
0 (cero).   Nota: Escogiendo NO se tomara el costo de ordenar definido
por el usuario. ¿Desea que sean 0?.", "Calculando (LFL)",
MessageBoxButtons.YesNo, MessageBoxIcon.Warning);
                    }


                    string MP_Product_name =
Products_table.Rows[j][1].ToString();
                    int Lead_time =
Convert.ToInt32(Products_table.Rows[j][6].ToString());


                    // ADD TITTLE
                    Add_Tittle_row(MP_Product_name);
Progbar.PerformStep();
                    // ADD DEMAND
                    Add_Dat_row_to_table(Demand_table, MP_Product_name,
"Demanda X Pedido"); Progbar.PerformStep();
                    Add_Zero_row_to_table("+ Demanda X Produccion");
Progbar.PerformStep();
                    // SUM Demanda Pedido + Demanda Produccion
                    Summatory_of_rows("= Demanda");
Progbar.PerformStep();
                    // ADD FNAL PERIOD INVENTORY
```

```
                    Add_Dat_row_to_table(Inventary_forecast_table,
MP_Product_name, "+ Inventario al final del periodo");
Progbar.PerformStep();
                    // SUM DEMAND + FINAL PEDIOD INVENTORY
                    Summatory_of_rows("= RQN Requerimientos netos");
Progbar.PerformStep();

                    //CALCULATE EOQ
                    if (Kase == 2)
                    {
                        int eoq = Calculate_EOQ(Products_table,
Demand_table, MP_Product_name, true, "$ (EOQ) Costo de Almacenamiento",
false);
                        if (eoq == 0) { Kase = 1; } //Si no hay valor de
TLP, se usa LFL // MessageBox.Show(MP_Product_name + " EOQ_: " +
eoq.ToString());
                    }

                    if (Kase == 5)
                    {
                        int eoq = Calculate_EOQ(Products_table,
Demand_table, MP_Product_name, true, "$ (EOQ) Costo de Almacenamiento",
true);
                        //if (eoq == 0) { Kase = 1; } //Si no hay valor
de TLP, se usa LFL // MessageBox.Show(MP_Product_name + " EOQ_: " +
eoq.ToString());
                    }

                    //CALCULATE POQ
                    if (Kase == 3) { Calculate_POQ(Products_table,
Demand_table, MP_Product_name, true, "$ (EOQ) Costo de Almacenamiento");
}

                    // ADD ENTER DELIVERIES
                    Add_Dat_row_to_table(Orders_2_receiv_table,
MP_Product_name, "Recursos programados"); Progbar.PerformStep();
                    // ADD INITIAL INVENTORY
                    Add_Initial_inventory(Products_table,
MP_Product_name, "+ Inventario inicial"); Progbar.PerformStep();
//Add_Initial_inventory("Productos", MP_Product_name, "+ Inventario
inicial"); Progbar.PerformStep();
                    // SUM ENTER DELIVERIES + INITIAL INVENTORY
                    Summatory_of_rows("= RPN Recursos programados
netos"); Progbar.PerformStep();
                    // SUBTRACT TOTALS
                    if (Kase < 2 || Kase == 5)
                    {
                        if (this.checkBox1.Checked == true)
```

```
                        {
                            Substract_of_rows_NO_INV("Produccion
requerida (RPN - RQN)");
                        }
                        else
                        {
                            Substract_of_rows("Produccion requerida (RPN
- RQN)   ---->");
                        } Progbar.PerformStep();
                    }
                    else
                    {
                        Substract_of_rows("Produccion requerida (RPN -
RQN)  ---->");

                        Progbar.PerformStep();
                    }


                    //ADECUATE TO EOQ
                    if (Kase == 2) { Adecuate_to_EOQ(); }

                    //ADECUATE TO POQ
                    if (Kase == 3) { Adecuate_to_POQ(); }

                    if (Kase == 4)
                    {
                        int TPL =
Convert.ToInt32(Products_table.Rows[j][11].ToString());
                        Adecuate_to_TLP(TPL, Lead_time);
                    }

                    if (Kase == 5)
                    {
                        int TPL =
Convert.ToInt32(Products_table.Rows[j][11].ToString());
                        Adecuate_to_TML(TPL, Lead_time);
                    }

                    //LOCATE REQUERIMENTS
                    //if (checkBox4.Checked == true)
                    if (Kase == 5)
                    {
                        int TPL =
Convert.ToInt32(Products_table.Rows[j][11].ToString());  //obtiene el
tamaño predeterminado de lote
                        Locate_requeriments("Colocacion de
requerimientos", Lead_time, TPL); Progbar.PerformStep();
                    }
```

```
                    else
                    {
                        Locate_requeriments("Colocacion de 
requerimientos", Lead_time); Progbar.PerformStep();
                    }

                    if (Kase == 1)
                    {
                        // CALCULATE STORAGE COST (LFL)
                        if (Result == DialogResult.No)
                        {
                            LFL_Storage_cost(Products_table, 
"Productos", MP_Product_name, "$ (LFL)Costo de Almacenamiento");
                        }
                        else
                        {
                            LFL_Storage_cost_0("$ (LFL)Costo de 
Almacenamiento");
                        }
                        // CALCULATE ORDER COST (LFL)
                        LFL_Order_cost(Products_table, "Productos", 
MP_Product_name, "$ (LFL)Costo de ordenar");
                        // CALCULATE TOTAL COS (LFL)
                        Summatory_of_rows("= (LFL)Total de costos"); 
progressBar2.PerformStep();
                        Summatory_of_columns();
                    }

                    if (Kase == 2)
                    {
                        //CALCULA LOS COSTOS DE COLOCACION DE PEDIDOS
                        EOQ_Order_cost(Products_table, "Productos", 
MP_Product_name, true, "$ (EOQ)Costo de pedir");
                        Summatory_of_columns();
                        EOQ_Storage_cost(Products_table, "Productos", 
MP_Product_name, true, "$ (EOQ)Costo de almacenar");
                        Summatory_of_columns();

                        Summatory_of_rows("= (EOQ)Total de costos"); 
progressBar2.PerformStep();
                        Summatory_of_columns();
                    }

                    if (Kase == 3)
                    {
                        //CALCULA LOS COSTOS DE COLOCACION DE PEDIDOS
                        EOQ_Order_cost(Products_table, "Productos", 
MP_Product_name, true, "$ (POQ)Costo de pedir");
```

```
                        Summatory_of_columns();
                        EOQ_Storage_cost(Products_table, "Productos",
MP_Product_name, true, "$ (POQ)Costo de almacenar");
                        Summatory_of_columns();

                        Summatory_of_rows("= (POQ)Total de costos");
progressBar2.PerformStep();
                        Summatory_of_columns();
                    }
                }

                //string the_list = "";
                ArrayList MP_listed_in_levels_order =
Get_MP_Ordered_list();  //OBTIENE LA LISTA DE MATERIAS EN EL ORDEN
CORRECTA PARA CALCULAR LOS REQUERIMIENTOS SEGUN LA POSICION EN LOS
ARBOLES DE PRODUCTO

                //MP
                //for (int j = 0; j < MP_table.Rows.Count; j++)
                for (int j = 0; j < MP_listed_in_levels_order.Count;
j++)
                {
                    string MP_Product_name =
MP_listed_in_levels_order[j].ToString();   //OBTIENE EL NOMBRE DE LA MP
A CALCULAR
                    int Lead_time = 0;
                    int TPL = 0;
                    DialogResult Result = new DialogResult();

                    //OBTENER EL LEAD TIME PARA LA MP
                    for (int k = 0; k < MP_table.Rows.Count; k++)
//RECORRE LA TABLA DE MP PARA OBTENER EL LEAD TIME CORRESPONDIENTE YA
QUE EN LA TABLA DE MP ESTAS NO ESTAN EN EL ORDEN CORRECTO
                    {
                        string MP_table_name =
MP_table.Rows[k][1].ToString();
                        if (MP_table_name == MP_Product_name)
//COMPARA EL NOMBRE DE LA MP SELECCIONADA CON EL NOMBRE DE LA MP EN LA
TABLA
                        {
                            Lead_time =
Convert.ToInt32(MP_table.Rows[k][6].ToString()); //ASOCIA EL LEAD TIME
EN LA TABLA CON EL LEAD TIME PARA LOS CALCULOS

                            Kase =
Convert.ToInt32(MP_table.Rows[k][11].ToString());  //obtiene la politica
de lotes
```

```
                            if (Kase == 1) //LFL
                            {
                                Result = MessageBox.Show("En los lotes
de produccion de Lote por Lote (LFL) los costos de Almacenar se toman
como 0 (cero).   Nota: Escogiendo NO se tomara el costo de ordenar
definido por el usuario. ¿Desea que sean 0?.", "Calculando (LFL)",
MessageBoxButtons.YesNo, MessageBoxIcon.Warning);
                            }

                            TPL =
Convert.ToInt32(MP_table.Rows[k][10].ToString());  //obtiene el tamaño
predeterminado de lote
                        }
                    }

                    // ADD TITTLE
                    Add_Tittle_row(MP_Product_name);
Progbar.PerformStep();
                    // ADD DEMAND
                    Add_Dat_row_to_table(Demand_table, MP_Product_name,
"Demanda X Pedido"); Progbar.PerformStep();
                    Add_MP_Dat_row_to_table(Products_table, MP_table,
MP_Product_name, "+ Demanda X Produccion"); Progbar.PerformStep();
                    // SUM Demanda Pedido + Demanda Produccion
                    Summatory_of_rows("= Demanda");
Progbar.PerformStep();
                    // ADD FNAL PERIOD INVENTORY
                    Add_Dat_row_to_table(Inventary_forecast_table,
MP_Product_name, "+ Inventario al final del periodo");
Progbar.PerformStep();
                    // SUM DEMAND + FINAL PEDIOD INVENTORY
                    Summatory_of_rows("= RQN Requerimientos netos");
Progbar.PerformStep();

                    //CALCULATE EOQ
                    if (Kase == 2)
                    {
                        int eoq = Calculate_EOQ(MP_table, Demand_table,
MP_Product_name, false, "$ (EOQ) Costo de Almacenamiento", false);
                        if (eoq == 0) { Kase = 1; } //Si no hay valor de
TLP, se usa LFL
                    }

                    if (Kase == 5)
                    {
                        int eoq = Calculate_EOQ(MP_table, Demand_table,
MP_Product_name, false, "$ (EOQ) Costo de Almacenamiento", true);
```

```
                        //if (eoq == 0) { Kase = 1; } //Si no hay valor
de TLP, se usa LFL
                    }

                    //CALCULATE EOQ
                    if (Kase == 3) { Calculate_POQ(MP_table,
Demand_table, MP_Product_name, false, "$ (EOQ) Costo de
Almacenamiento"); }


                    // ADD ENTER DELIVERIES
                    Add_Dat_row_to_table(Orders_2_receiv_table,
MP_Product_name, "Recursos programados"); Progbar.PerformStep();
                    // ADD INITIAL INVENTORY
                    Add_Initial_inventory(MP_table, MP_Product_name, "+
Inventario inicial"); Progbar.PerformStep();
//Add_Initial_inventory("MP", MP_Product_name, "+ Inventario inicial");
Progbar.PerformStep();
                    // SUM ENTER DELIVERIES + INITIAL INVENTORY
                    Summatory_of_rows("= RPN Recursos programados
netos"); Progbar.PerformStep();


                    // SUBTRACT TOTALS
                    if (Kase < 2 || Kase == 5)
                    {
                        if (this.checkBox1.Checked == true)
                        {
                            Substract_of_rows_NO_INV("Produccion
requerida (RPN - RQN)");
                        }
                        else
                        {
                            Substract_of_rows("Produccion requerida (RPN
- RQN)   ---->");
                        } Progbar.PerformStep();
                    }
                    else
                    {
                        Substract_of_rows("Produccion requerida (RPN -
RQN)  ---->");
                        Progbar.PerformStep();
                    }

                    //ADECUATE TO EOQ
                    if (Kase == 2) { Adecuate_to_EOQ(); }

                    //ADECUATE TO POQ
```

```
                    if (Kase == 3) { Adecuate_to_POQ(); }

                    if (Kase == 4) { Adecuate_to_TLP(TPL, Lead_time); }

                    if (Kase == 5) { Adecuate_to_TML(TPL, Lead_time); }


                    // LOCATE REQUERIMENTS

                    if (Kase == 5)
                    {

                        //MessageBox.Show(MP_Product_name + " " + TPL);
                        Locate_requeriments("Colocacion de
requerimientos", Lead_time, TPL); Progbar.PerformStep();
                    }
                    else
                    {
                        Locate_requeriments("Colocacion de
requerimientos", Lead_time); Progbar.PerformStep();
                    }

                    if (Kase == 1)
                    {
                        // CALCULATE STORAGE COST (LFL)
                        if (Result == DialogResult.No)
                        {
                            LFL_Storage_cost(MP_table, "MP",
MP_Product_name, "$ (LFL)Costo de Almacenamiento");
                        }
                        else
                        {
                            LFL_Storage_cost_0("$ (LFL)Costo de
Almacenamiento");
                        }
                        // CALCULATE ORDER COST (LFL)
                        LFL_Order_cost(MP_table, "MP", MP_Product_name,
"$ (LFL)Costo de ordenar");
                        // CALCULATE TOTAL COS (LFL)
                        Summatory_of_rows("= (LFL)Total de costos");
progressBar2.PerformStep();
                        Summatory_of_columns();
                    }

                    if (Kase == 2)
                    {
                        //CALCULA LOS COSTOS DE COLOCACION DE PEDIDOS
```

```
                        EOQ_Order_cost(MP_table, "MP", MP_Product_name,
false, "$ (EOQ)Costo de pedir");
                        Summatory_of_columns();
                        EOQ_Storage_cost(MP_table, "MP",
MP_Product_name, false, "$ (EOQ)Costo de almacenar");
                        Summatory_of_columns();

                        Summatory_of_rows("= (EOQ)Total de costos");
progressBar2.PerformStep();
                        Summatory_of_columns();
                    }

                    if (Kase == 3)
                    {
                        //CALCULA LOS COSTOS DE COLOCACION DE PEDIDOS
                        EOQ_Order_cost(MP_table, "MP", MP_Product_name,
false, "$ (POQ)Costo de pedir");
                        Summatory_of_columns();
                        EOQ_Storage_cost(MP_table, "MP",
MP_Product_name, false, "$ (POQ)Costo de almacenar");
                        Summatory_of_columns();

                        Summatory_of_rows("= (POQ)Total de costos");
progressBar2.PerformStep();
                        Summatory_of_columns();
                    }
                }

                dataGridView3.DataSource = MRP_Table;
                dataGridView3.AutoResizeColumns();


Select_a_Product_to_calculate_requirements(Products_table, "Productos",
comboBox1.SelectedItem.ToString(), false); //AS THE TREEVIEW1 IS USED
FOR CALCULES IS NECESARY REFRESH THE DATA IN THE TAB1


            }
            catch (Exception Error)
            {
                F.Show_Error_Message(Error.Message, "Solve_level_MRP");
            }
        }


        private void button2_Click(object sender, EventArgs e)
        {
            Solve_level_MRP(progressBar1); //0 Ordinary MRP
```

```
        }

        private void LFL_Storage_cost(DataTable Products_or_MP_table,
string Table_name, string Product_name, string Row_name)
        {
            try
            {
                DataRow row = MRP_Table.NewRow();
                row[0] = Row_name;
                MRP_Table.Rows.Add(row);

                int Actual_row = MRP_Table.Rows.Count;

                int Index_on_table_of_product =
F.Get_on_Access_Table_Index_of(Form1.Current_MRP, Table_name,
Product_name);
                string Readed_value =
Products_or_MP_table.Rows[Index_on_table_of_product][8].ToString();
                if (Readed_value == "") { Readed_value = "0"; }
                double Storage_cost_per_unit =
Convert.ToDouble(Readed_value);

                for (int j = 1; j < Numero_periodos; j++)
                {
                    string A = MRP_Table.Rows[Actual_row -
5][j].ToString();
                    if (A == "") { A = "0"; }
                    int Batch = Convert.ToInt32(A);

                    double Storage_cost = Storage_cost_per_unit * Batch;
                    if (Batch < 0)
                    {
                        Storage_cost = 0;
                    }

                    MRP_Table.Rows[Actual_row - 1][j] =
Storage_cost.ToString();
                }
            }
            catch (Exception Error)
            {
                F.Show_Error_Message(Error.Message, "LFL_Storage_cost");
            }
        }

        private void LFL_Storage_cost_0(string Row_name)
        {
            try
```

```
            {
                DataRow row = MRP_Table.NewRow();
                row[0] = Row_name;
                MRP_Table.Rows.Add(row);

                int Actual_row = MRP_Table.Rows.Count;

                string Readed_value = "0";

                for (int j = 1; j < Numero_periodos; j++)
                {
                    MRP_Table.Rows[Actual_row - 1][j] =
Readed_value.ToString();
                }
            }
            catch (Exception Error)
            {
                F.Show_Error_Message(Error.Message,
"LFL_Storage_cost_0");
            }
        }

        private void LFL_Order_cost(DataTable Products_or_MP_table,
string Table_name, string Product_name, string Row_name)
        {
            try
            {
                DataRow row = MRP_Table.NewRow();
                row[0] = Row_name;
                MRP_Table.Rows.Add(row);

                int Actual_row = MRP_Table.Rows.Count;
                int Index_on_table_of_product =
F.Get_on_Access_Table_Index_of(Form1.Current_MRP, Table_name,
Product_name);
                string Readed_value =
Products_or_MP_table.Rows[Index_on_table_of_product][7].ToString();

                for (int j = 1; j < Numero_periodos; j++)
                {
                    string A = MRP_Table.Rows[Actual_row -
3][j].ToString();
                    if (A == "") { A = "0"; }
                    int Batch = Convert.ToInt32(A);
                    if (Batch == 0) { Readed_value = "0"; }

                    MRP_Table.Rows[Actual_row - 1][j] =
Readed_value.ToString();
```

```
                }
            }
            catch (Exception Error)
            {
                F.Show_Error_Message(Error.Message, "LFL_Order_cost");
            }
        }

        private void Summatory_of_columns()
        {
            try
            {
                int Actual_row = MRP_Table.Rows.Count;
                double Sum = 0;

                for (int j = 1; j < Numero_periodos; j++)
                {
                    string Value = MRP_Table.Rows[Actual_row -
1][j].ToString();
                    if (Value == "") { Value = "0"; }
                    double Batch = Convert.ToDouble(Value);

                    Sum = Sum + Batch;
                    Sum = Math.Round(Sum, 4);
                }

                MRP_Table.Rows[Actual_row - 1][Numero_periodos] =
Sum.ToString();
            }
            catch (Exception Error)
            {
                F.Show_Error_Message(Error.Message,
"Summatory_of_columns");
            }
        }

        private void button3_Click(object sender, EventArgs e)
        {
            try
            {
                ArrayList Columns = new ArrayList();
                ArrayList Values = new ArrayList();

                Columns.Clear();
                Values.Clear();

                string Table_name = "MP";
```

```
                if (tabControl3.SelectedIndex == 0)
                {
                    Table_name = "MP";
                    Columns.Add("MP");
                    Values.Add(comboBox4.SelectedIndex.ToString());
                }

                if (tabControl3.SelectedIndex == 1)
                {
                    Table_name = "Productos";
                    Columns.Add("Productos");
                    Values.Add(comboBox5.SelectedIndex.ToString());
                }


                Columns.Add("Politica_dimensionamiento_lote");
                Values.Add(comboBox6.SelectedIndex.ToString());


                F.UpdateDataToAccessTable(Form1.Current_MRP, Table_name,
Columns, Values, Indx_to_change_batch_policy.ToString());

