MANUAL TÉCNICO

FRM

MODULO DE CONTABILIDAD

MAR-MAYO 2020

CONTENIDO DEL MANUAL

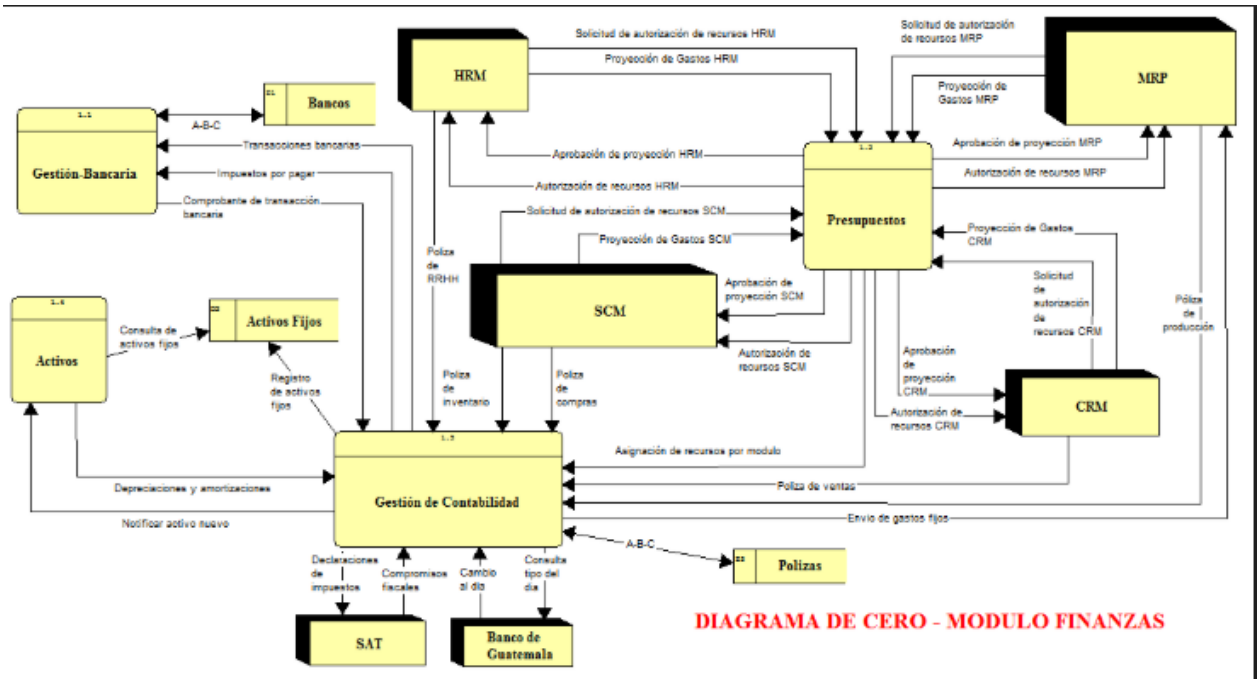
INTRODUCCION

En el siguiente Documento se presenta un detalle técnico del funcionamiento de los mantenimientos y procesos involucrados en el operar del módulo de finanzas, específicamente en el área de contabilidad, para esto se detallará el contenido con los diagramas de análisis del funcionamiento, y ciertas porciones de código claves para la comprensión del trabajo interno del módulo.