            }
            catch (Exception Error)
            {
                F.Show_Error_Message(Error.Message, "button3_Click");
            }
        }

        private void textBox6_KeyPress(object sender, KeyPressEventArgs
e)
        {
            // SOLO PERMITE INTRODUCIR Y BORRAR DATOS NUMERICOS EN EL
TEXTBOX2
            try
            {
                if (F.this_char_is_a_number_or_backspace(e.KeyChar) ==
true)
                {
                    e.Handled = true;
                }
            }
            catch (Exception Error)
            {
                F.Show_Error_Message(Error.ToString(), "Creando una
nueva unidad -> Filtrando teclas en el teclado para TextBox2");
            }
        }
```

```
        private void button4_Click(object sender, EventArgs e)
        {
            //Solve_level_MRP(progressBar3);
        }

        private void refresh_calculation_first_tab()
        {
            try
            {
                dataGridView1.DataSource =
Calculate_quantity_requierements(Convert.ToInt32(textBox1.Text));
                dataGridView2.DataSource =
Calculate_Capacity_requierements();
            }
            catch (Exception Error)
            {
                F.Show_Error_Message(Error.Message,
"refresh_calculation_first_tab");
            }
            //dataGridView4.DataSource =
Calculate_req_qntty_lead(treeView1, Convert.ToInt32(textBox1.Text));
        }

        private void toolStripButton5_Click(object sender, EventArgs e)
        {
            try
            {
                //Added_Nodes.Clear();

                dataGridView1.DataSource =
Calculate_quantity_requierements(Convert.ToInt32(textBox1.Text));
                dataGridView2.DataSource =
Calculate_Capacity_requierements();
                dataGridView4.DataSource =
Calculate_req_qntty_lead(treeView1, Convert.ToInt32(textBox1.Text));

                //MessageBox.Show(list);
            }
            catch (Exception Error)
            {
                F.Show_Error_Message(Error.Message,
"toolStripButton5_Click");
            }
        }

#endregion
```

```
        private void Add_row_to_capacity_table()
        {
            try
            {
                DataRow row = Capacity_table.NewRow();
                Capacity_table.Rows.Add(row);

                row = Capacity_table_use_percent.NewRow();
                Capacity_table_use_percent.Rows.Add(row);
            }
            catch (Exception Error)
            {
                F.Show_Error_Message(Error.Message,
"Add_row_to_capacity_table");
            }
        }

        private void Add_row_to_Workstations_download_table()
        {
            try
            {
                DataRow row = WorkStations_download.NewRow();
                WorkStations_download.Rows.Add(row);
            }
            catch (Exception Error)
            {
                F.Show_Error_Message(Error.Message,
"Add_row_to_Workstations_download_table");
            }
        }

        private void Calculate_capacity(DataTable Products_table,
DataTable Operations_table, DataTable Storages_table, ProgressBar
ProgBar)
        {
            try
            {
                int Number_of_columns = MRP_Table.Columns.Count;
                int Actual_row_capacity;
                for (int i = 0; i < Number_of_columns; i++) {
Capacity_table.Columns.Add(i.ToString());
Capacity_table_use_percent.Columns.Add(i.ToString()); } // CREA LAS
COLUMNAS PARA LA TABLA DE CALCULOS DE CAPACIDAD

                ProgBar.PerformStep();

                //CREA LA LISTA DE PRODUCTOS
                ArrayList Products_list = new ArrayList();
```

```
                for (int i = 0; i < Products_table.Rows.Count; i++)
//LEE LA TABLA DE PRODUCTOS
                {
                    string Product_name =
Products_table.Rows[i][1].ToString(); //OBTIENE EL NOMBRE DEL PRODUCTO
                    Products_list.Add(Product_name);
                }

                ProgBar.PerformStep();

                //RESUME LA COLOCACION DE PEDIDOS PARA CADA PRODUCTO
                for (int j = 0; j < MRP_Table.Rows.Count - 1; j++)
                {
                    string Label_of_row =
MRP_Table.Rows[j][0].ToString();

                    if (Products_list.Contains(Label_of_row))
//SI ENCUENTRA EL NOMBRE DEL PRODUCTO EN LA TABLA DEL MRP
                    {
                        //MessageBox.Show(Label_of_row);
                        Actual_row_capacity = Capacity_table.Rows.Count;
                        Add_row_to_capacity_table();
                        Capacity_table.Rows[Actual_row_capacity][0] =
Label_of_row;   //AGREGA LA ETIQUETA PARA LA FILA DEL PRODUCTO

Capacity_table_use_percent.Rows[Actual_row_capacity][0] = Label_of_row;
//AGREGA LA ETIQUETA PARA LA FILA DEL PRODUCTO


                        string val = MRP_Table.Rows[j +
10][0].ToString();
                        //if (val == "") { val = "0"; }

                        if (val == "Colocacion de requerimientos")
                        {
                            Add_row_to_capacity_table();
                            for (int k = 0; k < MRP_Table.Columns.Count;
k++)
                            {

                                Actual_row_capacity =
Capacity_table.Rows.Count;
                                string Value = MRP_Table.Rows[j +
10][k].ToString();
                                if (Value == "") { Value = "0"; }

                                Capacity_table.Rows[Actual_row_capacity
- 1][k] = Value;
```

```
Capacity_table_use_percent.Rows[Actual_row_capacity - 1][k] = Value;
                            }
                        }
                    }
                }

                ProgBar.PerformStep();

                //CALCULA LOS REQUERIMIENTOS DE MANO DE OBRA EN CADA
PUESTO DE TRABAJO
                Add_row_to_capacity_table();
                Add_row_to_capacity_table();
                Actual_row_capacity = Capacity_table.Rows.Count;
                Capacity_table.Rows[Actual_row_capacity - 1][0] =
"Requerimientos de capacidad de Mano de Obra";
                Capacity_table_use_percent.Rows[Actual_row_capacity -
1][0] = "Requerimientos de capacidad de Mano de Obra";

                //CREA LA LISTA DE LAS OPERACIONES Y LOS ALMACENAMIENTOS
                ArrayList Items_list = new ArrayList();
                ArrayList Operations_list = new ArrayList();
                ArrayList Storege_list = new ArrayList();

                ProgBar.PerformStep();

                for (int i = 0; i < Operations_table.Rows.Count; i++)
//LEE LA TABLA DE OPERACIONES
                {
                    string Operation_name =
Operations_table.Rows[i][1].ToString(); //OBTIENE EL NOMBRE DE LA
OPERACION Y LOS AGREGA A LA LISTA DE ITEMS
                    if (!Items_list.Contains(Operation_name))
                    {
                        Items_list.Add(Operation_name);
                        Operations_list.Add(Operation_name);
                    }
                }

                ProgBar.PerformStep();


                for (int i = 0; i < Storage_table.Rows.Count; i++)
//LEE LA TABLA DE ALMACENAMIENTOS
                {
                    string Storage_name =
Storage_table.Rows[i][1].ToString(); //OBTIENE EL NOMBRE DEL
ALMACENAMIENTO Y LOS AGREGA A LA LISTA DE ITEMS
```

```
                    if (!Items_list.Contains(Storage_name))
                    {
                        Items_list.Add(Storage_name);
                        Storege_list.Add(Storage_name);
                    }

                }

                ProgBar.PerformStep();

                //RECORRE LA LISTA DE OPERACIONES Y ALMACENAMIENTOS Y
LOS AGREGA A LA TABLA
                for (int i = 0; i < Items_list.Count; i++)
                {
                    ArrayList Workstations_list = new ArrayList();
                    string Item_name = Items_list[i].ToString();

                    if (!Workstations_list.Contains(Item_name))
                    {
                        Add_row_to_capacity_table();
                        Actual_row_capacity = Capacity_table.Rows.Count;
                        Capacity_table.Rows[Actual_row_capacity - 1][0]
= Item_name;

Capacity_table_use_percent.Rows[Actual_row_capacity - 1][0] = Item_name;

                        ////MessageBox.Show(Item_name + " (" +
Item_table.Rows[Index_on_table][4].ToString() + ")" + "|" + Item_name +
" (" + Item_table.Rows[Index_on_table][6].ToString() + ")");
                        Workstations_list.Add(Item_name);
                    }
                }

                Add_row_to_capacity_table();
                Add_row_to_capacity_table();
                Actual_row_capacity = Capacity_table.Rows.Count;
                Capacity_table.Rows[Actual_row_capacity - 1][0] =
"Requerimientos de Capacidad de Maquinaria";
                Capacity_table_use_percent.Rows[Actual_row_capacity -
1][0] = "Requerimientos de Capacidad de Maquinaria";

                //RECORRE LA LISTA DE OPERACIONES Y ALMACENAMIENTOS
                for (int i = 0; i < Items_list.Count; i++)
                {
                    string Item_name = Items_list[i].ToString();
                    Add_row_to_capacity_table();
                    Actual_row_capacity = Capacity_table.Rows.Count;
```

```
                    Capacity_table.Rows[Actual_row_capacity - 1][0] =
Item_name;
                    Capacity_table_use_percent.Rows[Actual_row_capacity
- 1][0] = Item_name;
                }

                ProgBar.PerformStep();

                for (int i = 0; i < comboBox1.Items.Count; i++)
//RECORRE LA LISTA DE PRODUCTOS EN EL COMBOBOX1
                {
                    comboBox1.SelectedIndex = i;
                    string Product_name =
comboBox1.SelectedItem.ToString(); // MessageBox.Show(Product_name);
//PARA CADA PRODUCTO


                    for (int Col = 1; Col < Capacity_table.Columns.Count
- 1; Col++) //RECORRE LAS COLUMNAS DE LA TABLA DE CALCULOS DE CAPACIDAD
                    {
                        int Act_row =
Detect_row_number_of(Capacity_table, comboBox1.Items[i].ToString()); //
MessageBox.Show(Product_name + " " + Act_row); //OBTIENE EL ROW DONDE
ESTA EL PRODUCTO LEIDO EN EL COMBOBOX1

                        string Val = Capacity_table.Rows[Act_row +
1][Col].ToString(); //LEE LA COLOCACION DE PEDIDOS PARA EL PRODUCTO DEL
COMBOBOX1
                        //MessageBox.Show(Product_name + " " + Val); //
//string Val = Capacity_table.Rows[Act_row][Col].ToString(); //LEE LA
COLOCACION DE PEDIDOS PARA EL PRODUCTO DEL COMBOBOX1

                        this.textBox1.Text = Val;  //CARGA LA CANTIDAD
EN EL TEXTBOX1 PARA CALCULAS LOS REQUERIMIENTOS DE CAPACIDAD

                        string Product =
comboBox1.SelectedItem.ToString();
                        //ACTUALIZA LAS TABLAS Y EL TREEVIEW CON LOS
DATOS DEL PRODUCTO SELECCIONADO PARA PODER REALIZAR LOS CALCULOS CD
CAPACIDAD

Select_a_Product_to_calculate_requirements(Products_table, "Productos",
Product, true);  //SELECCIONA UN PRODUCTO PARA REALIZAR LOS CALCULOS DE
CAPACIDAD

                        refresh_calculation_first_tab();  //REFRESCA LAS
TABLAS EN LAS QUE APARECEN LOS CALCULOS DE CAPACIDAD REQUERIDA PARA EL
PRODUCTO SELECCIONADO EN LA PESTAÑA 1
```

```
                        //DataTable CREQ = new DataTable();
                        var Creq = dataGridView2.DataSource as
DataTable;  // OBTIENE LOS RESULTADOS DE LOS CALCULOS DE CAPACIDAD DEL
DATAGRID2 Y LOS UBICA EN LA TABLA Creq
                        int Number_of_colums = Creq.Columns.Count;
                        int Number_of_rows = Creq.Rows.Count;

                        for (int _h = 0; _h < Number_of_rows; _h++)
//lee las filas de los calculos de capacidad
                        {
                            for (int _i = 0; _i < Number_of_colums;
_i++) //lee las columnas de los calculos de capacidad
                            {
                                if (_i == 1)  //Si es la primera columna
== MO
                                {

                                    int MO_row =
Detect_row_number_of(Capacity_table, "Requerimientos de capacidad de
Mano de Obra");
                                    int Operation_row =
Detect_row_number_of__from_(Capacity_table, Creq.Rows[_h][0].ToString(),
MO_row); //Obtiene el row de la operacion actual
//MessageBox.Show(Creq.Rows[_h][0].ToString() + " " +
Operation_row.ToString());

                                    string Readed_val_00 =
Capacity_table.Rows[Operation_row][Col].ToString();
                                    if (Readed_val_00 == "") {
Readed_val_00 = "0"; }
                                    double A =
Convert.ToDouble(Readed_val_00); //LEE LOS REQUERIMIENTOS RE CAPACIDAD
ACUMULADOS
                                    double B =
Convert.ToDouble(Creq.Rows[_h][1].ToString()); //LEE LOS RESULTADOS DE
CALCULOS DE CAPACIDAD
                                    double C = A + B; //SUMA EL
ACUMULADO CON EL CALCULO
                                    //double Vertical_Sumatory =
Vertical_Sumatory + C;


Capacity_table.Rows[Operation_row][Col] = C.ToString();

                                    //CALCULA EL PORCENTAJE DE
UTILIZACION
```

```
                                    double D = 0;



                                    string Readed_val_01 = 
Capacity_table.Rows[Operation_row][0].ToString();  //NOMBRE DE LA 
OPERACION O CENTRO DE TRABAJO
                                    if (Readed_val_01 == "") { 
Readed_val_01 = "0"; }

                                    //SELECCIONAR LOS DATOS CORRECTOS 
PARA BUSCAR LA CAPACIDAD MAXIMA PARA EL PUESTO DE TRABAJO EN EL PERIODO
                                    ArrayList Columns = new ArrayList(); 
Columns.Add("Operaciones");
                                    ArrayList Values = new ArrayList(); 
Values.Add(Readed_val_01);
                                    bool Is_Operation = 
F.Exist_row_on_Access_Table(Form1.Current_MRP, "Operaciones", Columns, 
Values);

                                    Columns = new ArrayList(); 
Columns.Add("Almacenamientos");
                                    Values = new ArrayList(); 
Values.Add(Readed_val_01);
                                    bool Is_Storage = 
F.Exist_row_on_Access_Table(Form1.Current_MRP, "Almacenamientos", 
Columns, Values); DataTable Worstations_table = new DataTable();

                                    if (Is_Operation) { 
Worstations_table = Operations_table; }
                                    if (Is_Storage) { Worstations_table 
= Storages_table; }


                                    //FIND THE MAXIMUM CAPACITY FOR THE 
WORKSTARION IN THE PERIOD  Operations_table, DataTable Storages_table
                                    double Capacity_MO_for_this_period = 
0;
                                    for (int x = 0; x < 
Worstations_table.Rows.Count; x++)
                                    {
                                        string Operation_name = 
Worstations_table.Rows[x][1].ToString();
                                        int Period = 
Convert.ToInt32(Worstations_table.Rows[x][5].ToString());

                                        if (Col == Period && 
Operation_name == Readed_val_01)
```

```
                                    {
                                        Capacity_MO_for_this_period
= Convert.ToDouble(Worstations_table.Rows[x][6].ToString());
//cAPACIDAD MO
                                    }
                                }

                                //MessageBox.Show("OP" +
Is_Operation + " ST " + Is_Storage +  "Periodo: " + Col + "Puesto de
trabajo" + Readed_val_01 + "Horas MO" + Capacity_MO_for_this_period);
                                //MessageBox.Show(name + "-> " + "
D: " + cant);

                                double Percent = 0;
                                if (Capacity_MO_for_this_period !=
0) { Percent = (C * 100) / Capacity_MO_for_this_period; }


Capacity_table.Rows[Operation_row][Col] = C.ToString();

Capacity_table_use_percent.Rows[Operation_row][Col] = C.ToString() + "
(" + Percent + "%)";


//MessageBox.Show(Capacity_MO_for_this_period + "-> " + " D: " +
Percent);

                                if (C > Capacity_MO_for_this_period)
                                {
                                    string The_cell =
(Col.ToString() + " (" + Convert.ToString(Operation_row) + ")");

Cells_MO_capacity_out.Add(The_cell);
                                }
                            }

                            if (_i == 2)  //Si es la primera columna
== MQ Maquinaria
                            {

                                int MQ_row =
Detect_row_number_of(Capacity_table, "Requerimientos de Capacidad de
Maquinaria");
                                int Operation_row =
Detect_row_number_of__from_(Capacity_table, Creq.Rows[_h][0].ToString(),
MQ_row); //Obtiene el row de la operacion actual
```

```
//MessageBox.Show(Creq.Rows[_h][0].ToString() + " " +
Operation_row.ToString());

                                    string Readed_val_00 =
Capacity_table.Rows[Operation_row][Col].ToString();
                                    if (Readed_val_00 == "") {
Readed_val_00 = "0"; }
                                    double A =
Convert.ToDouble(Readed_val_00); //LEE LOS REQUERIMIENTOS RE CAPACIDAD
ACUMULADOS
                                    double B =
Convert.ToDouble(Creq.Rows[_h][2].ToString()); //LEE LOS RESULTADOS DE
CALCULOS DE CAPACIDAD
                                    double C = A + B; //SUMA EL
ACUMULADO CON EL CALCULO


Capacity_table.Rows[Operation_row][Col] = C.ToString();

                                    //CALCULA EL PORCENTAJE DE
UTILIZACION
                                    double D = 0;

                                    string Readed_val_01 =
Capacity_table.Rows[Operation_row][0].ToString();  //NOMBRE DE LA
OPERACION O CENTRO DE TRABAJO
                                    if (Readed_val_01 == "") {
Readed_val_01 = "0"; }
                                    //MessageBox.Show(Readed_val_01);

                                    //SELECCIONAR LOS DATOS CORRECTOS
PARA BUSCAR LA CAPACIDAD MAXIMA PARA EL PUESTO DE TRABAJO EN EL PERIODO
                                    ArrayList Columns = new ArrayList();
Columns.Add("Operaciones");
                                    ArrayList Values = new ArrayList();
Values.Add(Readed_val_01);
                                    bool Is_Operation =
F.Exist_row_on_Access_Table(Form1.Current_MRP, "Operaciones", Columns,
Values);

                                    Columns = new ArrayList();
Columns.Add("Almacenamientos");
                                    Values = new ArrayList();
Values.Add(Readed_val_01);
                                    bool Is_Storage =
F.Exist_row_on_Access_Table(Form1.Current_MRP, "Almacenamientos",
Columns, Values); DataTable Worstations_table = new DataTable();
```

```
                                    if (Is_Operation) {
Worstations_table = Operations_table; }
                                    if (Is_Storage) { Worstations_table
= Storages_table; }



                                    //ENCONTRAR LA CAPACIDAD MAXIMA PARA
EL PUESTO DE TRABAJO EN EL PERIODO
                                    double Capacity_MQ_for_this_period =
0;
                                    for (int x = 0; x <
Worstations_table.Rows.Count; x++)
                                    {
                                        string Storage_name =
Worstations_table.Rows[x][1].ToString();
                                        int Period =
Convert.ToInt32(Worstations_table.Rows[x][5].ToString());

                                        if (Col == Period &&
Storage_name == Readed_val_01)
                                        {
                                            Capacity_MQ_for_this_period
= Convert.ToDouble(Worstations_table.Rows[x][8].ToString()); //CAACIDAD
MQ
                                        }
                                    }
                                    //MessageBox.Show("Periodo: " + Col
+ "Puesto de trabajo" + Readed_val_01 + "Horas MO" +
Capacity_MQ_for_this_period);

                                    double Percent = 0;
                                    if (Capacity_MQ_for_this_period !=
0) { Percent = (C * 100) / Capacity_MQ_for_this_period; }


Capacity_table.Rows[Operation_row][Col] = C.ToString();

Capacity_table_use_percent.Rows[Operation_row][Col] = C.ToString() + "
(" + Percent + "%)";

                                    if (C > Capacity_MQ_for_this_period)
                                    {
                                        string The_cell =
(Col.ToString() + " (" + Convert.ToString(Operation_row) + ")");

Cells_MQ_capacity_out.Add(The_cell);
```

```
                                }
                            }
                        }
                    }