DIAGRAMACIÓN

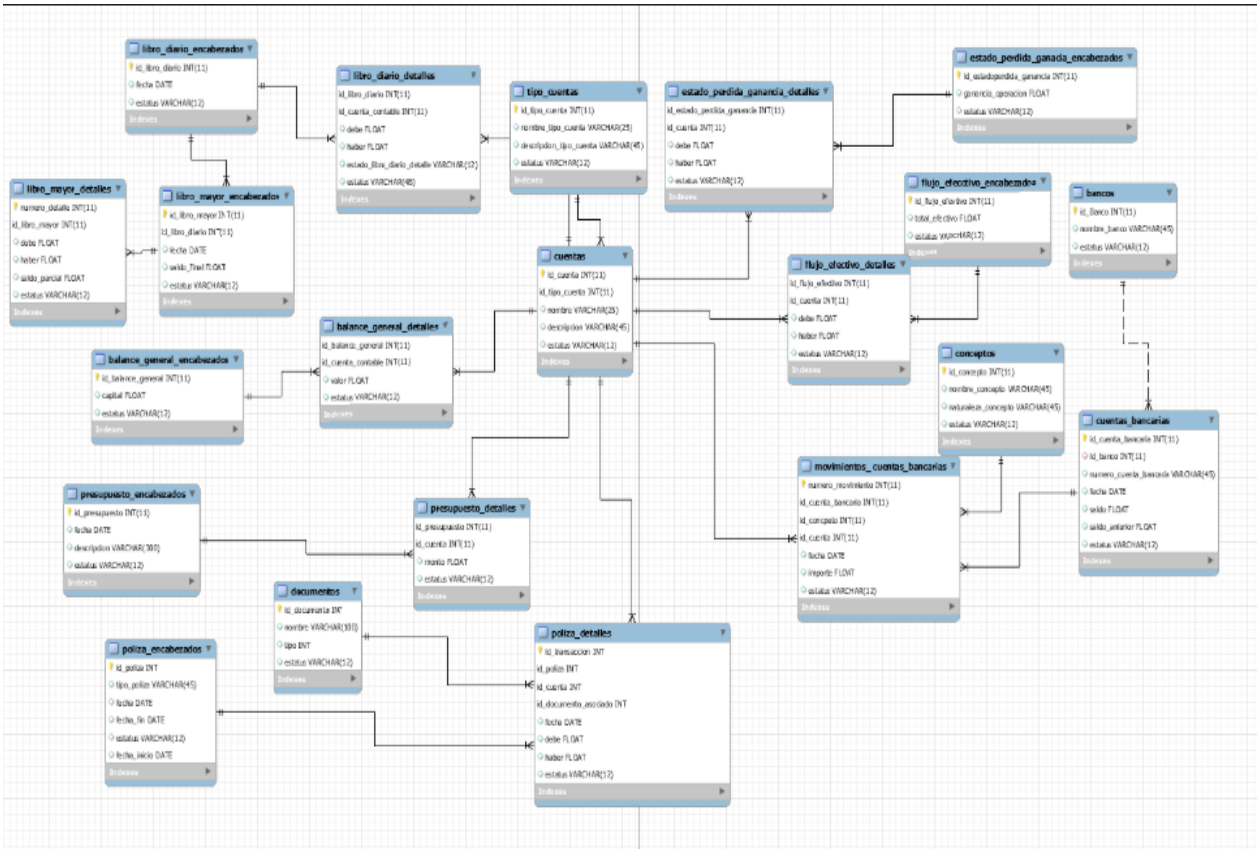
DFD

Diagramas



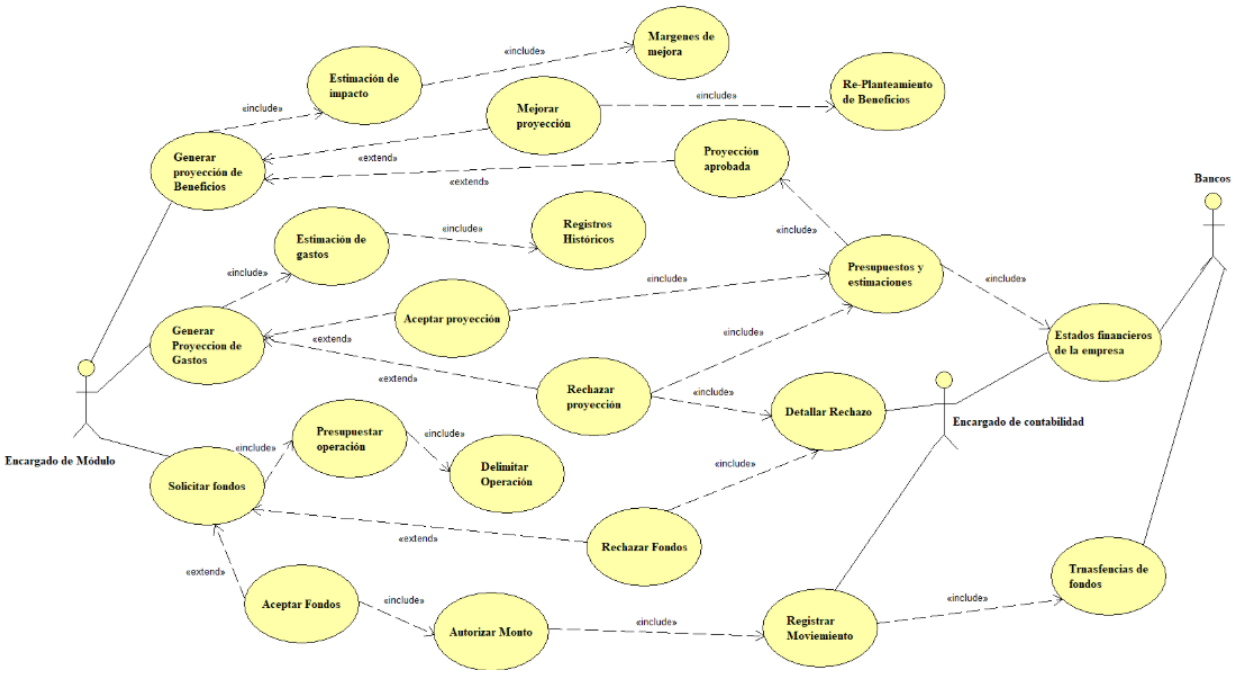
ER

Diagramas



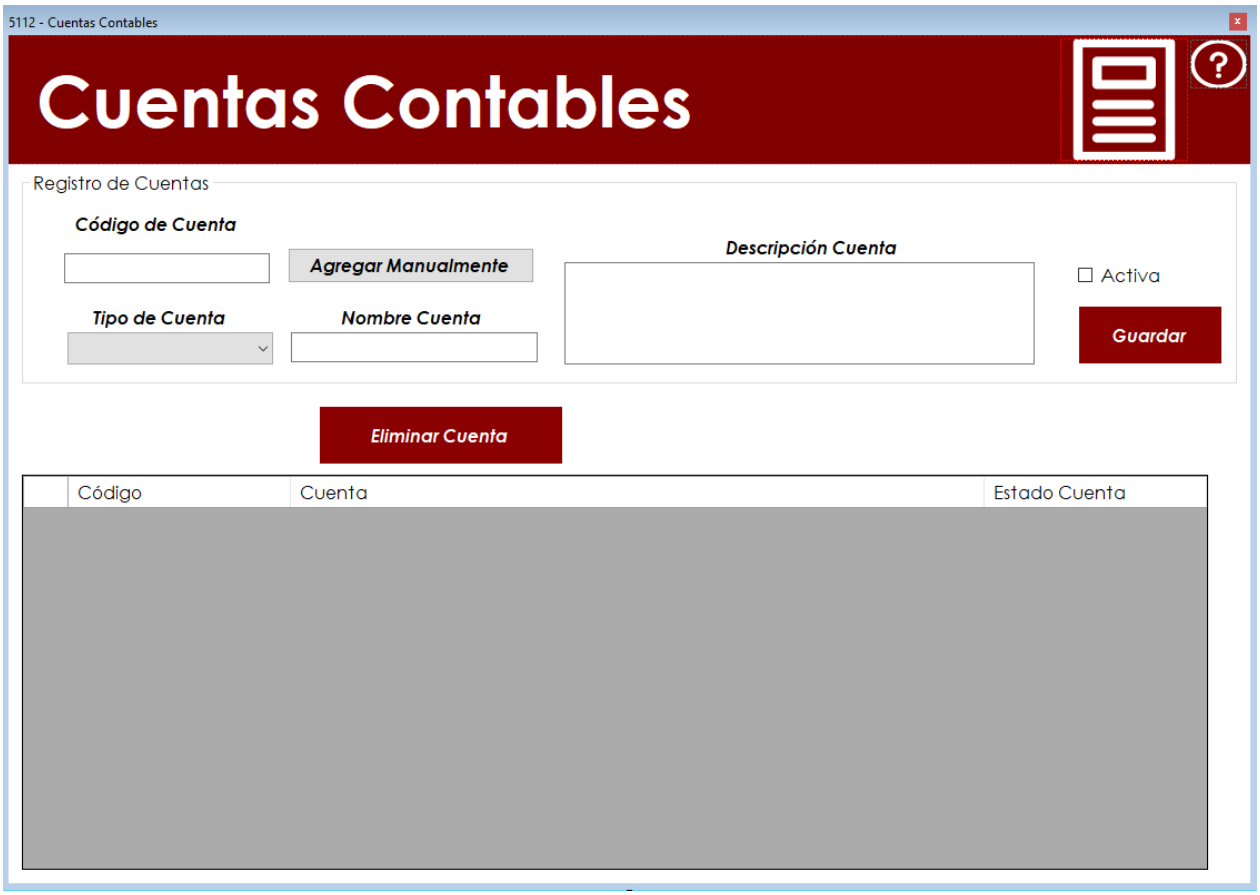
CASO DE USO

Diagramas



MANTENIMIENTOS

Cuentas Contables



Entre los aspectos a resaltar de este mantenimiento está el funcionamiento de los códigos asociados a las cuentas

Primeramente, es importante notar que el código puede ser asignado de forma manual, es decir agregando el código deseado o de forma automática, para la forma automática se consulta los registros de tipo de cuenta, para asociar un tipo de cuenta al código.

```
private void Cmb_Tipo_SelectedIndexChanged(object sender, EventArgs e)
{
    if (estadoBtn == 0)
    {
        if (Cmb_Tipo.Text != "" && Cmb_Tipo.Text != " " && Cmb_Tipo.Text != null)
        {
            Txt_Id.Enabled = false;
            string idTipo = mod.ObtenerNextIdCuenta(mod.ObtenerIdTIpoCuenta(Cmb_Tipo.Text.ToString()));
            Txt_Id.Text = mod.ObtenerIdTIpoCuenta(Cmb_Tipo.Text.ToString());
            Txt_Id.Text += "." + (Convert.ToInt32(idTipo) + 1).ToString();
        }
    }
    else
    {
        ToolTip toolTip1 = new ToolTip();
        toolTip1.AutoPopDelay = 5000;
        toolTip1.ShowAlways = true;
        toolTip1.ReshowDelay = 0;
        toolTip1.SetToolTip(Btn_guardar, "No es posible modificar el Tipo de Cuenta");
    }


}
```

Los códigos de Cuenta se asociarán a la tabla de los tipos de cuenta:

id_cuenta	id_tipo_cuenta	codigo	nombre
0.0	0	0	VARIOS
1.1.1	1.1	1	CAJA
1.1.1.1	1	1	CAJA GENERAL
1.1.2	1.1	2	BANCOS
1.1.2.1	1.1	6	BANCO G&T
1.1.3	1.1	3	CLIENTES
1.1.4	1.1	4	IVA POR COBRAR
1.1.5	1.1	5	CUENTAS POR COBRAR
1.1.7	1.1	7	MERCADERIAS

En el caso de la opción de escritura manual del código fue necesario validar la correcta escritura del código proporcionado, es decir su formato correcto según lo establecido.

id_cuenta
0.0
1.1.1
1.1.1.1
1.1.2
1.1.2.1
1.1.3

Para esto se utilizó una expresión regular para validar la construcción de este código al momento de almacenar el dato