                }
            }



            Add_row_to_capacity_table();
            ProgBar.PerformStep();

            for (int times = 0; times < 2; times++)//SUMA
HORIZONTALMENTE Y VERTICALMENTE (LOS TOTALES) LAS FILAS PARA CALCULAR EL
TOTAL DE CADA ITEM (OPERACION Y ALMACENAMIENTOS)
            {
                double Sum_vert = 0;
                int Current_row = 0;
                if (times == 0) { Current_row =
Detect_row_number_of(Capacity_table, "Requerimientos de capacidad de
Mano de Obra"); }
                if (times == 1) { Current_row =
Detect_row_number_of(Capacity_table, "Requerimientos de Capacidad de
Maquinaria"); }

                for (int i = 0; i < Items_list.Count; i++)
                {

                    int Actual_row = 0;
                    if (times == 0) { Actual_row =
Detect_row_number_of(Capacity_table, "Requerimientos de capacidad de
Mano de Obra"); }
                    if (times == 1) { Actual_row =
Detect_row_number_of(Capacity_table, "Requerimientos de Capacidad de
Maquinaria"); }

                    int Start_row = Actual_row;
                    Actual_row = Actual_row + i + 1;


                    double Sum = 0;

                    int Count_01 = Capacity_table.Columns.Count;
                    for (int j = 1; j < Count_01 - 1; j++)
                    {
                        string Value =
Capacity_table.Rows[Actual_row][j].ToString();
```

```
                            if (Value == "") { Value = "0"; }
                            double Batch = Convert.ToDouble(Value);

                            Sum = Sum + Batch;
                        }

                        Sum_vert = Sum_vert + Sum;  //OBTIENE LA SUMA
VERTICAL DE LA SUMAS TOTALES

                        Capacity_table.Rows[Actual_row][Count_01 - 1] =
Sum.ToString();
                        Capacity_table.Rows[Start_row + Items_list.Count
+ 1][Count_01 - 1] = Sum_vert.ToString();
                    }
                }

                ProgBar.PerformStep();

                dataGridView5.DataSource = Capacity_table;  //CARGAR LA
TABLA EN EL DATAGRID
                dataGridView5.AutoResizeColumns();

                ProgBar.PerformStep();
                //MOSTRAR EN ROJO LAS CELDAS DONDE NO SE CUMPLE CON LAS
CAPACIDADES
                Redraw();
            }
            catch (Exception Error)
            {
                F.Show_Error_Message(Error.Message,
"Calculate_capacity");
            }
        }

        private int Detect_row_number_of(DataTable The_table, string
To_detect)
        {
            //READ THE 0 COLUMN OF THE TABLE SEARCHING A MATCH TO
TO_DETECT STRING
            int Actual_row = 0;

            try
            {
                DataTable Table = The_table;

                int Count_00 = Table.Rows.Count;
                for (int i = 0; i < Count_00; i++)
                {
```

```
                    string Value = The_table.Rows[i][0].ToString();

                    if (Value == To_detect)
                    {
                        Actual_row = i;
                    }
                }
            }
            catch (Exception Error)
            {
                F.Show_Error_Message(Error.Message, "Detectando Row");
            }

            return Actual_row;
        }

        private int Detect_row_number_of__from_(DataTable The_table,
string To_detect, int from_this_row)
        {
            //READ THE 0 COLUMN OF THE TABLE SEARCHING A MATCH TO
TO_DETECT STRING
            int Actual_row = 0;

            try
            {
                DataTable Table = The_table;
                int Count_00 = Table.Rows.Count;

                if (from_this_row < Count_00)
                {
                    for (int i = (from_this_row - 1); i < Count_00; i++)
                    {
                        string Value = The_table.Rows[i][0].ToString();

                        if (Value == To_detect)
                        {
                            Actual_row = i;
                            goto exit;
                        }
                    }
                }
            }
            catch (Exception Error)
            {
                F.Show_Error_Message(Error.Message, "Detectando Row");
            }

            exit:
```

```
            return Actual_row;
        }

        private void button5_Click(object sender, EventArgs e)
        {
            try
            {
                if (MRP_Table.Rows.Count < 2)
                {
                    F.Show_Advisory_Message("Debe desarrollar el cuadro
de MRP en la tarjeta anterior", "Calculando capacidades");
                }
                else
                {
                    button6.Enabled = true;

                    progressBar5.Minimum = 1;
                    progressBar5.Maximum = 11;
                    progressBar5.Value = 1;
                    progressBar5.Step = 1;

                    //LIMPIAR LAS TABLAS
                    Capacity_table.Rows.Clear();
                    Capacity_table.Columns.Clear();

                    Capacity_table_use_percent.Rows.Clear();
                    Capacity_table_use_percent.Columns.Clear();

                    Calculate_capacity(Products_table, Operations_table,
Storage_table, progressBar5);
                }
            }
            catch (Exception Error)
            {
                F.Show_Error_Message(Error.Message, "button5_Click");
            }
        }

        private void button6_Click(object sender, EventArgs e)
        {
            try
            {
                if (button6.Text == "% Utilizacion On")
                {
                    button6.Text = "% Utilizacion Off";
```

```
                    dataGridView5.DataSource = Capacity_table;  //CARGAR
LA TABLA EN EL DATAGRID
                    dataGridView5.AutoResizeColumns();
                }
                else if (button6.Text == "% Utilizacion Off")
                {
                    button6.Text = "% Utilizacion On";

                    dataGridView5.DataSource =
Capacity_table_use_percent;  //CARGAR LA TABLA EN EL DATAGRID
                    dataGridView5.AutoResizeColumns();
                }

                // INDICA LAS CELDAS EN LAS QUE SE SOBREPASA LA
CAPACIDAD
                Redraw();
            }
            catch (Exception Error)
            {
                F.Show_Error_Message(Error.Message, "button6_Click");
            }
        }


        private void Redraw()
        {
            //MARCA LOS EXCESOS DE CAPACIDAD DE MANO DE OBRA Y
MAQUINARIA
            try
            {
                for (int i = 0; i < Cells_MO_capacity_out.Count; i++)
//MANO DE OBRA
                {
                    string Cell = Cells_MO_capacity_out[i].ToString();
                    string Name = F.Read_name(Cell);
                    int cant = F.Read_cant(Cell);

                    dataGridView5.Rows[cant].Cells[Name].Style.BackColor
= Color.Red;
                }

                for (int i = 0; i < Cells_MQ_capacity_out.Count; i++)
//MAQUINARIA
                {
                    string Cell = Cells_MQ_capacity_out[i].ToString();
                    string Name = F.Read_name(Cell);
                    int cant = F.Read_cant(Cell);
```

```
                    dataGridView5.Rows[cant].Cells[Name].Style.BackColor
= Color.Red;
                }
            }
            catch (Exception Error)
            {
                F.Show_Error_Message(Error.Message, "Redibujando
Celdas");
            }
        }


        private void Calculate_Workstations_Download(DataTable MP_table,
DataTable Requiered_capacity, ProgressBar ProgBar)
        {
            try
            {
                if (comboBox2.SelectedIndex >= 0)
                {
                    int Number_of_columns = MRP_Table.Columns.Count;
                    string Selected_workstation_name =
comboBox2.SelectedItem.ToString();
                    //MessageBox.Show(Selected_workstation_name);

                    for (int i = 0; i < Number_of_columns; i++)
                    { WorkStations_download.Columns.Add(i.ToString()); }
// CREA LAS COLUMNAS PARA LA TABLA

                    ProgBar.PerformStep();


                    Add_row_to_Workstations_download_table();


                    ProgBar.PerformStep();

                    //LISTA DE MATERIAS PRIMAS
                    if (radioButton1.Checked == true ||
radioButton3.Checked == true)
                    {

                        ArrayList MP_list = new ArrayList();
                        for (int i = 0; i < MP_table.Rows.Count; i++)
                        {
                            string MP_name =
MP_table.Rows[i][1].ToString();    //MP_list.Add(MP_name);

                            ProgBar.PerformStep();
```

```
                            for (int l = 0; l <
Requiered_capacity.Rows.Count; l++)
                            {
                                string Workstation_name_on_table =
Requiered_capacity.Rows[l][2].ToString();
                                string MP_name_on_table =
Requiered_capacity.Rows[l][1].ToString();

                                //MessageBox.Show(MP_name + "-" +
MP_name_on_table + "\r\n " + Selected_workstation_name + "-" +
Workstation_name_on_table);

                                if (Workstation_name_on_table ==
Selected_workstation_name && MP_name_on_table == MP_name)
                                {
                                    string Row =
Requiered_capacity.Rows[l][0].ToString() + "|" +
Requiered_capacity.Rows[l][1].ToString() + "|" +
Requiered_capacity.Rows[l][2].ToString() + "|" +

Requiered_capacity.Rows[l][3].ToString() + "|" +
Requiered_capacity.Rows[l][4].ToString() + "|" +
Requiered_capacity.Rows[l][5].ToString();
                                    //MessageBox.Show(Row);

                                    double A =
Convert.ToDouble(Requiered_capacity.Rows[l][4].ToString());
//Requerimiento de MO X Und de MP
                                    double B =
Convert.ToDouble(Requiered_capacity.Rows[l][5].ToString());
//Requerimiento de MQ X Und de MP
                                    double C = A + B;

                                    if (C > 0) //SI SE REQUIERE ALGO EN
MO O MQ
                                    {

//MessageBox.Show(F.Shorted_file_name.ToString());
                                        for (int j = 0; j <
MRP_Table.Rows.Count - 1; j++)
                                        {
                                            string Value =
MRP_Table.Rows[j][0].ToString();
                                            if (Value == MP_name)
                                            {
                                                int Actual_row =
WorkStations_download.Rows.Count;
```

```
WorkStations_download.Rows[Actual_row - 1][0] = MP_name;  //AGREGA EL
NOMBRE DE LA MP A LA TABLA

Add_row_to_Workstations_download_table();

                                                string label =
MRP_Table.Rows[j + 10][0].ToString(); //COLOCA LOS REQUERIMIENTOS
                                                if (label == "Colocacion
de requerimientos")
                                                {
                                                    for (int k = 0; k <
MRP_Table.Columns.Count; k++)
                                                    {
                                                        int Act_row =
WorkStations_download.Rows.Count;
                                                        string Val =
MRP_Table.Rows[j + 10][k].ToString();

WorkStations_download.Rows[Act_row - 1][k] = Val;
                                                    }

Add_row_to_Workstations_download_table();
                                                }
                                            }

                                        }
                                    }
                                }
                            }
                        }
                    }

                    if (radioButton2.Checked == true ||
radioButton3.Checked == true)
                    {
                        ArrayList Products_list = new ArrayList();
                        for (int i = 0; i < Products_table.Rows.Count;
i++)
                        {
                            string Product_name =
Products_table.Rows[i][1].ToString();    //MP_list.Add(Product_name);

                            ProgBar.PerformStep();
```

```
                            for (int l = 0; l <
Requiered_capacity.Rows.Count; l++)
                            {
                                string Workstation_name_on_table =
Requiered_capacity.Rows[l][2].ToString();
                                string Product_name_on_table =
Requiered_capacity.Rows[l][1].ToString();

                                //MessageBox.Show(Product_name + "-" +
Product_name_on_table + "\r\n " + Selected_workstation_name + "-" +
Workstation_name_on_table);

                                if (Workstation_name_on_table ==
Selected_workstation_name && Product_name_on_table == Product_name)
                                {
                                    string Row =
Requiered_capacity.Rows[l][0].ToString() + "|" +
Requiered_capacity.Rows[l][1].ToString() + "|" +
Requiered_capacity.Rows[l][2].ToString() + "|" +

Requiered_capacity.Rows[l][3].ToString() + "|" +
Requiered_capacity.Rows[l][4].ToString() + "|" +
Requiered_capacity.Rows[l][5].ToString();
                                    //MessageBox.Show(Row);

                                    double A =
Convert.ToDouble(Requiered_capacity.Rows[l][4].ToString());
//Requerimiento de MO X Und de MP
                                    double B =
Convert.ToDouble(Requiered_capacity.Rows[l][5].ToString());
//Requerimiento de MQ X Und de MP
                                    double C = A + B;

                                    if (C > 0) //SI SE REQUIERE ALGO EN
MO O MQ
                                    {

//MessageBox.Show(F.Shorted_file_name.ToString());
                                        for (int j = 0; j <
MRP_Table.Rows.Count - 1; j++)
                                        {
                                            string Value =
MRP_Table.Rows[j][0].ToString();
                                            if (Value == Product_name)
                                            {
                                                int Actual_row =
WorkStations_download.Rows.Count;
```

```
WorkStations_download.Rows[Actual_row - 1][0] = Product_name;  //AGREGA
EL NOMBRE DE LA MP A LA TABLA

Add_row_to_Workstations_download_table();

                                                string label =
MRP_Table.Rows[j + 10][0].ToString(); //COLOCA LOS REQUERIMIENTOS
                                                if (label == "Colocacion
de requerimientos")
                                                {
                                                    for (int k = 0; k <
MRP_Table.Columns.Count; k++)
                                                    {
                                                        int Act_row =
WorkStations_download.Rows.Count;
                                                        string Val =
MRP_Table.Rows[j + 10][k].ToString();

WorkStations_download.Rows[Act_row - 1][k] = Val;
                                                    }

Add_row_to_Workstations_download_table();
                                                }
                                            }

                                        }
                                    }
                                }
                            }
                        }
                    }

                    ProgBar.PerformStep();

                    //CARGAR LA TABLA EN EL DATAGRIDVIEW
                    dataGridView6.DataSource = WorkStations_download;
                    dataGridView6.AutoResizeColumns();
                }
            }
            catch (Exception Error)
            {
                F.Show_Error_Message(Error.Message,
"Calculate_Workstations_Download");
            }
        }
```

```
        private void Calc_download()
        {
            try
            {
                if (MRP_Table.Rows.Count < 2)
                {
                    F.Show_Advisory_Message("Debe desarrollar el cuadro
de MRP en la tarjeta anterior", "Descarga a puestos de trabajo");
                }
                else
                {
                    progressBar6.Minimum = 1;
                    progressBar6.Maximum = 3 + MP_table.Rows.Count;
                    progressBar6.Value = 1;
                    progressBar6.Step = 1;

                    //LIMPIAR LA TABLA
                    WorkStations_download.Rows.Clear();
                    WorkStations_download.Columns.Clear();

                    Calculate_Workstations_Download(MP_table,
Requiered_capacity, progressBar6);
                }
            }
            catch (Exception Error)
            {
                F.Show_Error_Message(Error.Message, "Calc_download");
            }
        }

        private void comboBox2_SelectedIndexChanged(object sender,
EventArgs e)
        {
            Calc_download();
        }

        private void toolStripMenuItem1_Click(object sender, EventArgs
e)
        {
            try
            {
                DataGridView DatGrdVw = new DataGridView();

                //DataTable Table =  new DataTable();
                switch (Clicked_datagridview)
                {
                    case 1: DatGrdVw = dataGridView1; break;
```

```
                case 2: DatGrdVw = dataGridView2; break;
                case 3: DatGrdVw = dataGridView3; break;
                case 4: DatGrdVw = dataGridView5; break;
                case 5: DatGrdVw = dataGridView6; break;
            }

            if
(DatGrdVw.GetCellCount(DataGridViewElementStates.Selected) > 0)
            {

Clipboard.SetDataObject(DatGrdVw.GetClipboardContent());
            }
        }
        catch (Exception Error)
        {
            F.Show_Error_Message(Error.Message,
"toolStripMenuItem1_Click");
        }
    }

    private void dataGridView1_MouseHover(object sender, EventArgs
e)
    {
        try
        {
            Clicked_datagridview = 1;
        }
        catch (Exception Error)
        {
            F.Show_Error_Message(Error.Message,
"dataGridView1_MouseHover");
        }
    }

    private void dataGridView2_MouseHover(object sender, EventArgs
e)
    {
        try
        {
            Clicked_datagridview = 2;
        }
        catch (Exception Error)
        {
            F.Show_Error_Message(Error.Message,
"dataGridView2_MouseHover");
        }
    }
```

```
        private void dataGridView3_MouseHover(object sender, EventArgs
e)
        {
            try
            {
                Clicked_datagridview = 3;
            }
            catch (Exception Error)
            {
                F.Show_Error_Message(Error.Message,
"dataGridView3_MouseHover");
            }
        }

        private void dataGridView5_MouseHover(object sender, EventArgs
e)
        {
            try
            {
                Clicked_datagridview = 4;
            }
            catch (Exception Error)
            {
                F.Show_Error_Message(Error.Message,
"dataGridView4_MouseHover");
            }
        }

        private void dataGridView6_MouseHover(object sender, EventArgs
e)
        {
            try
            {
                Clicked_datagridview = 5;
            }
            catch (Exception Error)
            {
                F.Show_Error_Message(Error.Message,
"dataGridView5_MouseHover");
            }
        }

        private void toolStripMenuItem3_Click(object sender, EventArgs
e)
        {
            try
            {
                DataTable Table = new DataTable();
```

```
                switch (Clicked_datagridview)
                {
                    case 1: Table = dataGridView1.DataSource as
DataTable; break;
                    case 2: Table = dataGridView2.DataSource as
DataTable; break;
                    case 3: Table = dataGridView3.DataSource as
DataTable; break;
                    case 4: Table = dataGridView5.DataSource as
DataTable; break;
                    case 5: Table = dataGridView6.DataSource as
DataTable; break;
                }

                F.Export_table_Excel(Table);
            }
            catch (Exception Error)
            {
                F.Show_Error_Message(Error.Message,
"toolStripMenuItem3_Click");
            }
        }

        private void Graficar(int Type)
        {
            try
            {
                DataTable Table = new DataTable();
                switch (Clicked_datagridview)
                {
                    case 1: Table = dataGridView1.DataSource as
DataTable; break;
                    case 2: Table = dataGridView2.DataSource as
DataTable; break;
                    case 3: Table = dataGridView3.DataSource as
DataTable; break;
                    case 4: Table = dataGridView5.DataSource as
DataTable; break;
                    case 5: Table = dataGridView6.DataSource as
DataTable; break;
                }

                int Selected_row = Selected_row_index;
                int Number_of_columns = Table.Columns.Count;

                int[] Data_array = new int[Number_of_columns];

                for (int i = 1; i < Number_of_columns; i++)
```

```
                {
                    string Value =
Table.Rows[Selected_row][i].ToString();
                    if (Value == "") { Value = "0"; }
                    //Read_Cell(2, i, Selected_row);
                    int Numeric_value = Convert.ToInt16(Value);
                    Data_array.SetValue(Numeric_value, i);
                }

                string Val = Table.Rows[Selected_row][0].ToString();
                if (Val == "") { Val = "0"; }
                string Row_name = Val;

                if (Type < 0)
                {
                    Type = 2;
                }

                F.Draw_chart(Number_of_columns, Data_array, Row_name,
Type);
            }
            catch (Exception Error)
            {
                F.Show_Advisory_Message(Error.Message, "Graficar");
            }
        }

        private void areaToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            Graficar(0);
        }

        private void lineaToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            Graficar(1);
        }

        private void columnasToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            Graficar(2);
        }

        private void dataGridView3_Click(object sender, EventArgs e)
        {
            try
```

```
            {
                int Count_selected_cells =
dataGridView3.GetCellCount(DataGridViewElementStates.Selected);

                if (Count_selected_cells > 0)
                {
                    for (int i = 0; i < Count_selected_cells; i++)
                    {
                        Selected_row_index =
dataGridView3.SelectedCells[i].RowIndex;
                    }
                }
                else
                {
                    F.Show_Advisory_Message("Debe seleccionar una fila
para graficar", "Graficando Fila");
                }
            }
            catch (Exception Error)
            {
                F.Show_Error_Message(Error.Message,
"dataGridView3_Click");
            }
        }

        private void dataGridView1_Click(object sender, EventArgs e)
        {
            try
            {
                int Count_selected_cells =
dataGridView1.GetCellCount(DataGridViewElementStates.Selected);

                if (Count_selected_cells > 0)
                {
                    for (int i = 0; i < Count_selected_cells; i++)
                    {
                        Selected_row_index =
dataGridView1.SelectedCells[i].RowIndex;
                    }
                }
                else
                {
                    F.Show_Advisory_Message("Debe seleccionar una fila
para graficar", "Graficando Fila");
                }
            }
            catch (Exception Error)
            {
```