```
private Boolean codigo_correcto(String codigo)
{
    String expresion;
    expresion = @"\d+(\.{1}\d+)+";
    if (Regex.IsMatch(codigo, expresion))
    {
        if (Regex.Replace(codigo, expresion, String.Empty).Length == 0)
        {
            return true;
        }
        else
        {
            return false;
        }
    }
    else
    {
        return false;
    }
}
```

PROCESOS

Libro Diario

Para la elaboración del libro diario fue necesario tomar en cuenta importantes aspectos relacionados a las validaciones de las partidas contables

Entre ellas:

El conteo de Filas de la partida debe ser mayor a 0

```
if (fil==0)
{
    MessageBox.Show("No se han ingresado partidas");
    proceder = false;
}
```

Toda partida debe poseer un concepto

```
if (Txt_Concepto.Text=="")
{
    MessageBox.Show("Por favor coloque un un concepto a la partida");
    proceder = false;
}
```

No pueden existir valore vacíos en las celdas de las partidas

```
for (int i = 0; i < fil; i++)
{
    for (int j = 0; j < col; j++)
    {
        if (Dtg_Movimientos.Rows[i].Cells[j].Value == null)
        {
            MessageBox.Show("Por Favor llene la tabla en la fila " + (i+1).ToString()+" y Columna " + (j+1).ToString());
            proceder = false;
        }
        else if (Dtg_Movimientos.Rows[i].Cells[j].Value.ToString() =="" || Dtg_Movimientos.Rows[i].Cells[j].Value.ToString()==" ")
        {
            MessageBox.Show("Por Favor llene la tabla en la fila " + (i + 1).ToString() + " y Columna " + (j + 1).ToString());
            proceder = false;
        }

    }
}
```

Toda partida debe tener sumas iguales

```
for (int i = 0; i < Dtg_Movimientos.Rows.Count-1; i++)
{
    debe += Convert.ToDecimal( Dtg_Movimientos.Rows[i].Cells[1].Value);
    haber += Convert.ToDecimal(Dtg_Movimientos.Rows[i].Cells[2].Value);
}
if (debe!=haber)
{
    MessageBox.Show("Las sumas no son iguales, revise ambas columnas");
    proceder = false;
}
```