```
                F.Show_Error_Message(Error.Message,
"dataGridView1_Click");
            }
        }

        private void dataGridView2_Click(object sender, EventArgs e)
        {
            try
            {
                int Count_selected_cells =
dataGridView2.GetCellCount(DataGridViewElementStates.Selected);

                if (Count_selected_cells > 0)
                {
                    for (int i = 0; i < Count_selected_cells; i++)
                    {
                        Selected_row_index =
dataGridView2.SelectedCells[i].RowIndex;
                    }
                }
                else
                {
                    F.Show_Advisory_Message("Debe seleccionar una fila
para graficar", "Graficando Fila");
                }
            }
            catch (Exception Error)
            {
                F.Show_Error_Message(Error.Message,
"dataGridView2_Click");
            }
        }

        private void dataGridView5_Click(object sender, EventArgs e)
        {
            try
            {
                int Count_selected_cells =
dataGridView5.GetCellCount(DataGridViewElementStates.Selected);

                if (Count_selected_cells > 0)
                {
                    for (int i = 0; i < Count_selected_cells; i++)
                    {
                        Selected_row_index =
dataGridView5.SelectedCells[i].RowIndex;
                    }
                }
```

```
                else
                {
                    F.Show_Advisory_Message("Debe seleccionar una fila
para graficar", "Graficando Fila");
                }
            }
            catch (Exception Error)
            {
                F.Show_Error_Message(Error.Message,
"dataGridView5_Click");
            }
        }

        private void dataGridView6_Click(object sender, EventArgs e)
        {
            try
            {
                int Count_selected_cells =
dataGridView6.GetCellCount(DataGridViewElementStates.Selected);

                if (Count_selected_cells > 0)
                {
                    for (int i = 0; i < Count_selected_cells; i++)
                    {
                        Selected_row_index =
dataGridView6.SelectedCells[i].RowIndex;
                    }
                }
                else
                {
                    F.Show_Advisory_Message("Debe seleccionar una fila
para graficar", "Graficando Fila");
                }
            }
            catch (Exception Error)
            {
                F.Show_Error_Message(Error.Message,
"dataGridView6_Click");
            }
        }

        private void comboBox3_SelectedIndexChanged(object sender,
EventArgs e)
        {
            try
            {
                textBox5.Text = comboBox3.SelectedItem.ToString();
```

```
                if (comboBox3.SelectedIndex == 0) { textBox6.Text =
"350"; }
                if (comboBox3.SelectedIndex == 1) { textBox6.Text =
"50"; }
                if (comboBox3.SelectedIndex == 2) { textBox6.Text =
"12"; }
                if (comboBox3.SelectedIndex == 3) { textBox6.Text = "4";
}
                if (comboBox3.SelectedIndex == 4) { textBox6.Text = "";
}
            }
            catch (Exception Error)
            {
                F.Show_Error_Message(Error.Message,
"comboBox3_SelectedIndexChanged");
            }
        }

        private void toolStripMenuItem2_Click(object sender, EventArgs
e)
        {
            try
            {
                DataGridView DatGrdVw = new DataGridView();
                switch (Clicked_datagridview)
                {
                    case 1: DatGrdVw = dataGridView1; break;
                    case 2: DatGrdVw = dataGridView2; break;
                    case 3: DatGrdVw = dataGridView3; break;
                    case 4: DatGrdVw = dataGridView5; break;
                    case 5: DatGrdVw = dataGridView6; break;
                }


                SaveFileDialog Save = new SaveFileDialog();

                Save.Filter = "*.xml|*.xml|*.*|*.*";
                if (Save.ShowDialog() == DialogResult.OK)
                {
                    DataTable Table = DatGrdVw.DataSource as DataTable;
                    DataSet ds = new DataSet();
                    ds.Tables.Add(Table);
                    ds.WriteXml(Save.FileName);
                }
            }
            catch (Exception Error)
            {
```

```
                F.Show_Error_Message(Error.Message,
"toolStripMenuItem2_Click");
            }
        }

        private void radioButton1_CheckedChanged(object sender,
EventArgs e)
        {
            try
            {
                if (radioButton1.Checked == true)
                {
                    Calc_download();
                }
            }
            catch (Exception Error)
            {
                F.Show_Error_Message(Error.Message,
"radioButton1_CheckedChanged");
            }
        }

        private void radioButton2_CheckedChanged(object sender,
EventArgs e)
        {
            try
            {
                if (radioButton2.Checked == true)
                {
                    Calc_download();
                }
            }
            catch (Exception Error)
            {
                F.Show_Error_Message(Error.Message,
"radioButton2_CheckedChanged");
            }
        }

        private void radioButton3_CheckedChanged(object sender,
EventArgs e)
        {
            try
            {
                if (radioButton3.Checked == true)
                {
                    Calc_download();
                }
```

```
            }
            catch (Exception Error)
            {
                F.Show_Error_Message(Error.Message,
"radioButton3_CheckedChanged");
            }
        }

        private void Select_batch_policy_MP()
        {
            try
            {
                if (comboBox4.SelectedIndex >= 0)
                {
                    MP_table =
F.Get_table_from_Access_File(Form1.Current_MRP, "MP");
                    for (int i = 0; i < MP_table.Rows.Count; i++)  //LEE
LA TABLA DE OPERACIONES
                    {
                        string Operation_name =
MP_table.Rows[i][1].ToString();
                        if (Operation_name ==
comboBox4.SelectedItem.ToString())
                        {
                            int Index_of_Batch_policy =
Convert.ToInt32(MP_table.Rows[i][11].ToString());
                            comboBox6.SelectedIndex =
Index_of_Batch_policy;
                            Indx_to_change_batch_policy = i;
                        }
                    }
                }
            }
            catch (Exception Error)
            {
                F.Show_Error_Message(Error.Message,
"radioButton3_CheckedChanged");
            }
        }

        private void Select_batch_policy_Product()
        {
            try
            {
                if (comboBox5.SelectedIndex >= 0)
                {
                    Products_table =
F.Get_table_from_Access_File(Form1.Current_MRP, "Productos");
```

```
                    for (int i = 0; i < Products_table.Rows.Count; i++)
//LEE LA TABLA DE OPERACIONES
                    {
                        string Operation_name =
Products_table.Rows[i][1].ToString();
                        if (Operation_name ==
comboBox5.SelectedItem.ToString())
                        {
                            int Index_of_Batch_policy =
Convert.ToInt32(Products_table.Rows[i][12].ToString());
                            comboBox6.SelectedIndex =
Index_of_Batch_policy;
                            Indx_to_change_batch_policy = i;
                        }
                    }
                }
            }
            catch (Exception Error)
            {
                F.Show_Error_Message(Error.Message,
"Select_batch_policy_Product");
            }
        }

        private void comboBox4_SelectedIndexChanged(object sender,
EventArgs e)
        {
            Select_batch_policy_MP();
        }

        private void comboBox5_SelectedIndexChanged(object sender,
EventArgs e)
        {
            Select_batch_policy_Product();
        }

        private void tabControl3_SelectedIndexChanged(object sender,
EventArgs e)
        {
            try
            {
                if (tabControl3.SelectedIndex == 0)
                {
                    Select_batch_policy_MP();
                }

                if (tabControl3.SelectedIndex == 1)
                {
```

```
                    Select_batch_policy_Product();
                }
            }
            catch (Exception Error)
            {
                F.Show_Error_Message(Error.Message,
"tabControl3_SelectedIndexChanged");
            }
        }
    }
}
```

E-7. VENTANA ADMINISTRADOR DE IMAGENES

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.IO;
using Functions;
using MRProgram_00.Properties;

namespace MRProgram_00
{
    public partial class Form2 : Form
    {
        Functions.Operatives F = new Operatives();

        public bool extended = false;  // SI ESTE FORM NO SE INVOCA
DESDE UNA VENTANA PARA LA CREACION DE MP,PT,OPERACIONES,
ALMACENAMIENTOS, ETC.
                                       // ESTE ES = FALSE Y NO MUESTRA
EL BOTON DE SELECCIONAR IMAGEN, SI ES TRUE, MUESTRA EL BOTON DE
SELECCIONAR IMAGEN.

        //MANEJADOR DE EVENTOS PARA ENVIAR A OTRO FORM LA INFORMACION
SOBRE LA IMAGEN SELECCIOANDA
        public delegate void UpdateHandler(object sender,
UpdateEventArgs e);
        public event UpdateHandler IdentityUpdated;

        //VARIABLES PARA EL FORM PARA CREAR UNA NUEVA COLECCION DE
IMAGENES
        Form Create_collection = new Form();
        TextBox Text1 = new TextBox();
        Button Button_01 = new Button();
        Button Button_02 = new Button();


        // VARIABLES PARA LAS IMAGENES
        int Image_Height =
Convert.ToInt32(Settings.Default["Size_of_images_Height"]);
        int Image_Width =
Convert.ToInt32(Settings.Default["Size_of_images_Widht"]);
        Image Image_00;
```

```
        public Form2()
        {
            InitializeComponent();
        }

        public void Fill_image_collections_list()
        {
            //LLENA EL LISTBOX 1 CON LOS NOMBRES DE LAS CARPETAS DE CADA
COLECCION DE IMAGENES EN LA CARPETA DE IMAGENES
            try
            {
                F.Identify_folders();
                F.Get_folders(F.Images_folder);

                listBox1.Items.Clear();
                foreach (string folder in F.Folders_list)
                {
                    listBox1.Items.Add(F.Get_file_name_n_ext(folder));
//FILTRA EL NOMBRE DE LA CARPETA
                }
            }
            catch (Exception Error)
            {
                F.Show_Error_Message(Error.ToString(),
"Fill_image_collections_list");
            }
        }

        private void Create_the_new_collection_form()
        {
            //CREA UN FORM PARA QUE EL USUARIO INTRODUZCA EL NOMBRE DE
LA NUEVA COLECCION DE IMAGENES
            try
            {
                Text1.Size = new Size(170, 20);
                Text1.Top = 10;
                Text1.Left = 10;

                Button_01.Size = new Size(60, 20);
                Button_01.Top = 35;
                Button_01.Left = 60;
                Button_01.Text = "Crear";
                Button_01.Click += new
System.EventHandler(this.Button_01_Click);

                Button_02.Size = new Size(60, 20);
                Button_02.Top = 35;
```

```
                Button_02.Left = 120;
                Button_02.Text = "Cancel";
                Button_02.Click += new
System.EventHandler(this.Button_02_Click);


                Create_collection.Size = new Size(200, 90);
                Create_collection.Text = "Nueva Coleccion";
                Create_collection.FormBorderStyle =
FormBorderStyle.FixedToolWindow;
                Create_collection.StartPosition =
FormStartPosition.CenterParent;

                Create_collection.Controls.Add(Button_01);
                Create_collection.Controls.Add(Button_02);
                Create_collection.Controls.Add(Text1);
                Create_collection.Disposed += new
EventHandler(form_Disposed);
            }
            catch(Exception Error)
            {
                F.Show_Error_Message(Error.ToString(),
"Create_the_new_collection_form");
            }
        }

        private void Form2_Load(object sender, EventArgs e)
        {
            try
            {
                Create_the_new_collection_form();
                Fill_image_collections_list();

                if (extended == false) { this.Size = new Size(800, 425);
}  //AMPLIA O RECORTA EL TAMAÑO DE LA VENTANA PARA MOSTRAR EL BOTON DE
SELECCIONAR IMAGEN
                else { this.Size = new Size(800, 470); }
            }
            catch (Exception Error)
            {
                F.Show_Error_Message(Error.ToString(), "Form2_Load");
            }
        }


        private void Show_selected_collection_images()
        {
```

```
            // LLENA EL LISTVIEW1 CON LAS IMAGENES EN MINIATURA QUE
ESTAN EN LA COLECCION SELECCIONADA EN EL LISTBOX1
            try
            {
                ArrayList Extensions = new ArrayList(); //ARREGLO CON
LAS EXTENSIONES ADMITIDAS POR EL PROGRAMA
                Extensions.Clear();
                Extensions.Add("jpg");
                Extensions.Add("gif");
                Extensions.Add("bmp");
                Extensions.Add("png");

                ImageList List_images = new ImageList();
                List_images.Images.Clear();
                List_images.ImageSize = new Size(Image_Height,
Image_Width);

                this.listView1.Items.Clear();
                this.listView1.SmallImageList = List_images;
                this.listView1.LargeImageList = List_images;


                string path = F.Images_folder + "\\" +
listBox1.SelectedItem;
                ArrayList Images_files = F.List_files_of_ext(path,
Extensions);

                foreach (string file in Images_files)
                {
                    string pic = string.Format(@file);
                    Image Img = Image.FromFile(pic);
                    List_images.Images.Add(Img);
                    Img.Dispose();     //LIBERA LOS RECURSOS DEL
ARCHIVO, PARA QUE SEA POSIBLE BORRARLA SI EL USUARIO OPRIME EL BOTON DE
BORRAR
                }

                if (Images_files.Count > 0)
                {
                    progressBar1.Minimum = 1;
                    progressBar1.Maximum = Images_files.Count;
                    progressBar1.Value = 1;
                    progressBar1.Step = 1;
                }

                for (int i = 0; i < Images_files.Count; i++)
                {
```

```
                    string file =
F.Get_file_name_n_ext(Images_files[i].ToString());
                    this.listView1.Items.Add(file.ToString());
                    this.listView1.Items[i].ImageIndex = i;

                    progressBar1.PerformStep();
                }
            }
            catch(Exception Error)
            {
                F.Show_Error_Message(Error.ToString(),
"Show_selected_collection_images");
            }
        }

        private void listBox1_SelectedIndexChanged(object sender,
EventArgs e)
        {
            Show_selected_collection_images();
        }

        private void Change_image_size_and_reload(int amount)
        {
            // CAMBIA EL TAMAÑO DE LAS IMAGENES EN EL LISTVIEW1
            try
            {
                Image_Height = Image_Height + amount;
                Image_Width = Image_Width + amount;

                if (Image_Height > 256 || Image_Width > 128)
                {
                    Image_Height = 256;
                    Image_Width = 256;
                }

                if (Image_Height < 16 || Image_Width < 16)
                {
                    Image_Height = 16;
                    Image_Width = 16;
                }

                Settings.Default["Size_of_images_Height"] =
Image_Height;  //GUARDA LOS DATOS DEL ULTIMO TAMAÑO DE IMAGEN ELEGIDO
POR EL USUARIO
                Settings.Default["Size_of_images_Widht"] = Image_Width;
                Settings.Default.Save();

                Show_selected_collection_images();
```

```
            }
            catch (Exception Error)
            {
                F.Show_Error_Message(Error.ToString(),
"Change_image_size_and_reload");
            }
        }

        private void Button_01_Click(object sender, EventArgs e)
        {
            //MUESTRA UN NUEVO FORM, PARA INTRODUCIR EL NOMBRE PARA LA
NUEVA COLECCION DE IMAGENES
            // Y CREA UNA CARPETA CON EL NOMBRE DE LA NUEVA COLECCION EN
LA CARPETA DEFAULT PARA LAS IMAGENES ..\IMAGES
            try
            {
                if (Text1.Text != "")
                {
                    if (F.Check_existent_folder(F.Images_folder,
Text1.Text) == true)// REVISA SI YA EXISTE UNA COLECCION DE IMAGENES CON
ESE NOMBRE
                    {
                        DialogResult Result = MessageBox.Show("Ya existe
una coleccion con este nombre.  ¿Desea reemplazarla?."
                                              + "\r\n"
                                              + "\r\n" + "Si hace click
en 'Si' es posible que se pierdan las imagenes en esa coleccion.",
                        "Advertencia", MessageBoxButtons.YesNo,
MessageBoxIcon.Warning);

                        if (Result == DialogResult.Yes)
                        {
                            F.Create_folder(F.Images_folder,
Text1.Text);  //LA CARPETA PARA LA COLECCION DE IMAGENES YA EXISTE, SI
HACE CLIC EN 'YES' REEMPLAZA LA CARPETA EXISTENTE
                        }
                    }
                    else
                    {
                        F.Create_folder(F.Images_folder, Text1.Text);
//CREA LA CARPETA QUE NO EXISTE PARA LA COLECCION DE IMAGENES
                    }

                    Create_collection.Close();
                    Fill_image_collections_list();
                }
                else
                {
```

```
                    MessageBox.Show("Debe escribir un nombre para la
nueva coleccion", "Advertencia", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
                }
            }
            catch (Exception Error)
            {
                F.Show_Error_Message(Error.ToString(), "Creando nueva
coleccion de imagenes");
            }
        }

        private void Button_02_Click(object sender, EventArgs e)
        {
            try
            {
                Create_collection.Hide();
            }
            catch (Exception Error)
            {
                F.Show_Error_Message(Error.ToString(), "Closing Create
Collection");
            }
        }

        void form_Disposed(object sender, EventArgs e)
        {
            try
            {
                Create_collection = null;
            }
            catch (Exception Error)
            {
                F.Show_Error_Message(Error.ToString(), "form_Disposed");
            }
        }

        private void listView1_SelectedIndexChanged(object sender,
EventArgs e)
        {
            //MUESTRA EL PREVIEW EN EL PICTUREBOX1 DE LA IMAGEN
SELECCIONADA EL EL LISTVIEW1
            try
            {
                int count=  listView1.SelectedItems.Count;

                if (count == 1) //SOLO MUESTRA EL PREVIEW DE LA PRIMERA
IMAGEN SELECCIONADA Y NO DE LOS DEMAS ITEMS SELECCIONADOS
```

```
                {
                    string path_of_image = F.Images_folder + "\\" +
listBox1.SelectedItem + "\\" + listView1.SelectedItems[0].Text;
                    Image_00 = Image.FromFile(path_of_image);
                    pictureBox1.Image = Image_00;
                }


            }
            catch (Exception Error)
            {
                F.Show_Error_Message(Error.Message, "Administrador de
imagenes -> Mostrando el preview de la imagen seleccionada");
            }
        }

        private void button5_Click(object sender, EventArgs e)
        {
            //MUESTRA EL PREVIEW EN EL PICTUREBOX1 DE LA IMAGEN
SELECCIONADA EL EL LISTVIEW1
            try
            {
                if (listBox1.SelectedIndex < 0)
                {
                    F.Show_Advisory_Message("Debe seleccionar una
coleccion de imagenes", "Advertencia");
                }
                else
                {
                    int count = listView1.SelectedItems.Count;
                    if(count == 0) { F.Show_Advisory_Message("Debe
seleccionar una imagen", "Advertencia"); }
                    else
                    {
                        if (count == 1) //SOLO MUESTRA EL PREVIEW DE LA
PRIMERA IMAGEN SELECCIONADA Y NO DE LOS DEMAS ITEMS SELECCIONADOS
                        {
                            string path_of_image = F.Images_folder +
"\\" + listBox1.SelectedItem + "\\" + listView1.SelectedItems[0].Text;
                            UpdateEventArgs args = new
UpdateEventArgs(path_of_image);
                            IdentityUpdated(this, args);

                            this.Dispose();
                        }
                    }
                }
            }
```

```
            catch (Exception Error)
            {
                F.Show_Error_Message(Error.Message, "Administrador de 
imagenes -> Mostrando el preview de la imagen seleccionada");
            }
        }

        private void toolStripButton1_Click(object sender, EventArgs e)
        {
            Create_collection.ShowDialog();
        }

        private void toolStripButton2_Click(object sender, EventArgs e)
        {
            //ELIMINA LA COLECCION DE IMAGENES SELECCIONADA EN EL 
LISTBOX1
            try
            {
                if (listBox1.SelectedIndex < 0)
                {
                    MessageBox.Show("Debe seleccionar una coleccion de 
la lista", "Advertencia", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                }
                else
                {
                    DialogResult result = MessageBox.Show("¿Esta seguro 
de querer eliminar esta coleccion de la lista?"
                                                    + "\r\n" + "Si hace 
clic en 'Si' se perderan todas las imagenes en la coleccion.",
                                                    "Advertencia", 
MessageBoxButtons.YesNo, MessageBoxIcon.Warning);
                    if (result == DialogResult.Yes)
                    {
                        if (pictureBox1.Image != null) { 
Image_00.Dispose(); }
                        listView1.Items.Clear();
                        F.Get_files(F.Images_folder + "\\" + 
listBox1.SelectedItem);
                        foreach (string path in F.Files_list)
                        {
                            File.Delete(path);
                        }
                        F.Delete_folder(F.Images_folder, 
listBox1.SelectedItem.ToString());
                        Fill_image_collections_list();
                    }
                }
            }
```

```
            catch (Exception Error)
            {
                F.Show_Error_Message(Error.ToString(), "Eliminando una
coleccion de imagenes");
            }
        }

        private void toolStripButton3_Click(object sender, EventArgs e)
        {
            //AGREGA UNA IMAGEN A LA COLECCION DE IMAGENES SELECCIONADA
EN EL LISTBOX1 Y LA MUESTRA EN EL LISTVIEW1
            try
            {
                if (listBox1.SelectedIndex > 0) //REVISA SI HAY UNA
COLECCION DE IMAGENES SELECCIONADA
                {
                    OpenFileDialog open = new OpenFileDialog();
                    open.Multiselect = true;
                    open.Filter = "JPG files (*.jpg)|*.jpg|GIF files
(*.gif)|*.gif|PNG files (*.png)|*.png|BMP files (*.bmp)|*.bmp|All files
(*.*)|*.*";

                    if (open.ShowDialog() == DialogResult.OK)
                    {
                        progressBar1.Minimum = 1;
                        progressBar1.Maximum = open.FileNames.Length;
                        progressBar1.Value = 1;
                        progressBar1.Step = 1;

                        foreach (string file in open.FileNames)
                        {
                            if (F.Check_existent_file(F.Images_folder +
"\\" + listBox1.SelectedItem, F.Get_file_name(file),
F.Get_file_extension(file)) == true)
                            {
                                //SE REVISA SI YA EXISTE EL ARCHIVO EN
LA COLECCION Y MUESTRA UN MENSAJE SI EXISTE
                                if (MessageBox.Show("Ya existe un
archivo con el nombre: " + F.Get_file_name_n_ext(file) + " en la
coleccion."
                                                     + "\r\n" + "¿Desea
remplazar el archivo existente?", "Advertencia",

MessageBoxButtons.YesNo, MessageBoxIcon.Warning) == DialogResult.Yes)
                                {
                                    File.Copy(file, F.Images_folder +
"\\" + listBox1.SelectedItem + "\\" + F.Get_file_name_n_ext(file),
true); //SI HACE CLIC EN SI SE REEMPLAZA EL ARCHIVO
```

```
                                }
                            }
                            else
                            {
                                File.Copy(file, F.Images_folder + "\\" +
listBox1.SelectedItem + "\\" + F.Get_file_name_n_ext(file)); //COPIA EL
ARCHIVO SELECCIONADO A LA COLECCION SELECCIONADA
                            }

                            progressBar1.PerformStep();
                        }

                        Show_selected_collection_images(); //REFRESCA LA
LISTA DE IMAGENES DE LA COLECCION SELECCIONADA
                    }
                }
                else
                {
                    F.Show_Advisory_Message("Debe seleccionar una
coleccion de imagenes.", "Administrador de Imagenes");
                }
            }
            catch (Exception Error)
            {
                F.Show_Error_Message(Error.ToString(), "Administrador de
imagenes -> Agregar imagen a una coleccion");
            }
        }

        private void toolStripButton4_Click(object sender, EventArgs e)
        {
            //ELIMINA LA IMAGEN SELECCIONADA EN EL LISTVIEW1 DE LA
COLECCION DE IMAGENES SELECCIONADA EN EL LISTBOX1
            try
            {
                if (listBox1.SelectedIndex > 0)
                {
                    int count = listView1.SelectedItems.Count;

                    if (count > 0)
                    {
                        if (MessageBox.Show("La o las imagenes
seleccionadas seran borradas de forma permanente de la colección de
imagenes" + "\r\n" +
                            "¿Desea Continuar?", "Advertencia",
MessageBoxButtons.YesNo, MessageBoxIcon.Warning) == DialogResult.Yes)
                        {
                            for (int i = 0; i < count; i++)
```

```
                            {
                                string path_of_image = F.Images_folder +
"\\" + listBox1.SelectedItem + "\\" + listView1.SelectedItems[i].Text;
                                Image_00.Dispose();

                                File.Delete(path_of_image);
                            }

                            Show_selected_collection_images();
//REFRESCA LA LISTA DE IMAGENES DE LA COLECCION SELECCIONADA
                        }
                    }
                    else
                    {
                        F.Show_Advisory_Message("Debe seleccionar al
menos una imagen", "Advertencia");
                    }
                }
                else
                {
                    F.Show_Advisory_Message("Debe seleccionar una
coleccion de imagenes", "Advertencia");
                }
            }
            catch (Exception Error)
            {
                F.Show_Error_Message(Error.Message, "Administrador de
imagenes -> Borrando imagen");
            }
        }

        private void toolStripButton5_Click(object sender, EventArgs e)
        {
            Change_image_size_and_reload(8);
        }

        private void toolStripButton6_Click(object sender, EventArgs e)
        {
            Change_image_size_and_reload(-8);
        }
    }

    public class UpdateEventArgs : System.EventArgs
    {
        public string Image_path;

        public UpdateEventArgs(string Path_of_image)
        {
```

```
            this.Image_path = Path_of_image;
        }
    }
}
```