En la eliminación de partidas también se validó la correcta decisión al eliminar la partida, es decir se pregunta al usuario si se desea realizar una acción destructiva

```
concepto = Dtg_Partidas.CurrentRow.Cells[1].Value.ToString();
DialogResult Eliminar;
Eliminar = MessageBox.Show("Se eliminarán todos los movimientos con concepto " + concepto + ", desea continuar?", "Eliminar Partida", MessageBoxButtons.YesNo);
```

Libro Mayor

Para la elaboración del libro mayor se realizo el siguiente proceso para trasladar los datos del libro diario a registros de libro Mayor, para ello se utilizó el siguiente query de consulta

Para este proceso se utilizó el listado de las cuantas que pertenecen al libro seleccionado

```
int noDetalle = 0;
OdbcDataAdapter ODBC_Cuentas = Mayor.Cuentas(idLibroDiario);//Cuentas
DataTable cuentas = new DataTable();// Tabla de Cuentas
//DEBE
OdbcDataAdapter ODBC_Partidas_Cantidad_Debe = new OdbcDataAdapter();//Partidas  en las que parace en debe una cuenta y cantidad
OdbcDataAdapter ODBC_Cuenta_Opuesta_Debe = new OdbcDataAdapter(); //Cuenta que aparece en el haber con la misma cantidad
DataTable Partidas_Cantidad_Debe = new DataTable(); // Tabla de partidas y cantidades
DataTable Cuenta_Opuesta_Debe = new DataTable(); // Cuenta opuesta
 // HABER
OdbcDataAdapter ODBC_Partidas_Cantidad_Haber = new OdbcDataAdapter();
DataTable Partidas_Cantidad_Haber = new DataTable();
OdbcDataAdapter ODBC_Cuenta_Opuesta_Haber = new OdbcDataAdapter();
DataTable Cuenta_Opuesta_Haber = new DataTable();
```

Luego de esto se itera sobre cada una de las cuentas para poder realizar los ingresos respectivos a libro mayor

Sobre El debe de las cuentas

```
foreach (DataRow partidadebe in Partidas_Cantidad_Debe.Rows)
{
    //MessageBox.Show("ENTRO A PARTIDAS");
    noDetalle++;
    //MessageBox.Show("DEBE NO PARTIDA "+partidadebe[0].ToString());
    //MessageBox.Show("DEBE CANTIDAD "+partidadebe[1].ToString());
    ODBC_Cuenta_Opuesta_Debe = Mayor.Cuenta_Opuesta_Debe(partidadebe[0].ToString(), partidadebe[1].ToString(), idLibroDiario);
    Cuenta_Opuesta_Debe.Clear();
    ODBC_Cuenta_Opuesta_Debe.Fill(Cuenta_Opuesta_Debe);
    //OPUESTO DEBE
    //MessageBox.Show("DEBE CUENTA " + cuenta[0].ToString());
    //MessageBox.Show("DEBE OPUESTA" + Cuenta_Opuesta_Debe.Rows[0].ItemArray[0].ToString());
    InsertarDetalleMayor(noDetalle.ToString(), idLibroMayor, partidadebe[0].ToString(), cuenta[0].ToString(), Cuenta_Opuesta_Debe.Rows[0].ItemArray[0].ToString(), "0",partidadebe
    //MessageBox.Show("SALIO DE PARTIDAS");
}
```

Sobre el Haber de las cuentas

```
foreach (DataRow partida in Partidas_Cantidad_Haber.Rows)
{
    noDetalle++;
    //MessageBox.Show("HABER NO PARTIDA " + partida[0].ToString());
    //MessageBox.Show("HABER CANTIDAD " + partida[1].ToString());
    ODBC_Cuenta_Opuesta_Haber = Mayor.Cuenta_Opuesta_Haber(partida[0].ToString(), partida[1].ToString(),idLibroDiario);
    Cuenta_Opuesta_Haber.Clear();
    ODBC_Cuenta_Opuesta_Haber.Fill(Cuenta_Opuesta_Haber);
    //MessageBox.Show("HABER OPUESTA " + Cuenta_Opuesta_Haber.Rows[0].ItemArray[0].ToString());


        //MessageBox.Show("HABER CUENTA " + cuenta[0].ToString());
        //MessageBox.Show("HABER OPUESTA" + Cuenta_Opuesta_Haber.Rows[0].ItemArray[0].ToString());
        InsertarDetalleMayor(noDetalle.ToString(), idLibroMayor, partida[0].ToString(), cuenta[0].ToString(), Cuenta_Opuesta_Haber.Rows[0].ItemArray[0].ToString(), parti

}
```

Una vez Realizados estos ingresos se procede a realizar la consulta necesaria para mostrar el libreo con una estructura correcta

```
foreach (DataRow cuenta in cuentas.Rows)
{
    //MessageBox.Show("Cuenta " + cuenta[0]);
    if (i==cuentas.Rows.Count-1)
    {
        query += "SELECT '','','"+cuenta[0].ToString()+"','' "+
                "UNION ALL "+
                "SELECT id_partida, concat('a: ', cuenta_contable), concat('Q.', haber), '' " +
                "FROM libro_mayor_detalles WHERE cuenta_mayor = '" + cuenta[0].ToString() + "' AND id_libro_mayor = "+idlibroMayor+" AND haber<>0 " +
                "UNION ALL "+
                "SELECT '','','', IF(ROUND(SUM(haber), 2) > 0, CONCAT('SALDO: Q.', ROUND(SUM(haber))), '') " +
                "FROM libro_mayor_detalles WHERE cuenta_mayor = '" + cuenta[0].ToString() + "' AND id_libro_mayor = " + idlibroMayor + " AND haber<>0 " +
                "UNION ALL "+
                "SELECT id_partida, concat('por: ', cuenta_contable), concat('Q.', debe), '' FROM libro_mayor_detalles " +
                "WHERE cuenta_mayor = '" + cuenta[0].ToString() + "' AND id_libro_mayor = " + idlibroMayor + " AND debe<>0 " +
                "UNION ALL "+
                "SELECT '','','', IF((ROUND(SUM(debe), 2)) > 0, CONCAT('SALDO: Q.', ROUND(SUM(debe))), '') " +
                "FROM libro_mayor_detalles WHERE cuenta_mayor = '" + cuenta[0].ToString() + "' AND id_libro_mayor = " + idlibroMayor + " AND debe<>0 ;";
    }
    else
    {
        query += "SELECT '','','" + cuenta[0].ToString() + "','' " +
                "UNION ALL " +
                "SELECT id_partida, concat('a: ', cuenta_contable), concat('Q.', haber), ''  " +
                "FROM libro_mayor_detalles WHERE cuenta_mayor = '" + cuenta[0].ToString() + "' AND id_libro_mayor = " + idlibroMayor + " AND haber<>0 " +
                "UNION ALL " +
                "SELECT '','','', IF(ROUND(SUM(haber), 2) > 0, CONCAT('SALDO: Q.', ROUND(SUM(haber))), '') " +
                "FROM libro_mayor_detalles WHERE cuenta_mayor = '" + cuenta[0].ToString() + "' AND id_libro_mayor = " + idlibroMayor + " AND haber<>0 " +
                "UNION ALL " +
                "SELECT id_partida, concat('por: ', cuenta_contable), concat('Q.', debe), '' FROM libro_mayor_detalles " +
                "WHERE cuenta_mayor = '" + cuenta[0].ToString() + "' AND id_libro_mayor = " + idlibroMayor + " AND debe<>0 " +
                "UNION ALL " +
                "SELECT '','','', IF((ROUND(SUM(debe), 2)) > 0, CONCAT('SALDO: Q.', ROUND(SUM(debe))), '') " +
                "FROM libro_mayor_detalles WHERE cuenta_mayor = '" + cuenta[0].ToString() + "' AND id_libro_mayor = " + idlibroMayor + " AND debe<>0 UNION ALL ";
    }
    i++;

}
return query;
```

Este Query opera sobre cada una de las cuentas realizando la unión de las consultas seleccionadas

```
query += "SELECT '','','"+cuenta[0].ToString()+"','' "+
        "UNION ALL "+
        "SELECT id_partida, concat('a: ', cuenta_contable), concat('Q.', haber), '' " +
        "FROM libro_mayor_detalles WHERE cuenta_mayor = '" + cuenta[0].ToString() + "' AND id_libro_mayor = "+idlibroMayor+" AND haber<>0 " +
        "UNION ALL "+
        "SELECT '','','', IF(ROUND(SUM(haber), 2) > 0, CONCAT('SALDO: Q.', ROUND(SUM(haber))), '') " +
        "FROM libro_mayor_detalles WHERE cuenta_mayor = '" + cuenta[0].ToString() + "' AND id_libro_mayor = " + idlibroMayor + " AND haber<>0 " +
        "UNION ALL "+
        "SELECT id_partida, concat('por: ', cuenta_contable), concat('Q.', debe), '' FROM libro_mayor_detalles " +
        "WHERE cuenta_mayor = '" + cuenta[0].ToString() + "' AND id_libro_mayor = " + idlibroMayor + " AND debe<>0 " +
        "UNION ALL "+
        "SELECT '','','', IF((ROUND(SUM(debe), 2)) > 0, CONCAT('SALDO: Q.', ROUND(SUM(debe))), '') " +
        "FROM libro_mayor_detalles WHERE cuenta_mayor = '" + cuenta[0].ToString() + "' AND id_libro_mayor = " + idlibroMayor + " AND debe<>0 ;";
```

Libro Balances

Para la elaboración del libro balances fue necesario tomar en cuenta importantes aspectos relacionados el cálculo de los saldos de las cuentas contables

Para realizar este proceso se obtuvo las cuantos presentes en ese libro como primer paso

```
OdbcDataAdapter ODBC_Cuentas = Mayor.Cuentas(idLibroMayor);//Cuentas
DataTable cuentas = new DataTable();// Tabla de Cuentas
//DEBE

OdbcDataAdapter ODBC_Debe_Haber_Cuenta = new OdbcDataAdapter(); //Cuenta que aparece en el haber con la misma cantidad
DataTable Debe_Haber_Cuenta = new DataTable(); // Tabla de partidas y cantidades

ODBC_Cuentas.Fill(cuentas);
valorxCuenta = 100/cuentas.Rows.Count;
//APARICION
// DEBE
foreach (DataRow cuenta in cuentas.Rows)...

ODBC_Debe_Haber_Cuenta = Mayor.Debe_Haber(sql);
ODBC_Debe_Haber_Cuenta.Fill(Debe_Haber_Cuenta);
string[] valores = Mayor.ArrayCuentas(idLibroMayor).Split(',');
```