E-8. VENTANA ADMINISTRADOR DE UNIDADES

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using Functions;

namespace MRProgram_00
{
    public partial class Form3 : Form
    {
        Functions.Operatives F = new Operatives();

        public bool extended = false;  // SI ESTE FORM NO SE INVOCA
DESDE UNA VENTANA PARA LA CREACION DE MP,PT,OPERACIONES,
ALMACENAMIENTOS, ETC.
        // ESTE ES = FALSE Y NO MUESTRA EL BOTON DE SELECCIONAR IMAGEN,
SI ES TRUE, MUESTRA EL BOTON DE SELECCIONAR IMAGEN.

        public string Initial_value = "";
        public string Selected_unit = "";

        //MANEJADOR DE EVENTOS PARA ENVIAR A OTRO FORM LA INFORMACION
SOBRE LA IMAGEN SELECCIOANDA
        public delegate void UpdateHandler2(object sender,
UpdateEventArgs2 e);
        public event UpdateHandler2 IdentityUpdated2;

        //VARIABLES PARA EL FORM PARA CREAR UNA NUEVA UNIDAD
        Form Create_collection = new Form();
        TextBox Text1 = new TextBox();
        Button Button_01 = new Button();
        Button Button_02 = new Button();

        private void Create_the_new_collection_form()
        {
            //CREA UN FORM PARA QUE EL USUARIO INTRODUZCA EL NOMBRE DE
LA NUEVA UNIDAD
            try
            {
                Text1.Size = new Size(170, 20);
```

```
                Text1.Top = 10;
                Text1.Left = 10;

                Button_01.Size = new Size(60, 20);
                Button_01.Top = 35;
                Button_01.Left = 60;
                Button_01.Text = "Crear";
                Button_01.Click += new
System.EventHandler(this.Button_01_Click);

                Button_02.Size = new Size(60, 20);
                Button_02.Top = 35;
                Button_02.Left = 120;
                Button_02.Text = "Cancel";
                Button_02.Click += new
System.EventHandler(this.Button_02_Click);


                Create_collection.Size = new Size(200, 90);
                Create_collection.Text = "Nueva Unidad";
                Create_collection.FormBorderStyle =
FormBorderStyle.FixedToolWindow;
                Create_collection.StartPosition =
FormStartPosition.CenterParent;

                Create_collection.Controls.Add(Button_01);
                Create_collection.Controls.Add(Button_02);
                Create_collection.Controls.Add(Text1);
                Create_collection.Disposed += new
EventHandler(form_Disposed);
            }
            catch (Exception Error)
            {
                F.Show_Error_Message(Error.ToString(),
"Create_the_new_collection_form");
            }
        }

        public Form3()
        {
            F.Identify_folders();
            InitializeComponent();
        }

        private void Fill_units_list()
        {
            //LLENA EL LISTBOX1 Y EL COMBOBOX1 CON LA LISTA DE LAS
UNIDADES EN EL PROGRAMA
```

```
            try
            {
                DataTable Table =
F.Get_table_from_Access_File(Form1.Current_MRP, "Unidades");
                ArrayList Categorias = new ArrayList();
                ArrayList Unidades = new ArrayList();

                for (int i = 0; i < Table.Rows.Count; i++)
                {
                    string Categoria = Table.Rows[i][1].ToString();
                    string Unidad = Table.Rows[i][2].ToString();

                    if (!Categorias.Contains(Categoria))
                    {
                        Categorias.Add(Categoria);  //Obtiene la lista
de las diferentes categorias para las Unidades
                    }

                    if (!Unidades.Contains(Unidad))
                    {
                        Unidades.Add(Unidad);  //Obtiene la lista de las
diferentes categorias para las Unidades
                    }
                }

                listBox2.Items.Clear();
                foreach (string Categ in Categorias)
                {
                    listBox2.Items.Add(Categ);  //LLENA EL LISTBOX2 CON
LA LISTA DE CATEGORIAS PARA LAS UNIDADES
                }

                comboBox1.Items.Clear();
                foreach (string Unit in Unidades)
                {
                    comboBox1.Items.Add(Unit); //LLENA EL COMBOBOX1 CON
LA LISTA DE TODAS LAS UNIDADES UNIDADES
                }
            }
            catch (Exception Error)
            {
                F.Show_Error_Message(Error.ToString(),
"Fill_units_list");
            }
        }

        private void Form3_Load(object sender, EventArgs e)
        {
```

```
            try
            {
                Create_the_new_collection_form();
                Fill_units_list();

                //FILL THE DATA IF THE FOR WAS INVOCKED FROM ANOTHER
FORM
                textBox1.Text = Initial_value;
                if (Selected_unit != "") { int indx =
comboBox1.Items.IndexOf(Selected_unit); comboBox1.SelectedIndex = indx;
}


                if (extended == false) { this.Size = new Size(772, 160);
button6.Visible = true; button7.Visible = true; }//AMPLIA O RECORTA EL
TAMAÑO DE LA VENTANA
                else { this.Size = new Size(772, 391);
groupBox1.Location = new Point(12, 97); }

                //HIDE MISSING DATA INDICATORS
                pictureBox1.Hide();
                pictureBox2.Hide();
                pictureBox3.Hide();
            }
            catch (Exception Error)
            {
                F.Show_Error_Message(Error.ToString(), "Form3_Load");
            }
        }

        private void Button_01_Click(object sender, EventArgs e)
        {
            // CREA UN ARCHIVO PARA LA NUEVA UNIDAD
            try
            {
                ArrayList Columns = new ArrayList();
                ArrayList Values = new ArrayList();

                string Table_name = "Unidades";

                Columns.Clear();
                Columns.Add("Index");
                Columns.Add("Categoria");
                Columns.Add("De_A");
                Columns.Add("Unidad");
                Columns.Add("Factor_de_conversion");
                Columns.Add("Simbolo");
```

```
                Values.Clear();

Values.Add(F.Get_right_index_to_add_Data_on_Access_Table(Form1.Current_M
RP, Table_name));
                Values.Add("Personalizada");
                Values.Add(Text1.Text);
                Values.Add(Text1.Text);
                Values.Add(1);
                Values.Add(Text1.Text);

                //IF DOESNT EXIST ADD IT
                F.AddDataToAccessTable(Form1.Current_MRP, Table_name,
Columns, Values);

                listBox1.Items.Clear();
                Fill_units_list();
            }
            catch (Exception Error)
            {
                F.Show_Error_Message(Error.ToString(), "Creando una
nueva unidad");
            }
        }

        private void Button_02_Click(object sender, EventArgs e)
        {
            // CIERRA EL FORM PARA CREAR UNA NUEVA UNIDAD
            try
            {
                Create_collection.Hide();
            }
            catch (Exception Error)
            {
                F.Show_Error_Message(Error.ToString(), "Creando una
nueva unidad -> Boton cancelar");
            }
        }

        void form_Disposed(object sender, EventArgs e)
        {
            // CIERRA EL FORM PARA CREAR UNA NUEVA UNIDAD
            try
            {
                Create_collection = null;
            }
            catch (Exception Error)
            {
```

```
                F.Show_Error_Message(Error.ToString(), "Creando una
nueva unidad -> Boton cerrar");
            }
        }

        private void listBox1_SelectedIndexChanged(object sender,
EventArgs e)
        {
            try
            {
                DataTable Table =
F.Get_table_from_Access_File(Form1.Current_MRP, "Unidades");
                //ArrayList Unidades = new ArrayList();
                string Selected_unidad =
listBox1.SelectedItem.ToString();

                DataTable Unit_Table = new DataTable(); //CREA LA TABLA
Y LAS COLUMNAS PARA LA UNIDAD SELECCIONADA
                Unit_Table.Columns.Add("UNIDAD");
                Unit_Table.Columns.Add("FACTOR DE CONVERSION");
                Unit_Table.Columns.Add("SIMBOLO");

                for (int i = 0; i < Table.Rows.Count; i++)
                {
                    string Unidad = Table.Rows[i][2].ToString();
                    DataRow Row = Unit_Table.NewRow();

                    if (Selected_unidad == Unidad)
                    {
                        Row["UNIDAD"] = Table.Rows[i][3].ToString();
                        Row["FACTOR DE CONVERSION"] =
Table.Rows[i][4].ToString();
                        Row["SIMBOLO"] = Table.Rows[i][5].ToString();

                        Unit_Table.Rows.Add(Row);
                        //Unidades.Add(Unidad);  //Obtiene la lista de
las diferentes unidades
                    }
                }

                dataGridView1.DataSource = Unit_Table;
            }
            catch (Exception Error)
            {
                F.Show_Error_Message(Error.ToString(),
"listBox1_SelectedIndexChanged");
            }
        }
```

```
        private void textBox1_KeyPress(object sender, KeyPressEventArgs
e)
        {
            // SOLO PERMITE INTRODUCIR Y BORRAR DATOS NUMERICOS EN EL
TEXTBOX1
            try
            {
                if
(F.this_char_is_a_number_point_or_backspace(e.KeyChar, textBox1.Text) ==
true)
                {
                    e.Handled = true;
                }
            }
            catch (Exception Error)
            {
                F.Show_Error_Message(Error.ToString(),"Creando una nueva
unidad -> Filtrando teclas en el teclado para TextBox1");
            }
        }

        private void textBox2_KeyPress(object sender, KeyPressEventArgs
e)
        {
            // SOLO PERMITE INTRODUCIR Y BORRAR DATOS NUMERICOS EN EL
TEXTBOX2
            try
            {
                ArrayList chars = new ArrayList();

                chars.Add('1'); chars.Add('2'); chars.Add('3');
chars.Add('4'); chars.Add('5');
                chars.Add('6'); chars.Add('7'); chars.Add('8');
chars.Add('9'); chars.Add('0'); chars.Add('\b');

                if (!chars.Contains(e.KeyChar))
                {
                    e.Handled = true;
                }
            }
            catch (Exception Error)
            {
                F.Show_Error_Message(Error.ToString(), "Creando una
nueva unidad -> Filtrando teclas en el teclado para TextBox2");
            }
        }
```

```
        private void Calculate_conversion()
        {
            // CALCULA LA CONVERSION DE UNIDADES
            try
            {
                //SHOW MISSING DATA INDICATORS
                if (textBox1.Text == "") { pictureBox1.Show(); }
                if (comboBox1.SelectedIndex < 0) { pictureBox2.Show(); }
                if (comboBox2.SelectedIndex < 0) { pictureBox3.Show(); }

                //HIDE MISSING DATA INDICATORS
                if (textBox1.Text != "") { pictureBox1.Hide(); }
                if (comboBox1.SelectedIndex > -1) { pictureBox2.Hide();
}
                if (comboBox2.SelectedIndex > -1) { pictureBox3.Hide();
}

                if (textBox1.Text == "" || comboBox1.SelectedIndex < 0
|| comboBox2.SelectedIndex < 0)
                {
                    F.Show_Advisory_Message("Debe llenar los datos en
los campos indicados", "Advertencia");
                }
                else
                {
                    DataTable Table =
F.Get_table_from_Access_File(Form1.Current_MRP, "Unidades");
                    DataTable table = new DataTable(); //CREA LA TABLA Y
LAS COLUMNAS PARA LA UNIDAD SELECCIONADA
                    table.Columns.Add("UNIDAD");
                    table.Columns.Add("FACTOR DE CONVERSION");
                    table.Columns.Add("SIMBOLO");

                    string Selected_unidad =
comboBox1.SelectedItem.ToString();

                    for (int i = 0; i < Table.Rows.Count; i++)
                    {
                        string Unidad = Table.Rows[i][2].ToString();
                        DataRow Row = table.NewRow();

                        if (Selected_unidad == Unidad)
                        {
                            Row["UNIDAD"] = Table.Rows[i][3].ToString();
                            Row["FACTOR DE CONVERSION"] =
Table.Rows[i][4].ToString();
                            Row["SIMBOLO"] =
Table.Rows[i][5].ToString();
```

```
                            table.Rows.Add(Row);
                            //Unidades.Add(Unidad);  //Obtiene la lista
de las diferentes unidades
                        }
                    }

                    double quantity = Convert.ToDouble(textBox1.Text);
                    double conversion_factor =
Convert.ToDouble(table.Rows[comboBox2.SelectedIndex].ItemArray[1]);
                    double result = quantity * conversion_factor;

                    textBox2.Text = result.ToString();
                }
            }
            catch (Exception Error)
            {
                F.Show_Error_Message(Error.ToString(), "Creando una
nueva unidad -> Converting Units");
            }
        }

        private void comboBox1_SelectedIndexChanged(object sender,
EventArgs e)
        {
            // MUESTRA EN EL COMBOBOX2 A QUE UNIDADES SE PUEDE CONVERTIR
LA UNIDAD SELECCIONADA EN EL COMBOBOX1
            try
            {
                //DataTable table = F.Read_table_file(F.Units_folder,
comboBox1.SelectedItem.ToString(), "unt");
                DataTable Table =
F.Get_table_from_Access_File(Form1.Current_MRP, "Unidades");
                DataTable table = new DataTable(); //CREA LA TABLA Y LAS
COLUMNAS PARA LA UNIDAD SELECCIONADA
                table.Columns.Add("UNIDAD");
                table.Columns.Add("FACTOR DE CONVERSION");
                table.Columns.Add("SIMBOLO");

                string Selected_unidad =
comboBox1.SelectedItem.ToString();

                for (int i = 0; i < Table.Rows.Count; i++)
                {
                    string Unidad = Table.Rows[i][2].ToString();
                    DataRow Row = table.NewRow();

                    if (Selected_unidad == Unidad)
```

```
                    {
                        Row["UNIDAD"] = Table.Rows[i][3].ToString();
                        Row["FACTOR DE CONVERSION"] = 
Table.Rows[i][4].ToString();
                        Row["SIMBOLO"] = Table.Rows[i][5].ToString();

                        table.Rows.Add(Row);
                        //Unidades.Add(Unidad);  //Obtiene la lista de 
las diferentes unidades
                    }
                }

                int count = table.Rows.Count;
                comboBox2.Items.Clear();

                for (int i = 0; i < count; i++)
                {
                    
comboBox2.Items.Add(table.Rows[i].ItemArray[0].ToString());
                }
            }
            catch (Exception Error)
            {
                F.Show_Error_Message(Error.ToString(), "Creando una 
nueva unidad -> Llenando ComboBox2 con los datos disponibles sobre las 
unidades para convertir.");
            }
        }

        private void comboBox2_SelectedIndexChanged(object sender, 
EventArgs e)
        {
            Calculate_conversion();
        }

        private void button6_Click(object sender, EventArgs e)
        {
            try
            {
                UpdateEventArgs2 args2;
                if (textBox2.Text == "" || comboBox2.SelectedIndex < 0)
                {
                    args2 = new UpdateEventArgs2(textBox1.Text, 
comboBox1.SelectedItem.ToString());
                }
                else
                {
```

```
                    args2 = new UpdateEventArgs2(textBox2.Text,
comboBox2.SelectedItem.ToString());
                }
                IdentityUpdated2(this, args2);

                this.Dispose();
            }
            catch (Exception Error)
            {
                F.Show_Error_Message(Error.ToString(), "button6_Click");
            }
        }

        private void button7_Click(object sender, EventArgs e)
        {
            try
            {
                //this.Size = new Size(772, 423);
                F.Identify_folders();

                if (extended == false)
                {
                    extended = true;
                    this.Size = new Size(772, 423);
                    groupBox1.Location = new Point(12, 134);
                    button7.Text = "Cerrar \r\n Editor de Factores de
Conversion";
                    button7.Image = Image.FromFile(F.Icons_folder +
"\\pencil_delete.png");
                }
                else
                {
                    extended = false;
                    this.Size = new Size(772, 160);
                    button7.Text = "Editar Factores de Conversion";
                    button7.Image = Image.FromFile(F.Icons_folder +
"\\pencil.png");
                }
            }
            catch (Exception Error)
            {
                F.Show_Error_Message(Error.ToString(), "button7_Click");
            }
        }

        private void toolStripButton1_Click(object sender, EventArgs e)
        {
            Calculate_conversion();
```

```
        }

        private void toolStripButton2_Click(object sender, EventArgs e)
        {
            // MUESTRA EL FORM PARA CREAR UNA NUEVA UNIDAD
            try
            {
                Create_collection.ShowDialog();
            }
            catch (Exception Error)
            {
                F.Show_Error_Message(Error.ToString(), "Creando una
nueva unidad -> Mostrando form para nombrar la nueva unidad");
            }
        }

        private void toolStripButton3_Click(object sender, EventArgs e)
        {
            // BORRA UNA UNIDAD DE LA COLECCION DE UNIDADES
            try
            {
                if (listBox2.SelectedIndex >= 0)
                {
                    if (listBox1.SelectedIndex >= 0)
                    {
                        if(MessageBox.Show("Se eliminara esta coleccion
de la base de datos de forma permanente." + "\r\n" +
                                           "Esto podria causar el mal
funcionamiento del MRP." + "\r\n" + "¿Desea continuar?", "Advertencia" ,
MessageBoxButtons.YesNo, MessageBoxIcon.Warning) == DialogResult.Yes)

                        {
                            //GET THE INDEXES OF ROWS IN THE TABLE TO
DELETE
                            string Table_name = "Unidades";
                            DataTable Table =
F.Get_table_from_Access_File(Form1.Current_MRP, Table_name);
                            ArrayList Indexes_of_rows_selected = new
ArrayList();

                            for (int i = 0; i < Table.Rows.Count; i++)
                            {
                                string Categoria =
Table.Rows[i][1].ToString();
                                string De_A =
Table.Rows[i][2].ToString();
                                string Unidad =
Table.Rows[i][3].ToString();
```

```
                                string Categoria_seleccionada =
listBox2.SelectedItem.ToString();
                                string De_A_seleccionada =
listBox1.SelectedItem.ToString();

                                if (Categoria_seleccionada == Categoria
&& De_A_seleccionada == De_A)
                                {

                                    int Index_on_table =
Convert.ToInt32(Table.Rows[i][0].ToString());


F.DeleteDataToAccessTable(Form1.Current_MRP, Table_name,
Index_on_table.ToString());

                                }
                            }

                            listBox1.Items.Clear();
                            DataTable Empty = new DataTable();
                            dataGridView1.DataSource = Empty;
                        }
                    }
                    else
                    {
                        F.Show_Advisory_Message("Debe seleccionar una
unidad", "Borrando informacion de unidad");
                    }
                }
                else
                {
                    F.Show_Advisory_Message("Debe seleccionar una
categoria", "Borrando informacion de unidad");
                }

                //F.Delete_file(F.Units_folder,
listBox1.SelectedItem.ToString(), "unt", "Esta seguro de que desea
eliminar la unidad seleccionada.");
                Fill_units_list();
            }
            catch (Exception Error)
            {
                F.Show_Error_Message(Error.ToString(), "Creando una
nueva unidad -> Borrando Unidad");
            }
        }
```

```
        private void toolStripButton4_Click(object sender, EventArgs e)
        {
            // FORMATEA Y GUARDA LOS DATOS EN UN ARCHIVO .UNT EN LA
CARPETA UNITS DEL PROGRAMA
            try
            {
                if (listBox2.SelectedIndex >= 0)
                {
                    if (listBox1.SelectedIndex >= 0)
                    {
                        if (MessageBox.Show("Se cambiara de factores de
conversion de esta unidad en la base de datos de forma permanente." +
"\r\n" +
                                               "Esto podria causar el
mal funcionamiento del MRP y errores en los calculos." + "\r\n" +
"¿Desea continuar?", "Advertencia", MessageBoxButtons.YesNo,
MessageBoxIcon.Warning) == DialogResult.Yes)
                        {

                            var table = dataGridView1.DataSource as
DataTable;
                            int Number_of_colums = table.Columns.Count;
                            int Number_of_rows = table.Rows.Count;


                            ArrayList Columns = new ArrayList();
                            ArrayList Values = new ArrayList();

                            string Table_name = "Unidades";


//MessageBox.Show(Number_of_rows.ToString());

                            for (int i = 0; i < Number_of_rows; i++)
                            {
                                //int indx =
Convert.ToInt32(Indexes_of_rows_selected[i].ToString());
                                bool Exist_on_Demand = false;

                                Columns.Clear();
                                Columns.Add("Categoria");
                                Columns.Add("De_A");
                                Columns.Add("Unidad");

                                Values.Clear();
```

```
Values.Add(listBox2.SelectedItem.ToString());  //Obtiene el numero de
productos en la tabla

Values.Add(listBox1.SelectedItem.ToString());
                                Values.Add(table.Rows[i][0].ToString());

                                //MessageBox.Show("Categoria" +
listBox2.SelectedItem.ToString() + "  De_A " +
listBox1.SelectedItem.ToString() + "  Unidad " +
table.Rows[i][0].ToString());

                                //CHECK IF EXIST A ROW WITH THE SAME
NAME AND PERIOD
                                Exist_on_Demand =
F.Exist_a_row_with_this_info_of_different_columns_on_Access_Table(Form1.
Current_MRP, Table_name, Columns, Values);
                                int Index_on_access_table =
F.Get_on_Access_Table_Index_of(Form1.Current_MRP, Table_name, Columns,
Values);

                                if (Exist_on_Demand == false)
                                {
                                    Columns.Clear();
                                    Columns.Add("Index");
                                    Columns.Add("Categoria");
                                    Columns.Add("De_A");
                                    Columns.Add("Unidad");
                                    Columns.Add("Factor_de_conversion");
                                    Columns.Add("Simbolo");


                                    Values.Clear();

Values.Add(F.Get_right_index_to_add_Data_on_Access_Table(Form1.Current_M
RP, Table_name));

Values.Add(listBox2.SelectedItem.ToString());  //Obtiene el numero de
productos en la tabla

Values.Add(listBox1.SelectedItem.ToString());
                                    Values.Add(table.Rows[i][0]);
                                    Values.Add(table.Rows[i][1]);
                                    Values.Add(table.Rows[i][2]);

                                    //IF DOESNT EXIST ADD IT

F.AddDataToAccessTable(Form1.Current_MRP, Table_name, Columns, Values);
```

```
                                }
                                else
                                {
                                    Columns.Clear();
                                    Columns.Add("Unidad");
                                    Columns.Add("Factor_de_conversion");
                                    Columns.Add("Simbolo");


                                    Values.Clear();
                                    Values.Add(table.Rows[i][0]);
                                    Values.Add(table.Rows[i][1]);
                                    Values.Add(table.Rows[i][2]);

F.UpdateDataToAccessTable(Form1.Current_MRP, Table_name, Columns,
Values, Index_on_access_table.ToString());
                                }
                            }
                        }
                    }
                    else
                    {
                        F.Show_Advisory_Message("Debe seleccionar una
unidad", "Guardando informacion de unidad");
                    }
                }
                else
                {
                    F.Show_Advisory_Message("Debe seleccionar una
categoria", "Guardando informacion de unidad");
                }
            }
            catch (Exception Error)
            {
                F.Show_Error_Message(Error.ToString(), "Creando /
Modificando unidad -> Guardando datos de la unidad");
            }
        }

        private void toolStripButton5_Click(object sender, EventArgs e)
        {
            // EJECUTA LA CALCULADORA DE WINDOWS
            try
            {
                System.Diagnostics.Process.Start("Calc.exe");
            }
            catch (Exception Error)
            {
```

```
                F.Show_Error_Message(Error.ToString(), "Creando una
nueva unidad -> Llenando DataGrid");
            }
        }

        private void listBox2_SelectedIndexChanged(object sender,
EventArgs e)
        {
            try
            {
                DataTable Table =
F.Get_table_from_Access_File(Form1.Current_MRP, "Unidades");
                ArrayList Unidades = new ArrayList();
                string Selected_categoria =
listBox2.SelectedItem.ToString();

                for (int i = 0; i < Table.Rows.Count; i++)
                {
                    string Unidad = Table.Rows[i][2].ToString();
                    string Categoria = Table.Rows[i][1].ToString();

                    if (!Unidades.Contains(Unidad) && Selected_categoria
== Categoria)
                    {
                        Unidades.Add(Unidad);  //Obtiene la lista de las
diferentes unidades
                    }
                }

                listBox1.Items.Clear();
                foreach (string Unit in Unidades)
                {
                    listBox1.Items.Add(Unit);  //LLENA EL LISTBOX2 CON
LA LISTA DE UNIDADES
                }
            }
            catch (Exception Error)
            {
                F.Show_Error_Message(Error.ToString(),
"listBox2_SelectedIndexChanged");
            }
        }

        private void Delete_rows()
        {
            try
            {
                if (listBox2.SelectedIndex >= 0)
```

```
                {
                    if (listBox1.SelectedIndex >= 0)
                    {
                        //GET THE INDEXES OF ROWS IN THE DATAGRID TO
DELETE
                        int Count_selected_cells =
dataGridView1.GetCellCount(DataGridViewElementStates.Selected);
                        ArrayList Indexes_of_rows_selected = new
ArrayList();

                        if (Count_selected_cells > 0)
                        {
                            for (int i = 0; i < Count_selected_cells;
i++)
                            {
                                if
(!Indexes_of_rows_selected.Contains(dataGridView1.SelectedCells[i].RowIn
dex))
                                {

Indexes_of_rows_selected.Add(dataGridView1.SelectedCells[i].RowIndex);

//MessageBox.Show(dataGridView1.SelectedCells[i].RowIndex.ToString());
                                }
                            }
                        }

                        //FILL THE DATA
                        var table = dataGridView1.DataSource as
DataTable;
                        ArrayList Columns = new ArrayList();
                        ArrayList Values = new ArrayList();
                        string Table_name = "Unidades";


                        for (int i = 0; i <
Indexes_of_rows_selected.Count; i++)
                        {
                            int indx =
Convert.ToInt32(Indexes_of_rows_selected[i].ToString());

                            Columns.Clear();
                            Columns.Add("Categoria");
                            Columns.Add("De_A");
                            Columns.Add("Unidad");

                            Values.Clear();
```

```
Values.Add(listBox2.SelectedItem.ToString());  //Obtiene el numero de 
productos en la tabla

Values.Add(listBox1.SelectedItem.ToString());
                            Values.Add(table.Rows[indx][0]);

                            int Index_on_access_table = 
F.Get_on_Access_Table_Index_of(Form1.Current_MRP, Table_name, Columns, 
Values);


//MessageBox.Show(Index_on_access_table.ToString());
                            if 
(F.DeleteDataToAccessTable(Form1.Current_MRP, Table_name, Columns, 
Values, Index_on_access_table.ToString()) > 0)
                            {
                                dataGridView1.Rows.RemoveAt(indx);
                            }
                        }
                    }
                    else
                    {
                        F.Show_Advisory_Message("Debe seleccionar una 
unidad", "Borrando informacion de unidad");
                    }
                }
                else
                {
                    F.Show_Advisory_Message("Debe seleccionar una 
categoria", "Borrando informacion de unidad");
                }
            }
            catch (Exception Error)
            {
                F.Show_Error_Message(Error.Message, "Borrando Filas");
            }
        }

        private void toolStripButton6_Click(object sender, EventArgs e)
        {
            try
            {
                DialogResult result = MessageBox.Show("Esta seguro de 
que desea borrar las filas seleccionadas.  Borrar esta informacion 
podria inhabilitar la utilizacion de los datos de este MRP." + "\r\n" + 
"¿Desea continuar? ", "Advertencia", MessageBoxButtons.YesNo, 
MessageBoxIcon.Question);
```

```
                if (result == DialogResult.Yes)
                {
                    Delete_rows();
                }

            }
            catch (Exception Error)
            {
                F.Show_Error_Message(Error.Message,
"toolStripButton6_Click");
            }
        }

        private void dataGridView1_KeyDown(object sender, KeyEventArgs
e)
        {
            try
            {
                if (e.KeyCode == Keys.Delete)
                {
                    DialogResult result = MessageBox.Show("Esta seguro
de que desea borrar las filas seleccionadas.  Borrar esta informacion
podria inhabilitar la utilizacion de los datos de este MRP." + "\r\n" +
"¿Desea continuar? ", "Advertencia", MessageBoxButtons.YesNo,
MessageBoxIcon.Question);

                    if (result == DialogResult.Yes)
                    {
                        Delete_rows();
                    }
                }
            }
            catch (Exception Error)
            {
                F.Show_Error_Message(Error.Message,
"dataGridView1_KeyDown");
            }
        }
    }

    public class UpdateEventArgs2 : System.EventArgs
    {
        public string Value;
        public string Unit_to_send;