Para cada una de las cuentas se consulto su total en Debe y en haber

Esto con la finalidad de tener los dos datos necesarios para determinar si el saldo es deudor o es acreedor

```
foreach (DataRow cuenta in cuentas.Rows)
{
    if (i==cuentas.Rows.Count-1)
    {
        sql += "SELECT " +
            "IF ((SELECT ROUND(SUM(debe), 2) FROM libro_diario_detalles WHERE debe <>0 AND cuenta_contable = '" + cuenta[0] + "' AND id_libro_diario = (SELECT id_libro_diario FROM libro_may
            "(SELECT ROUND(SUM(debe), 2) FROM libro_diario_detalles WHERE debe <>0 AND cuenta_contable = '" + cuenta[0] + "' AND id_libro_diario = (SELECT id_libro_diario FROM libro_mayor_e
            "IF((SELECT ROUND(SUM(haber), 2) FROM libro_diario_detalles WHERE haber<>0 AND cuenta_contable = '" + cuenta[0] + "' AND id_libro_diario = (SELECT id_libro_diario FROM libro_may
            "(SELECT ROUND(SUM(haber), 2) FROM libro_diario_detalles WHERE haber<>0 AND cuenta_contable = '" + cuenta[0] + "' AND id_libro_diario = (SELECT id_libro_diario FROM libro_mayor_
    }
    else
    {
        sql += "SELECT " +
            "IF ((SELECT ROUND(SUM(debe), 2) FROM libro_diario_detalles WHERE debe <>0 AND cuenta_contable = '" + cuenta[0] + "' AND id_libro_diario = (SELECT id_libro_diario FROM libro_may
            "(SELECT ROUND(SUM(debe), 2) FROM libro_diario_detalles WHERE debe <>0 AND cuenta_contable = '" + cuenta[0] + "' AND id_libro_diario = (SELECT id_libro_diario FROM libro_mayor_e
            "IF((SELECT ROUND(SUM(haber), 2) FROM libro_diario_detalles WHERE haber<>0 AND cuenta_contable = '" + cuenta[0] + "' AND id_libro_diario = (SELECT id_libro_diario FROM libro_may
            "(SELECT ROUND(SUM(haber), 2) FROM libro_diario_detalles WHERE haber<>0 AND cuenta_contable = '" + cuenta[0] + "' AND id_libro_diario = (SELECT id_libro_diario FROM libro_mayor_
    }

    i++;

}
```

Proceso se que llevo a cabo comparando los saldos de cada una de las cuantas involucradas

```
foreach (DataRow debehaber in Debe_Haber_Cuenta.Rows)
{
        noDetalle++;
        if (progreso + Convert.ToInt32(valorxCuenta) < 100)
        {
            progreso += Convert.ToInt32(valorxCuenta);
        }
        else
        {
            progreso = 100;
        }

        //MessageBox.Show("DEBE " + debehaber[0].ToString());
        //MessageBox.Show("HABER " + debehaber[1].ToString());
        if (debehaber[0].ToString() == "" || debehaber[0].ToString() == " " || debehaber[0] == null)
        {
            debe = 0;
        }
        else
        {
            debe = (float)Convert.ToDouble(debehaber[0].ToString());
        }

        if (debehaber[1].ToString() == "" || debehaber[1].ToString() == " " || debehaber[1] == null)
        {
            haber = 0;
        }
        else
        {
            haber = (float)Convert.ToDouble(debehaber[1].ToString());
        }
        //MessageBox.Show("debe " + debe.ToString());
        //MessageBox.Show("haber " + haber.ToString());
        if (debe > haber || debe == haber)
        {
            deudor = debe - haber;
            acreedor = 0;
        }
        else
        {
            acreedor = haber - debe;
            deudor = 0;
        }
```

Una vez ingresados todos los datos necesarios se procedió a crear el query necesario para la construcción de la vista del libro

```
public OdbcDataAdapter llenarBalance(string idBalance)
{

    string query = "SELECT numero,  id_balance, cuenta_contable,     IF (debe>0,concat('Q.', debe),''), IF (haber>0,concat('Q.', haber),''),    " +
        " IF(deudor>0,concat('Q.', deudor),''), IF(acreedor>0,concat('Q.', acreedor),'') FROM `balances_detalles` where id_balance = " + idBalance+" " +
        " UNION ALL " +
        " SELECT '', '', 'SUMAS IGUALES', concat('Q.', ROUND(SUM(debe), 2))," +
        " concat('Q.', ROUND(SUM(haber), 2)), concat('Q.', ROUND(SUM(deudor), 2)), " +
        " concat('Q.', ROUND(SUM(acreedor), 2)) FROM `balances_detalles` where id_balance = " + idBalance+"";

    OdbcDataAdapter dataTable = Mayor.LlenarTablaMayor(query);
    return dataTable;
}
```

Estado de Resultados

Para la elaboración del Estado de Resultados fue necesario tomar en cuenta aspectos relacionados a las cuentas involucradas y sus tipos

En este caso Ingresos Compras y Gastos

```
OdbcDataAdapter ODBC_IN = Mayor.Ingresos(idLibroBalances);//AC
DataTable ing = new DataTable();// Tabla AC
ODBC_IN.Fill(ing);

OdbcDataAdapter ODBC_COM = Mayor.Compras(idLibroBalances);//ANC
DataTable  com = new DataTable();// Tabla ANC
ODBC_COM.Fill(com);

OdbcDataAdapter ODBC_GAS = Mayor.Gastos(idLibroBalances);//PC
DataTable gas = new DataTable();// Tabla PC
ODBC_GAS.Fill(gas);
```

Para realizar el proceso de inserción de datos se iteró sobre cada uno de estos listados de cuentas, según corresponda su tipo

```
foreach (DataRow cuenta in ing.Rows)
{
    noDetalle++;
    ingresos += Convert.ToInt32(cuenta[1]);
    InsertarDetalleEstadoDeResultados(noDetalle.ToString(), idEstadoDeResultados, cuenta[0].ToString(), cuenta[1].ToString());
}
progreso = 25;
foreach (DataRow cuenta in com.Rows)
{
    noDetalle++;
    compras += Convert.ToInt32(cuenta[1]);
    InsertarDetalleEstadoDeResultados(noDetalle.ToString(), idEstadoDeResultados, cuenta[0].ToString(), cuenta[1].ToString());
}
progreso = 50;
foreach (DataRow cuenta in gas.Rows)
{
    noDetalle++;
    gastos += Convert.ToInt32(cuenta[1]);
    InsertarDetalleEstadoDeResultados(noDetalle.ToString(), idEstadoDeResultados, cuenta[0].ToString(), cuenta[1].ToString());
}
```