        public UpdateEventArgs2(string Unit_val, string Unit_selectd)
        {
```

```
            this.Value = Unit_val;
            this.Unit_to_send = Unit_selectd;
        }
    }
}
```

E-9. VENTANA DE CONSULTA A BASES DE DATOS EN FORMATOS MICROSOFT ACCESS (MDB Y ACCDB), SQL (VERSIONES 2005 Y 2008), XML (EXTENSIBLE MARKUP LANGUAGE) Y LIBROS DE MICROSOFT EXCEL COMO BASES DE DATOS EN FORMATOS (XLS Y XLSX)

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.OleDb;
using System.Data.Sql;
using System.Data.SqlClient;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using Functions;
using Microsoft.SqlServer.Server;
using System.Xml;
using System.Xml.XPath;

namespace MRProgram_00
{
    public partial class Form5 : Form
    {
        Functions.Operatives F = new Operatives();

        //OLEDB VARS
        private DataSet OleDB_Data_set;
        private DataTable Ole_Db_Schema_table;
        private OleDbConnection OleDB_conection;
        private OleDbDataAdapter OleDB_DatAdapter;
        public string Access_database_location;

        //SQL VARS
        SqlDataSourceEnumerator Servers;
        System.Data.DataTable Servers_table;
        private SqlConnection SQL_conection;
        private DataSet SQL_Data_set;
```

```
        private DataTable SQL_Schema_table;

        //XML VARS
        XmlDataDocument Xmldata = new XmlDataDocument();
        DataSet Data_XML = new DataSet();

        XPathNavigator Nav;
        XPathDocument DocNav;
        XPathNodeIterator NodeIter;
        String Expression;
        ArrayList XML_Columns;

        public Form5()
        {
            InitializeComponent();
        }

        private void Form5_Load(object sender, EventArgs e)
        {
            try
            {

            //INICIALIZA VARIABLES PARA LAS BASES DE DATOS
            OleDB_Data_set = new DataSet();
            Ole_Db_Schema_table = new DataTable();

            SQL_Data_set = new DataSet();
            SQL_Schema_table = new DataTable();

            F.Identify_folders(); //CHECK THE FOLDERS FOR THE PROGRAM
AND CREATE THE MISSING ONES.


            //INICIALIZA LAS VARIABLES PARA LAS BASES DE DATOS QUE USAN
SQL
            Servers = SqlDataSourceEnumerator.Instance;
            Servers_table = new DataTable();

            //INICIALIZA LAS VARIABLES PARA LAS BASES DE DATOS XML
            XML_Columns = new ArrayList();

            }
            catch (Exception Error)
            {
                F.Show_Advisory_Message(Error.Message, "Inicializando
Variables");
            }
        }
```

```
#region OleDB_region

        private void button1_Click(object sender, EventArgs e)
        {
            try
            {
                // OBTENER LA LISTA DE TABLAS EN LA BASE DE DATOS Y
COLOCARLA EN EL LISTBOX1
                openFileDialog1.Filter = "ACCDB Files|*.accdb|MDB
Files|*.mdb";
                DialogResult result = openFileDialog1.ShowDialog();

                if (result != DialogResult.Cancel)
                {
                    Access_database_location = openFileDialog1.FileName;
                    this.textBox1.Text = Access_database_location;

                    F.Find_selected_provider();

                    string String_conection = @"Provider=" +
F.OleDB_selected_provider + ";Data Source=" + Access_database_location;

                    OleDB_conection = new
OleDbConnection(String_conection);

                    dataGridView1.DataSource = null;
                    OleDB_conection.Open();

                    DataTable Schema_table =
OleDB_conection.GetOleDbSchemaTable(OleDbSchemaGuid.Tables, new object[]
{ null, null, null, "TABLE" });
                    listBox1.Items.Clear();

                    foreach (DataRow row in Schema_table.Rows)
                    {
                        listBox1.Items.Add(row[2].ToString());
                    }

                    OleDB_conection.Close();
                }
            }
            catch (Exception Error)
            {
                MessageBox.Show("Cargando base de datos -> " + "\r\n" +
Error.Message);
            }
        }
```

```
        private void button2_Click(object sender, EventArgs e)
        {
            try
            {
                //ABRIR LA VENTANA DE CONFIGURACION PARA LAS BASES DE
DATOS (.MDB Y .ACCDB -> ACCESS, QUE USAN OleDB)
                Form6 F6 = new Form6();

                F6.Show();
            }
            catch (Exception Error)
            {
                MessageBox.Show("button2_Click" + "\r\n" +
Error.Message);
            }
        }

        private void listBox1_SelectedIndexChanged(object sender,
EventArgs e)
        {
            // MOSTRAR EL CONTENIDO DE LA TABLA SELECCIONADA EN EL
DATAGRIDVIEW1
            try
            {
                OleDB_Data_set = new DataSet();
                int index = listBox1.SelectedIndex;
                string table_name = listBox1.Items[index].ToString();
                F.Find_selected_provider();

                OleDB_conection = new OleDbConnection(@"Provider=" +
F.OleDB_selected_provider + ";Data Source=" + Access_database_location);
                OleDB_DatAdapter = new OleDbDataAdapter("Select * FROM "
+ table_name, OleDB_conection);
                OleDB_DatAdapter.Fill(OleDB_Data_set, table_name);
                dataGridView1.DataSource =
OleDB_Data_set.Tables[table_name];

                //OBTENER EL PRIMARY KEY DE LA TABLA SELECCIONADA
                OleDB_conection.Open();
                Ole_Db_Schema_table =
OleDB_conection.GetOleDbSchemaTable(OleDbSchemaGuid.Primary_Keys, new
object[] { null, null, table_name });

                comboBox3.Items.Clear();
                foreach (DataRow row in Ole_Db_Schema_table.Rows) {
listBox2.Items.Add(row[3].ToString());
```

```
comboBox3.Items.Add(row[3].ToString()); label3.Text =
comboBox3.Items.Count.ToString(); }
                if (comboBox3.Items.Count > 0) { comboBox3.SelectedIndex
= 0; }

                checkedListBox1.Items.Clear();
                listBox2.Items.Clear();

                //OBTENER LISTA DE COLUMNAS EN LA TABLA
                DataSet ds = new DataSet();
                OleDbDataAdapter adapter = new OleDbDataAdapter("select
* from " + table_name, OleDB_conection);
                adapter.Fill(ds, table_name);
                DataTable dt = ds.Tables[0];
                foreach (DataColumn dr in dt.Columns) {
checkedListBox1.Items.Add(dr.ToString(), true);
listBox2.Items.Add(dr.ToString()); }

                OleDB_conection.Close();
            }
            catch (Exception Error)
            {
                MessageBox.Show("LLenando DataGrid con la informacion de
la base de datos / primary key -> " + "\r\n" + Error);
            }
        }

        private void button9_Click(object sender, EventArgs e)
        {
            try
            {

            int Index = checkedListBox1.Items.Count;
            for (int i = 0; i < Index; i++) {
checkedListBox1.SetItemChecked(i, true); }

            }
            catch (Exception Error)
            {
                F.Show_Advisory_Message(Error.Message, "Boton 9");
            }
        }

        private void button10_Click(object sender, EventArgs e)
        {
            try
            {
```

```
            int Index = checkedListBox1.Items.Count;
            for (int i = 0; i < Index; i++) {
checkedListBox1.SetItemChecked(i, false); }

            }
            catch (Exception Error)
            {
                F.Show_Advisory_Message(Error.Message, "Boton 10");
            }
        }

        private void button11_Click(object sender, EventArgs e)
        {
            try
            {

            int Index = checkedListBox1.Items.Count;
            for (int i = 0; i < Index; i++)
            {
                if (checkedListBox1.GetItemChecked(i) == true) {
checkedListBox1.SetItemChecked(i, false); }
                else if (checkedListBox1.GetItemChecked(i) == false) {
checkedListBox1.SetItemChecked(i, true); }
            }

            }
            catch (Exception Error)
            {
                F.Show_Advisory_Message(Error.Message, "Boton 11");
            }
        }

        string get_selected_colums()
        {
            string selected = "";

            try
            {
                int count = checkedListBox1.Items.Count;

                for (int i = 0; i < count; i++)
                {
                    if (checkedListBox1.GetItemChecked(i) == true) {
selected = selected + "[" + checkedListBox1.Items[i].ToString() + "]" +
", "; }
                }

                int Last_indx = selected.LastIndexOf(',');
```

```
                selected = selected.Remove(Last_indx);
            }
            catch (Exception Error)
            {
                F.Show_Advisory_Message(Error.Message, "Obteniendo 
columnas seleccionadas");
            }

            return selected;
        }

        private void filter()
        {
            try
            {
                int index = listBox1.SelectedIndex;
                string table_name = listBox1.Items[index].ToString();
                F.Find_selected_provider();

                OleDB_conection = new OleDbConnection(@"Provider=" + 
F.OleDB_selected_provider + ";Data Source=" + Access_database_location);

                // clear the grids data source
                dataGridView1.DataSource = null;

                // create a new DataSet
                DataSet ds = new DataSet();
                OleDbCommand command;
                OleDbDataAdapter adapter;
                command = OleDB_conection.CreateCommand();

                // open the connection
                OleDB_conection.Open();

                // run the query
                command.CommandText = "select " + get_selected_colums() 
+ " from " + table_name;
                if (textBox3.Text != "" && listBox2.SelectedIndex >= 0) 
{ command.CommandText = "select " + get_selected_colums() + " from " + 
table_name + " where [" + listBox2.SelectedItem + "] " + textBox3.Text; 
}

                MessageBox.Show(command.CommandText);

                adapter = new OleDbDataAdapter(command);
                adapter.Fill(ds);

                // close the connection
```

```
                OleDB_conection.Close();

                // set the grid's data source
                dataGridView1.DataSource = ds.Tables[0];
            }
            catch (Exception Error)
            {
                F.Show_Advisory_Message(Error.Message, "filter");
            }
        }

        private void button7_Click(object sender, EventArgs e)
        {
            filter();
        }

        private void button12_Click(object sender, EventArgs e)
        {
            filter();
        }

        private void button13_Click(object sender, EventArgs e)
        {
            try
            {
                int index = listBox1.SelectedIndex;
                string table_name = listBox1.Items[index].ToString();
                F.Find_selected_provider();

                OleDB_conection = new OleDbConnection(@"Provider=" +
F.OleDB_selected_provider + ";Data Source=" + Access_database_location);

                // clear the grids data source
                dataGridView1.DataSource = null;

                // create a new DataSet
                DataSet ds = new DataSet();
                OleDbCommand command;
                OleDbDataAdapter adapter;
                command = OleDB_conection.CreateCommand();

                // open the connection
                OleDB_conection.Open();

                // run the query
                command.CommandText = textBox2.Text;
```

```
                MessageBox.Show(command.CommandText);

                adapter = new OleDbDataAdapter(command);
                adapter.Fill(ds);

                // close the connection
                OleDB_conection.Close();

                // set the grid's data source
                dataGridView1.DataSource = ds.Tables[0];
            }
            catch (Exception Error)
            {
                F.Show_Advisory_Message(Error.Message, "boton 13");
            }
        }
#endregion


#region SQL_region

        private void Get_SQLservers_list()
        {
            //OBTENER LA LISTA DE LOS SERVIDORES SQL Y LLENAR EL
COMBOBOX1
            try
            {
                if (comboBox1.DataSource == null) {
comboBox1.Items.Clear(); } //to avoid clear error


                Servers_table = new DataTable();
                Servers_table = Servers.GetDataSources();
                List<string> Servers_list = new List<string>();

                foreach (DataRow Server in Servers_table.Rows)
                {
                    if
(String.IsNullOrEmpty(Server["InstanceName"].ToString()))
                    {

Servers_list.Add(Server["ServerName"].ToString());
                    }
                    else
                    {
                        Servers_list.Add(Server["ServerName"] + "\\" +
Server["InstanceName"]);
                    }
```

```
                }

                comboBox1.DataSource = Servers_list;
            }
            catch (Exception Error)
            {
                MessageBox.Show("Obteniendo la lista de Servidores SQL -
> " + Error);
            }
        }

        private void tabControl1_SelectedIndexChanged(object sender,
EventArgs e)
        {
            try
            {
                int Index = tabControl1.SelectedIndex;
                if (Index == 1) { Get_SQLservers_list(); }
            }
            catch (Exception Error)
            {
                F.Show_Advisory_Message(Error.Message, "Ejecutando lista
de servidores");
            }
        }

        private void Get_Table_list_of_selected_server(string
Selected_server)
        {
            //OBTIENE LA LISTA DE LAS BASES DE DATOS DEL SERVIDOR QUE SE
INDIQUE EN EL COMBOBOX1 LISTANDOLAS EN EL COMBOBOX2
            try
            {
                List<string> Data_list = new List<string>();
                string String_conection = "Data Source=" +
Selected_server + ";Integrated Security=True;Initial Catalog=master";
                SQL_conection = new SqlConnection(String_conection);

                SQL_conection.Open();

                string Comand_string = "SELECT NAME FROM
sys.databases;";

                SqlCommand comand = new SqlCommand(Comand_string,
SQL_conection);
                System.Data.SqlClient.SqlDataReader DataReader =
comand.ExecuteReader();
```

```
                while (DataReader.Read())
                {
                    Data_list.Add(DataReader[0].ToString());
                }

                comboBox2.DataSource = Data_list;
            }
            catch (Exception Error)
            {
                MessageBox.Show("Obteniendo la lista de Bases de datos
en el servidor SQL seleccionado -> " + Error);
            }
        }

        private void comboBox1_SelectedIndexChanged(object sender,
EventArgs e)
        {
            try
            {
                //OBTIENE LA LISTA DE LAS BASES DE DATOS DEL SERVIDOR
SELECCIONADO EN EL COMBOBOX1
                int Index = tabControl1.SelectedIndex;
                if (Index > 0)
                {

Get_Table_list_of_selected_server(comboBox1.SelectedItem.ToString());
                }
                else
                {
                    MessageBox.Show("Seleccione un servidor SQL");
                }
            }
            catch (Exception Error)
            {
                F.Show_Advisory_Message(Error.Message, "Obteniendo lista
de bases de datos en el servidor");
            }
        }

        private void Get_tables_list_of_selected_database()
        {
            //OBTIENE LA LISTA DE LAS TABLAS DE LA BASES DE DATOS QUE SE
INDIQUE EN EL COMBOBOX2
            try
            {
                string String_conection = "server=" + comboBox1.Text +
"; database=" + comboBox2.Text + "; trusted_connection=true";
                SQL_conection = new SqlConnection(String_conection);
```

```
                SQL_conection.Open();

                SqlCommand comando = new SqlCommand();
                SqlDataAdapter DataAdapter = new SqlDataAdapter();
                DataTable DataTable = new DataTable();

                comando.Connection = SQL_conection;
                comando.CommandType = CommandType.Text;
                comando.CommandText = "Use [" + comboBox2.Text + "] " +
"Select * From INFORMATION_SCHEMA.TABLES " + "Where TABLE_TYPE = 'BASE
TABLE'";

                DataAdapter.SelectCommand = comando;
                DataAdapter.Fill(DataTable);

                SQL_Schema_table = DataTable;
                listBox4.Items.Clear();

                foreach (DataRow row in SQL_Schema_table.Rows)
                {
                    listBox4.Items.Add(row[2].ToString());
                }
            }
            catch (Exception Error)
            {
                MessageBox.Show("Problema en la conexion -> " + Error);
            }

            finally { SQL_conection.Close(); }
        }

        private void comboBox2_SelectedIndexChanged(object sender,
EventArgs e)
        {
            try
            {

            //OBTIENE LA LISTA DE LAS TABLAS DE LA BASES DE DATOS QUE SE
INDIQUE EN EL COMBOBOX2 AL CAMBIAR EL INDEL DEL COMBOBOX2
            if (comboBox2.Text != "" && checkBox1.Checked == true)
            {
                Get_tables_list_of_selected_database();
            }
            else
            {
                listBox4.Items.Clear();
            }
            }
```

```
            catch (Exception Error)
            {
                F.Show_Advisory_Message(Error.Message, "Obteniendo lista
de bases de datos SQL");
            }
        }

        private void button5_Click(object sender, EventArgs e)
        {
            try
            {

            //OBTIENE LA LISTA DE LAS BASES DE DATOS DEL SERVIDOR QUE SE
INDIQUE EN EL COMBOBOX1 LISTANDOLAS EN EL COMBOBOX2
            int Index = tabControl1.SelectedIndex;
            if (Index > 0)
            {
                Get_Table_list_of_selected_server(comboBox1.Text);
            }
            else
            {
                MessageBox.Show("Seleccione o escriba el nombre de un
servidor SQL al que tenga acceso este equipo");
            }

            }
            catch (Exception Error)
            {
                F.Show_Advisory_Message(Error.Message, "BASES DE DATOS
DEL SERVIDOR QUE SE INDIQUE EN EL COMBOBOX1 LISTANDOLAS EN EL
COMBOBOX2");
            }
        }

        private void button3_Click(object sender, EventArgs e)
        {
            try
            {

            //OBTIENE LA LISTA DE LAS TABLAS DE LA BASES DE DATOS QUE SE
INDIQUE EN EL COMBOBOX2
            if (comboBox2.Text != "")
            {
                Get_tables_list_of_selected_database();
            }
            else
            {
                MessageBox.Show("Seleccione una base de datos");
```

```
            }

            }
            catch(Exception Error)
            {
                F.Show_Advisory_Message(Error.Message, "Obteniendo 
listas de bases de datos SQL");
            }
        }

        private void listBox4_SelectedIndexChanged(object sender, 
EventArgs e)
        {
            try
            {
                string table_name = listBox4.SelectedItem.ToString();

                SqlDataAdapter DataAdapter = new SqlDataAdapter();
                string String_conection = "server=" + comboBox1.Text + 
"; database=" + comboBox2.Text + "; trusted_connection=true";
                SQL_conection = new SqlConnection(String_conection);
                SQL_conection.Open();

                SqlCommand comand = new SqlCommand("SELECT * FROM [" + 
table_name + "]", SQL_conection);
                DataAdapter.SelectCommand = comand;
                DataAdapter.Fill(SQL_Data_set, table_name);

                dataGridView3.DataSource = 
SQL_Data_set.Tables[table_name];

                //OBTENER EL PRIMARY KEY DE LA TABLA SELECCIONADA
                SqlCommand comando = new SqlCommand();
                DataTable DataTable = new DataTable();

                comando.Connection = SQL_conection;
                comando.CommandType = CommandType.Text;

                comando.CommandText = "Use [" + comboBox2.Text + "] " + 
"SELECT T.TABLE_NAME,  T.CONSTRAINT_NAME,  K.COLUMN_NAME, 
K.ORDINAL_POSITION  FROM  INFORMATION_SCHEMA.TABLE_CONSTRAINTS T INNER 
JOIN INFORMATION_SCHEMA.KEY_COLUMN_USAGE K ON T.CONSTRAINT_NAME = 
K.CONSTRAINT_NAME  WHERE T.CONSTRAINT_TYPE = 'PRIMARY KEY' -- AND 
T.TABLE_NAME = 'table_name' ORDER BY T.TABLE_NAME, K.ORDINAL_POSITION";

                DataAdapter.SelectCommand = comando;
                DataAdapter.Fill(DataTable);
```

```
                SQL_Schema_table = DataTable;


                checkedListBox2.Items.Clear();

                comboBox4.Items.Clear();
                listBox5.Items.Clear();
                foreach (DataRow row in SQL_Schema_table.Rows) {
comboBox4.Items.Add(row[2].ToString()); label12.Text =
comboBox4.Items.Count.ToString(); }
                if (comboBox4.Items.Count > 0) { comboBox4.SelectedIndex
= 0; }

                //OBTENER LISTA DE COLUMNAS EN LA TABLA
                DataSet ds = new DataSet();
                SqlDataAdapter adapter = new SqlDataAdapter("select *
from " + table_name, SQL_conection);
                adapter.Fill(ds, table_name);
                DataTable dt = ds.Tables[0];
                foreach (DataColumn dr in dt.Columns) {
checkedListBox2.Items.Add(dr.ToString(), true);
listBox5.Items.Add(dr.ToString()); }

                SQL_conection.Close();
            }
            catch (Exception Error)
            {
                F.Show_Advisory_Message(Error.Message, "Mostrando base
de datos SQL");
            }
        }

        private void button17_Click(object sender, EventArgs e)
        {
            try
            {


            int Index = checkedListBox2.Items.Count;
            for (int i = 0; i < Index; i++) {
checkedListBox2.SetItemChecked(i, true); }

            }
            catch (Exception Error)
            {
                F.Show_Advisory_Message(Error.Message, "boton 17");
            }
        }
```

```
        private void button16_Click(object sender, EventArgs e)
        {
            try
            {


            int Index = checkedListBox2.Items.Count;
            for (int i = 0; i < Index; i++) {
checkedListBox2.SetItemChecked(i, false); }

            }
            catch (Exception Error)
            {
                F.Show_Advisory_Message(Error.Message, "boton 16");
            }
        }

        private void button15_Click(object sender, EventArgs e)
        {
            try
            {


            int Index = checkedListBox2.Items.Count;
            for (int i = 0; i < Index; i++)
            {
                if (checkedListBox2.GetItemChecked(i) == true) {
checkedListBox2.SetItemChecked(i, false); }
                else if (checkedListBox2.GetItemChecked(i) == false) {
checkedListBox2.SetItemChecked(i, true); }
            }

            }
            catch (Exception Error)
            {
                F.Show_Advisory_Message(Error.Message, "boton 15");
            }
        }

        string get_selected_colums2()
        {
            string selected = "";
            try
            {
                int count = checkedListBox2.Items.Count;

                for (int i = 0; i < count; i++)
```

```
                {
                    if (checkedListBox2.GetItemChecked(i) == true) {
selected = selected + "[" + checkedListBox2.Items[i].ToString() + "]" +
", "; }
                }

                int Last_indx = selected.LastIndexOf(',');
                selected = selected.Remove(Last_indx);
            }
            catch (Exception Error)
            {
                F.Show_Advisory_Message(Error.Message, "obtener columnas
2");
            }

            return selected;
        }

        private void Filter2()
        {
            try
            {

            string table_name = listBox4.SelectedItem.ToString();
            SqlDataAdapter DataAdapter = new SqlDataAdapter();
            string String_conection = "server=" + comboBox1.Text + ";
database=" + comboBox2.Text + "; trusted_connection=true";
            SQL_conection = new SqlConnection(String_conection);

            // clear the grids data source
            dataGridView1.DataSource = null;

            // create a new DataSet
            DataSet ds = new DataSet();
            SqlCommand command;
            SqlDataAdapter adapter;
            command = SQL_conection.CreateCommand();

            // open the connection
            SQL_conection.Open();

            // run the query
            //command.CommandText = "select " + get_selected_colums() +
" from customers where [customer id] = 1";
            command.CommandText = "select " + get_selected_colums2() + "
from " + table_name;
            if (textBox4.Text != "" && listBox5.SelectedIndex >= 0) {
command.CommandText = "select " + get_selected_colums2() + " from " +