Con estos datos ingresados se procede a la creación del query necesario para dar forma a la vista estructurada del libro

```
public OdbcDataAdapter llenarBalance(string idBalance)
{

    string query = "SELECT 'INGRESOS', '','','' "+
                   "UNION ALL "+
                   "SELECT cuenta_contable, saldo,'','' FROM estado_de_resultados_detalles  WHERE id_estado_de_resultado = " + idBalance + " AND '4.1' = (SELECT id_tipo_cuenta from cuentas WHERE n
                   "UNION ALL "+
                   "SELECT 'TOTAL INGRESOS','',ROUND(SUM(saldo),2),'' FROM estado_de_resultados_detalles   WHERE id_estado_de_resultado = " + idBalance + " AND '4.1' = (SELECT id_tipo_cuenta from
                   "UNION ALL "+
                   "SELECT 'COMPRAS', '','','' "+
                   "UNION ALL " +
                   "SELECT cuenta_contable, saldo,'','' FROM estado_de_resultados_detalles   WHERE id_estado_de_resultado = " + idBalance + " AND '5.1' = (SELECT id_tipo_cuenta from cuentas WHERE
                   "UNION ALL "+
                   "SELECT 'TOTAL COMPRAS','',ROUND(SUM(saldo),2),'' FROM estado_de_resultados_detalles   WHERE id_estado_de_resultado = " + idBalance + " AND '5.1' = (SELECT id_tipo_cuenta from c
                   "UNION ALL " +
                   "SELECT 'GANANCIA DE VENTAS','','', " +
                   "ROUND(( "+
                   "(SELECT SUM(saldo) FROM estado_de_resultados_detalles   WHERE id_estado_de_resultado = " + idBalance + " AND '4.1' = (SELECT id_tipo_cuenta from cuentas WHERE nombre = cuenta_c
                   "- "+
                   "(SELECT SUM(saldo) FROM estado_de_resultados_detalles   WHERE id_estado_de_resultado = " + idBalance + " AND '5.1' = (SELECT id_tipo_cuenta from cuentas WHERE nombre = cuenta_c
                   "),2) "+
                   "UNION ALL "+
                   "SELECT 'GASTOS', '','','' "+
                   "UNION ALL "+
                   "SELECT cuenta_contable, saldo,'','' FROM estado_de_resultados_detalles   WHERE id_estado_de_resultado = " + idBalance + " AND '6.1' = (SELECT id_tipo_cuenta from cuentas WHERE
                   "UNION ALL "+
                   "SELECT 'TOTAL GASTOS','',ROUND(SUM(saldo),2),'' FROM estado_de_resultados_detalles   WHERE id_estado_de_resultado = " + idBalance + " AND '6.1' = (SELECT id_tipo_cuenta from cu
                   "UNION ALL "+
                   "SELECT 'GANANCIA DEL EJERCICIO ANTES DE IMPUESTO','','', "+
                   "ROUND(( "+
                   "(SELECT SUM(saldo) FROM estado_de_resultados_detalles   WHERE id_estado_de_resultado = " + idBalance + " AND '4.1' = (SELECT id_tipo_cuenta from cuentas WHERE nombre = cuenta_c
                   "- "+
                   "(SELECT SUM(saldo) FROM estado_de_resultados_detalles   WHERE id_estado_de_resultado = " + idBalance + " AND '5.1' = (SELECT id_tipo_cuenta from cuentas WHERE nombre = cuenta_c
                   "- "+
                   "(SELECT SUM(saldo) FROM estado_de_resultados_detalles   WHERE id_estado_de_resultado = " + idBalance + " AND '6.1' = (SELECT id_tipo_cuenta from cuentas WHERE nombre = cuenta_c
                   "),2) ;";


    OdbcDataAdapter dataTable = Mayor.LlenarTablaMayor(query);
    return dataTable;
}
```

Tomando en cuenta el orden de las cuentas en su aparición, es decir, los tipos de cuentas, Ingresos, Compras y gastos

```
"SELECT 'INGRESOS', '','','' "+
"UNION ALL "+
"SELECT cuenta_contable, saldo,'','' FROM estado_de_resultados_detalles  WHERE id_estado_de_resultado = " + idBalance + " AND '4.1' =
```

```
"SELECT 'COMPRAS', '','','' "+
"UNION ALL " +
"SELECT cuenta_contable, saldo,'','' FROM estado_de_resultados_detalles   WHERE id_estado_de_resultado = " + idBalance + " AND '5.1' = (
"UNION ALL "+
"SELECT 'TOTAL COMPRAS','',ROUND(SUM(saldo),2),'' FROM estado_de_resultados_detalles   WHERE id_estado_de_resultado = " + idBalance + "
```

```
"SELECT 'GASTOS', '','','' "+
"UNION ALL "+
"SELECT cuenta_contable, saldo,'','' FROM estado_de_resultados_detalles   WHERE id_estado_de_resultado = " + idBalance + " AND '6.1'
"UNION ALL "+
```

Balance General

Para la elaboración del libro balances fue necesario tomar en cuenta importantes aspectos relacionados las cuentas utilizadas

Para esto se tomó en cuenta las cuentas de activo en corriente, y no corriente, además de las cuentas de pasivo corriente y no corriente.

Por último, la parte del patrimonio

```
OdbcDataAdapter ODBC_AC = Mayor.ActivoC(idLibroBalances);//AC
DataTable ac = new DataTable();// Tabla AC
ODBC_AC.Fill(ac);

OdbcDataAdapter ODBC_ANC = Mayor.ActivoNC(idLibroBalances);//ANC
DataTable anc = new DataTable();// Tabla ANC
ODBC_ANC.Fill(anc);

OdbcDataAdapter ODBC_PC = Mayor.PasivoC(idLibroBalances);//PC
DataTable pc = new DataTable();// Tabla PC
ODBC_PC.Fill(pc);

OdbcDataAdapter ODBC_PNC = Mayor.PasivoNC(idLibroBalances);//PNC
DataTable pnc = new DataTable();// Tabla PNC
ODBC_PNC.Fill(pnc);

OdbcDataAdapter ODBC_PAT = Mayor.Patrimonio(idLibroBalances);//Patrimonio
DataTable pat = new DataTable();// Tabla Patrimonio
ODBC_PAT.Fill(pat);
```

Se itera sobre cada uno del listado se cuenta según su tipo para poder obtener los saldos y realizar los ingresos de cada uno de estos

```
foreach (DataRow cuenta in ac.Rows)
{
    noDetalle++;
    activo += Convert.ToInt32(cuenta[1]);
    InsertarDetalleBalanceGeneral(noDetalle.ToString(), idBalanceGeneral, cuenta[0].ToString(), cuenta[1].ToString());
}
progreso = 25;
foreach (DataRow cuenta in anc.Rows)
{
    noDetalle++;
    activo += Convert.ToInt32(cuenta[1]);
    InsertarDetalleBalanceGeneral(noDetalle.ToString(), idBalanceGeneral, cuenta[0].ToString(), cuenta[1].ToString());
}
progreso = 50;
foreach (DataRow cuenta in pc.Rows)
{
    noDetalle++;
    pasivo += Convert.ToInt32(cuenta[1]);
    InsertarDetalleBalanceGeneral(noDetalle.ToString(), idBalanceGeneral, cuenta[0].ToString(), cuenta[1].ToString());
}
progreso = 75;
foreach (DataRow cuenta in pnc.Rows)
{
    noDetalle++;
    pasivo += Convert.ToInt32(cuenta[1]);
    InsertarDetalleBalanceGeneral(noDetalle.ToString(), idBalanceGeneral, cuenta[0].ToString(), cuenta[1].ToString());
}
foreach (DataRow cuenta in pat.Rows)
{
    noDetalle++;
    patrimonio += Convert.ToInt32(cuenta[1]);
    InsertarDetalleBalanceGeneral(noDetalle.ToString(), idBalanceGeneral, cuenta[0].ToString(), cuenta[1].ToString());
}
```

Con los datos ya ingresados se procede a crear los querys necesarios para las creación de la vista del Balance General

```
string query = "SELECT 'ACTIVO CORRIENTE', '','','' "+
               "UNION ALL "+
               "SELECT cuent  string string.operator +(string left, string right)  general_detalles  WHERE id_balance_general =" + idBalance + " AND '1.1' = (SELECT id_tipo_cuenta from cuentas WHERE nombre = cue
               "UNION ALL "+
               "SELECT 'TOTAL ACTIVO CORRIENTE','', ROUND(SUM(saldo),2),'' FROM balance_general_detalles   WHERE id_balance_general ="+idBalance+" AND '1.1' = (SELECT id_tipo_cuenta from cuenta
               "UNION ALL "+
               "SELECT 'ACTIVO NO CORRIENTE', '','','' "+
               "UNION ALL "+
               "SELECT cuenta_contable, saldo,'','' FROM balance_general_detalles   WHERE id_balance_general = "+ idBalance + " AND '1.2' = (SELECT id_tipo_cuenta from cuentas WHERE nombre = cu
               "UNION ALL "+
               "SELECT 'TOTAL ACTIVO NO CORRIENTE','', ROUND(SUM(saldo),2),'' FROM balance_general_detalles   WHERE id_balance_general = "+ idBalance + " AND '1.2' = (SELECT id_tipo_cuenta from
               "UNION ALL "+
               "SELECT 'TOTAL ACTIVO','','', ROUND(SUM(saldo),2) FROM balance_general_detalles   WHERE id_balance_general = "+ idBalance +" AND('1.1' = (SELECT id_tipo_cuenta from cuentas WHERE
               "UNION ALL "+
               "SELECT 'PASIVO CORRIENTE', '','','' "+
               "UNION ALL "+
               "SELECT cuenta_contable, saldo,'','' FROM balance_general_detalles   WHERE id_balance_general = " + idBalance + " AND '2.1' = (SELECT id_tipo_cuenta from cuentas WHERE nombre = c
               "UNION ALL "+
               "SELECT 'TOTAL PASIVO CORRIENTE','', ROUND(SUM(saldo),2),'' FROM balance_general_detalles   WHERE id_balance_general = " + idBalance + " AND '2.1' = (SELECT id_tipo_cuenta from c
               "UNION ALL "+
               "SELECT 'PASIVO NO CORRIENTE', '','','' "+
               "UNION ALL "+
               "SELECT cuenta_contable, saldo,'','' FROM balance_general_detalles   WHERE id_balance_general = " + idBalance + " AND '2.2' = (SELECT id_tipo_cuenta from cuentas WHERE nombre = c
               "UNION ALL "+
               "SELECT 'TOTAL PASIVO NO CORRIENTE','', ROUND(SUM(saldo),2),'' FROM balance_general_detalles   WHERE id_balance_general = " + idBalance + " AND '2.2' = (SELECT id_tipo_cuenta fro
               "UNION ALL "+
               "SELECT 'TOTAL PASIVO','','',  ROUND(SUM(saldo),2) FROM balance_general_detalles   WHERE id_balance_general = " + idBalance + " AND('2.1' = (SELECT id_tipo_cuenta from cuentas WH
               "UNION ALL "+ // CAPITAL
               "SELECT cuenta_contable,'', saldo,'' FROM balance_general_detalles   WHERE id_balance_general = " + idBalance + " AND '3.1' = (SELECT id_tipo_cuenta from cuentas WHERE nombre = c
               "UNION ALL "+
               "SELECT 'GANANCIA DEL EJECICIO','', "+
               "ROUND(("+
               "(SELECT ROUND(SUM(saldo),2) FROM balance_general_detalles   WHERE id_balance_general = " + idBalance + " AND('1.1' = (SELECT id_tipo_cuenta from cuentas WHERE nombre = cuenta_co
               "- "+
```

REFERENCIAS

Google Drive

Repositorio

AUTORIA

AUTORIA	
CREADO POR	NOMBRE Y APELLIDO
CREADO PARA	GRUPO BIENESTAR
CREADO EN	MARZO-MAYO 2020