```

```
table_name + " where [" + listBox5.SelectedItem + "] " + textBox4.Text;
}

            MessageBox.Show(command.CommandText);

            adapter = new SqlDataAdapter(command);
            adapter.Fill(ds);

            // close the connection
            SQL_conection.Close();

            // set the grid's data source
            dataGridView3.DataSource = ds.Tables[0];
            }
            catch (Exception Error)
            {
                F.Show_Advisory_Message(Error.Message, "Filtro 2");
            }
        }

        private void button18_Click(object sender, EventArgs e)
        {
            Filter2();
        }

        private void button19_Click(object sender, EventArgs e)
        {
            Filter2();
        }

        private void button20_Click(object sender, EventArgs e)
        {
            try
            {

            string table_name = listBox4.SelectedItem.ToString();
            SqlDataAdapter DataAdapter = new SqlDataAdapter();
            string String_conection = "server=" + comboBox1.Text + ";
database=" + comboBox2.Text + "; trusted_connection=true";
            SQL_conection = new SqlConnection(String_conection);

            // clear the grids data source
            dataGridView1.DataSource = null;

            // create a new DataSet
            DataSet ds = new DataSet();
            SqlCommand command;
            SqlDataAdapter adapter;
```

```
            command = SQL_conection.CreateCommand();

            // open the connection
            SQL_conection.Open();

            // run the query
            command.CommandText = textBox5.Text;

            MessageBox.Show(command.CommandText);

            adapter = new SqlDataAdapter(command);
            adapter.Fill(ds);

            // close the connection
            SQL_conection.Close();

            // set the grid's data source
            dataGridView3.DataSource = ds.Tables[0];

            }
            catch (Exception Error)
            {
                F.Show_Advisory_Message(Error.Message, "boton 20");
            }
        }

        private void button21_Click(object sender, EventArgs e)
        {
            try
            {

            string table_name = listBox4.SelectedItem.ToString();
            SqlDataAdapter DataAdapter = new SqlDataAdapter();
            string String_conection = "server=" + comboBox1.Text + ";
database=" + comboBox2.Text + "; trusted_connection=true";
            SQL_conection = new SqlConnection(String_conection);

            string insertStatement = "INSERT INTO MSreplication_options
" + "([optname], [value], [major_version]) " + "VALUES (?, ?, ?)";

            SqlCommand insertCommand = new SqlCommand(insertStatement,
SQL_conection);

            insertCommand.Parameters.Add("[optname]",
SqlDbType.Char).Value = "l";
            insertCommand.Parameters.Add("[value]", SqlDbType.Int).Value
= "2";
```

```
            insertCommand.Parameters.Add("[major_version]",
SqlDbType.Int).Value = "80";

            SQL_conection.Open();

            int count = insertCommand.ExecuteNonQuery();
            }
            catch (Exception Error)
            {
                F.Show_Advisory_Message(Error.Message, "boton 21");
            }
        }

#endregion


#region XML_region

        private void button23_Click(object sender, EventArgs e)
        {
            Data_XML = new DataSet();
            dataGridView4.DataSource = null;
            listBox6.Items.Clear();
            openFileDialog2.Filter = "Archivos XML|*.xml";

            try
            {
                if (openFileDialog2.ShowDialog() == DialogResult.OK)
                {
                    Data_XML.ReadXml(openFileDialog2.FileName);
                    dataGridView4.DataSource = Data_XML.Tables[0];
                    textBox8.Text = openFileDialog2.FileName;


                    int count = Data_XML.Tables.Count;

                    for (int i = 0; i < count; i++)
                    {

listBox6.Items.Add(Data_XML.Tables[i].TableName.ToString());

                        int K = Data_XML.Tables[i].Columns.Count;

                        checkedListBox3.Items.Clear();
                        for (int j = 0; j < K; j++)
                        {

//listBox3.Items.Add(Data_XML.Tables[i].Columns[j].ColumnName);
```

```
checkedListBox3.Items.Add(Data_XML.Tables[i].Columns[j].ColumnName,
true);
                        }
                    }
                }
            }
            catch (Exception Error)
            {
                MessageBox.Show(Error.Message);
            }
        }

        private void AddNode(XmlNode inXmlNode, TreeNode inTreeNode)
        {
            try
            {

            XmlNode xNode;
            TreeNode tNode;
            XmlNodeList nodeList;
            int i;

            // Loop through the XML nodes until the leaf is reached.
            // Add the nodes to the TreeView during the looping process.
            if (inXmlNode.HasChildNodes)
            {
                nodeList = inXmlNode.ChildNodes;
                for (i = 0; i < nodeList.Count; i++)
                {
                    xNode = inXmlNode.ChildNodes[i];
                    inTreeNode.Nodes.Add(new TreeNode(xNode.Name));
                    tNode = inTreeNode.Nodes[i];
                    AddNode(xNode, tNode);
                }
            }
            else
            {
                // Here you need to pull the data from the XmlNode based
on the
                // type of node, whether attribute values are required,
and so forth.
                inTreeNode.Text = (inXmlNode.OuterXml).Trim();
            }
            }
            catch (Exception Error)
            {
                F.Show_Advisory_Message(Error.Message, "boton 15");
```

```
            }
        }

        private void button25_Click(object sender, EventArgs e)
        {
            try
            {

            OpenFileDialog open = new OpenFileDialog();

            if (open.ShowDialog() == DialogResult.OK)
            {
                // SECTION 1. Create a DOM Document and load the XML
data into it.
                XmlDocument dom = new XmlDocument();
                dom.Load(open.FileName);

                // SECTION 2. Initialize the TreeView control.
                treeView1.Nodes.Clear();
                treeView1.Nodes.Add(new
TreeNode(dom.DocumentElement.Name));
                TreeNode tNode = new TreeNode();
                tNode = treeView1.Nodes[0];

                // SECTION 3. Populate the TreeView with the DOM nodes.
                AddNode(dom.DocumentElement, tNode);
                treeView1.ExpandAll();

                // SECTION 4. Create a new TreeView Node with only the
child nodes.
                XmlNodeList nodelist = dom.SelectNodes("//child");
                XmlDocument cDom = new XmlDocument();
                cDom.LoadXml("<children></children>");
                foreach (XmlNode node in nodelist)
                {
                    XmlNode newElem =
cDom.CreateNode(XmlNodeType.Element, node.Name, node.LocalName);
                    newElem.InnerText = node.InnerText;
                    cDom.DocumentElement.AppendChild(newElem);
                }

                treeView1.Nodes.Add(new
TreeNode(cDom.DocumentElement.Name));
                tNode = treeView1.Nodes[1];
                AddNode(cDom.DocumentElement, tNode);
            }
            }
            catch (Exception Error)
```

```
            {
                F.Show_Advisory_Message(Error.Message, "Cargando XML");
            }
        }

        private void listBox6_SelectedIndexChanged(object sender,
EventArgs e)
        {
            try
            {

            //OpenFileDialog open = new OpenFileDialog();

            //if (open.ShowDialog() != DialogResult.Cancel)
            //{
                //string File_path = open.FileName;
                string File_path = textBox8.Text;

                XmlDataDocument xmlDatadoc = new XmlDataDocument();
                xmlDatadoc.DataSet.ReadXml(File_path);
                DataSet ds = new DataSet();
                ds = xmlDatadoc.DataSet;
                dataGridView4.DataSource = ds.DefaultViewManager;
                //dataGridView4.DataMember = "book";
                dataGridView4.DataMember =
listBox6.SelectedItem.ToString();


                //GET THE COLUMS
                XmlTextReader Reader = new XmlTextReader(File_path);
                XmlNodeType NodeType = new XmlNodeType();

                Reader.Close();


//MessageBox.Show(xmlDatadoc.FirstChild.Name.ToString());

//MessageBox.Show(xmlDatadoc.ChildNodes.Count.ToString());

//MessageBox.Show(xmlDatadoc.ChildNodes[0].Name.ToString());
            //}
            }
            catch (Exception Error)
            {
                F.Show_Advisory_Message(Error.Message, "Seleccionando de
la lista de tablas XML");
            }
```

```
        }

        private void button28_Click(object sender, EventArgs e)
        {
            try
            {

            int Index = checkedListBox3.Items.Count;
            for (int i = 0; i < Index; i++) {
checkedListBox3.SetItemChecked(i, true); }
            }
            catch (Exception Error)
            {
                F.Show_Advisory_Message(Error.Message, "boton 28");
            }
        }

        private void button27_Click(object sender, EventArgs e)
        {
            try
            {

            int Index = checkedListBox3.Items.Count;
            for (int i = 0; i < Index; i++) {
checkedListBox3.SetItemChecked(i, false); }
            }
            catch (Exception Error)
            {
                F.Show_Advisory_Message(Error.Message, "boton 27");
            }
        }

        private void button26_Click(object sender, EventArgs e)
        {
            try
            {

            int Index = checkedListBox3.Items.Count;
            for (int i = 0; i < Index; i++)
            {
                if (checkedListBox3.GetItemChecked(i) == true) {
checkedListBox3.SetItemChecked(i, false); }
                else if (checkedListBox3.GetItemChecked(i) == false) {
checkedListBox3.SetItemChecked(i, true); }
            }
            }
            catch (Exception Error)
            {
```

```
                F.Show_Advisory_Message(Error.Message, "boton 26");
            }
        }

        void get_selected_colums3()
        {
            try
            {

            XML_Columns.Clear();
            int count = checkedListBox3.Items.Count;

            for (int i = 0; i < count; i++)
            {
                if (checkedListBox3.GetItemChecked(i) == true)
                {

XML_Columns.Add(checkedListBox3.Items[i].ToString());

//MessageBox.Show(checkedListBox3.Items[i].ToString());
                }
            }
            }
            catch (Exception Error)
            {
                F.Show_Advisory_Message(Error.Message, "Obtener columnas
3");
            }
        }

        private void button29_Click(object sender, EventArgs e)
        {
            try
            {

            string File_path = textBox8.Text;

            Data_XML = new DataSet();

            Data_XML.ReadXml(File_path);

            DataView DV = new DataView();
            DV.Table = Data_XML.Tables[0];

            dataGridView2.DataSource = DV;
            get_selected_colums3();

            listBox3.Items.Clear();
```

```
            ArrayList Toremove = new ArrayList();


            for (int i = 0; i < DV.Table.Columns.Count; i++)
            {
                if
(XML_Columns.Contains(DV.Table.Columns[i].ColumnName))
                {
                    listBox3.Items.Add(DV.Table.Columns[i].ColumnName);
                }
                else
                {

                    Toremove.Add(DV.Table.Columns[i].ColumnName);

//DV.Table.Columns.Remove(DV.Table.Columns[i].ColumnName);
                }
            }

            for (int i = 0; i < Toremove.Count; i++)
            {
                //MessageBox.Show(Toremove[i].ToString());
                DV.Table.Columns.Remove(Toremove[i].ToString());
            }
            }
            catch (Exception Error)
            {
                F.Show_Advisory_Message(Error.Message, "boton 29");
            }
        }

        private void button30_Click(object sender, EventArgs e)
        {
            try
            {
                string File_path = textBox8.Text;

                Data_XML = new DataSet();

                Data_XML.ReadXml(File_path);

                DataView DV = new DataView();
                DV.Table = Data_XML.Tables[0];

                DV.RowFilter = textBox9.Text;
                dataGridView2.DataSource = DV;
```

```
                get_selected_colums3();

                listBox3.Items.Clear();

                ArrayList Toremove = new ArrayList();

                for (int i = 0; i < DV.Table.Columns.Count; i++)
                {
                    if
(XML_Columns.Contains(DV.Table.Columns[i].ColumnName))
                    {

listBox3.Items.Add(DV.Table.Columns[i].ColumnName);
                    }
                    else
                    {

                        Toremove.Add(DV.Table.Columns[i].ColumnName);

//DV.Table.Columns.Remove(DV.Table.Columns[i].ColumnName);
                    }
                }

                for (int i = 0; i < Toremove.Count; i++)
                {
                    //MessageBox.Show(Toremove[i].ToString());
                    DV.Table.Columns.Remove(Toremove[i].ToString());
                }
            }
            catch (Exception Error)
            {
                MessageBox.Show(Error.Message);
            }
        }

        private void button31_Click(object sender, EventArgs e)
        {
            try
            {

                DocNav = new XPathDocument(textBox8.Text);

                // Create a navigator to query with XPath.
                Nav = DocNav.CreateNavigator();

                Expression = textBox10.Text;

                // Select the node and place the results in an iterator.
```

```
                NodeIter = Nav.Select(Expression);

                listBox3.Items.Clear();

                while (NodeIter.MoveNext())
                {

listBox3.Items.Add(NodeIter.Current.Value.ToString());
                }
            }
            catch (Exception Error)
            {
                F.Show_Advisory_Message(Error.Message, "boton 31");
            }
        }

        private void button24_Click(object sender, EventArgs e)
        {
            try
            {
                SaveFileDialog save = new SaveFileDialog();
                save.Filter = "Archivos XML|*.xml";

                if (save.ShowDialog() == DialogResult.OK)
                {
                    Data_XML.WriteXml(save.FileName,
XmlWriteMode.IgnoreSchema);
                }
            }
            catch (Exception Error)
            {
                F.Show_Advisory_Message(Error.Message, "boton 24");
            }
        }

#endregion


#region EXCEL_region

        private void button34_Click(object sender, EventArgs e)
        {
            try
            {

            OpenFileDialog open = new OpenFileDialog();
            open.Filter = "*.xlsx|*.xlsx|*.xls|*.xls|*.*|*.*";
```

```
            if (open.ShowDialog() != DialogResult.Cancel)
            {
                OleDbConnection Con = new OleDbConnection();
                this.textBox11.Text = open.FileName;


                F.Find_selected_provider();

                Con.ConnectionString = @"Provider = " +
F.OleDB_selected_provider + "; Data Source = " + open.FileName + ";
Extended Properties=Excel 8.0;";
                OleDbCommand Command = new OleDbCommand("SELECT * FROM
[SHEET1$]", Con);


                DataSet DS = new DataSet();
                OleDbDataAdapter Adapter = new
OleDbDataAdapter(Command);
                Adapter.Fill(DS);
                dataGridView5.DataSource = DS.Tables[0];

                Con.Open();

                DataTable esquemaTabla =
Con.GetOleDbSchemaTable(System.Data.OleDb.OleDbSchemaGuid.Tables, new
object[] { null, null, null, "TABLE" });

                string nombreTabla = "";
                listBox7.Items.Clear();

                for (int i = 0; i < esquemaTabla.Rows.Count; i++)
                {
                    nombreTabla =
esquemaTabla.Rows[i]["TABLE_NAME"].ToString();
                    int length = nombreTabla.Length;
                    nombreTabla = nombreTabla.Remove(length - 1);
                    listBox7.Items.Add(nombreTabla);
                }

                Con.Close();
            }

            }
            catch (Exception Error)
            {
                F.Show_Advisory_Message(Error.Message, "boton 34");
            }
        }
```

```
        private void listBox7_SelectedIndexChanged(object sender,
EventArgs e)
        {
            try
            {
                OleDbConnection conn = new OleDbConnection();
                dataGridView5.DataSource = null;
                string path = this.textBox11.Text;
                string table_name = listBox7.SelectedItem.ToString();

                F.Find_selected_provider();
                conn.ConnectionString = @"Provider = " +
F.OleDB_selected_provider + "; Data Source = " + path + "; Extended
Properties=Excel 8.0;";
                OleDbCommand Command = new OleDbCommand("SELECT * FROM
[" + table_name + "$]", conn);


                DataSet DS = new DataSet();
                OleDbDataAdapter Adapter = new
OleDbDataAdapter(Command);
                Adapter.Fill(DS);
                dataGridView5.DataSource = DS.Tables[0];

                int count = DS.Tables[0].Columns.Count;
                checkedListBox4.Items.Clear();
                listBox8.Items.Clear();

                for (int i = 0; i < count; i++)
                {

checkedListBox4.Items.Add(DS.Tables[0].Columns[i].ColumnName, true);

listBox8.Items.Add(DS.Tables[0].Columns[i].ColumnName);
                }
            }
            catch (Exception Error)
            {
                F.Show_Advisory_Message(Error.Message, "Seleccionando
tabla Excel");
            }
        }


#endregion

        private void button35_Click(object sender, EventArgs e)
```

```
        {
            try
            {

            int Index = checkedListBox4.Items.Count;
            for (int i = 0; i < Index; i++) {
checkedListBox4.SetItemChecked(i, true); }
            }
            catch (Exception Error)
            {
                F.Show_Advisory_Message(Error.Message, "boton 35");
            }
        }

        private void button33_Click(object sender, EventArgs e)
        {
            try
            {

            int Index = checkedListBox4.Items.Count;
            for (int i = 0; i < Index; i++) {
checkedListBox4.SetItemChecked(i, false); }
            }
            catch (Exception Error)
            {
                F.Show_Advisory_Message(Error.Message, "boton 33");
            }
        }

        private void button32_Click(object sender, EventArgs e)
        {
            try
            {

            int Index = checkedListBox4.Items.Count;
            for (int i = 0; i < Index; i++)
            {
                if (checkedListBox4.GetItemChecked(i) == true) {
checkedListBox4.SetItemChecked(i, false); }
                else if (checkedListBox4.GetItemChecked(i) == false) {
checkedListBox4.SetItemChecked(i, true); }
            }
            }
            catch (Exception Error)
            {
                F.Show_Advisory_Message(Error.Message, "boton 32");
            }
        }
```

```
        string get_selected_colums4()
        {

            string selected = "";
            try
            {

            int count = checkedListBox4.Items.Count;

            for (int i = 0; i < count; i++)
            {
                if (checkedListBox4.GetItemChecked(i) == true) {
selected = selected + "[" + checkedListBox4.Items[i].ToString() + "]" +
", "; }
            }

            int Last_indx = selected.LastIndexOf(',');
            selected = selected.Remove(Last_indx);
            }
            catch (Exception Error)
            {
                F.Show_Advisory_Message(Error.Message, "Obtener columnas
seleccionada 4");
            }
            return selected;
        }

        private void button36_Click(object sender, EventArgs e)
        {
            try
            {
                OleDbConnection conn = new OleDbConnection();
                dataGridView5.DataSource = null;
                string path = this.textBox11.Text;
                string table_name = listBox7.SelectedItem.ToString();

                F.Find_selected_provider();
                conn.ConnectionString = @"Provider = " +
F.OleDB_selected_provider + "; Data Source = " + path + "; Extended
Properties=Excel 8.0;";
                OleDbCommand Command = new OleDbCommand("SELECT " +
get_selected_colums4() + " FROM [" + table_name + "$]", conn);


                DataSet DS = new DataSet();
                OleDbDataAdapter Adapter = new
OleDbDataAdapter(Command);
```

```
                Adapter.Fill(DS);
                dataGridView5.DataSource = DS.Tables[0];
            }
            catch (Exception Error)
            {
                F.Show_Advisory_Message(Error.Message, "boton 36");
            }

        }

        private void button37_Click(object sender, EventArgs e)
        {
            try
            {
                string path = this.textBox11.Text;
                string table_name = listBox7.SelectedItem.ToString();

                string query = "SELECT " + get_selected_colums4() + "
FROM [" + table_name + "$]" + " WHERE " + "[" +listBox8.SelectedItem +
"]" +  textBox13.Text;
                //MessageBox.Show(query);

                OleDbConnection conn = new OleDbConnection();
                dataGridView5.DataSource = null;
                F.Find_selected_provider();
                conn.ConnectionString = @"Provider = " +
F.OleDB_selected_provider + "; Data Source = " + path + "; Extended
Properties=Excel 8.0;";
                OleDbCommand Command = new OleDbCommand(query, conn);

                DataSet DS = new DataSet();
                OleDbDataAdapter Adapter = new
OleDbDataAdapter(Command);
                Adapter.Fill(DS);
                dataGridView5.DataSource = DS.Tables[0];
            }
            catch (Exception Error)
            {
                MessageBox.Show(Error.Message);
            }
        }

        private void button38_Click(object sender, EventArgs e)
        {
            try
            {
                string path = this.textBox11.Text;
                string table_name = listBox7.SelectedItem.ToString();
```

```
                string query = textBox12.Text;
                //MessageBox.Show(query);

                OleDbConnection conn = new OleDbConnection();
                dataGridView5.DataSource = null;
                F.Find_selected_provider();
                conn.ConnectionString = @"Provider = " +
F.OleDB_selected_provider + "; Data Source = " + path + "; Extended
Properties=Excel 8.0;";
                OleDbCommand Command = new OleDbCommand(query, conn);

                DataSet DS = new DataSet();
                OleDbDataAdapter Adapter = new
OleDbDataAdapter(Command);
                Adapter.Fill(DS);
                dataGridView5.DataSource = DS.Tables[0];
            }
            catch (Exception Error)
            {
                MessageBox.Show(Error.Message);
            }
        }

        private void button4_Click(object sender, EventArgs e)
        {
            try
            {
                SaveFileDialog Save = new SaveFileDialog();

                Save.Filter = "*.xml|*.xml|*.*|*.*";
                if (Save.ShowDialog() == DialogResult.OK)
                {
                    DataTable Table = new DataTable();
                    for (int i = 0; i < dataGridView2.ColumnCount; i++)
                    {

Table.Columns.Add(dataGridView2.Columns[i].HeaderText.ToString());


//MessageBox.Show(dataGridView2.Columns[i].HeaderText.ToString());
                    }

                    for (int j = 0; j < dataGridView2.RowCount - 1; j++)
                    {
                        DataRow Row = Table.NewRow();
                        Table.Rows.Add(Row);
```

```
                        for (int i = 0; i < dataGridView2.ColumnCount;
i++)
                        {
                            string Val =
dataGridView2.Rows[j].Cells[i].Value.ToString();

                            //MessageBox.Show(Val);
                            if (Val == "") { Val = " "; }
                            Table.Rows[j][i] = Val;
                        }
                    }

                    DataSet ds = new DataSet();
                    ds.Tables.Add(Table);
                    ds.WriteXml(Save.FileName);
                }
            }
            catch (Exception Error)
            {
                MessageBox.Show(Error.Message);
            }
        }
    }
}
```