



# MANUAL TÉCNICO

---

## SCM

---

**MODULO DE SUMINISTROS**  
MAR-MAYO 2020

## CONTENIDO DEL MANUAL

<b>INTRODUCCION</b>	<b>2</b>
<b>DIAGRAMACIÓN</b>	<b>2</b>
DFD	2
ER	2
CASO DE USO	2
<b>MANTENIMIENTOS</b>	<b>2</b>
<b>PROCESOS</b>	<b>5</b>
Lista	5
Movimientos	6
Ordenes de Compra	8
<b>REFERENCIAS</b>	<b>10</b>
Carpetas	10
Repositorio	10

## INTRODUCCION

El siguiente documento cuenta con instrucciones a nivel tecnico del modulo de SCM. Incluye enlaces a las carpetas en drive a los diagramas de analisis del modulo y la descripcion de Mantenimientos y su implementacion. Tambien se cuenta con la descripcion de los procesos principales del modulo.

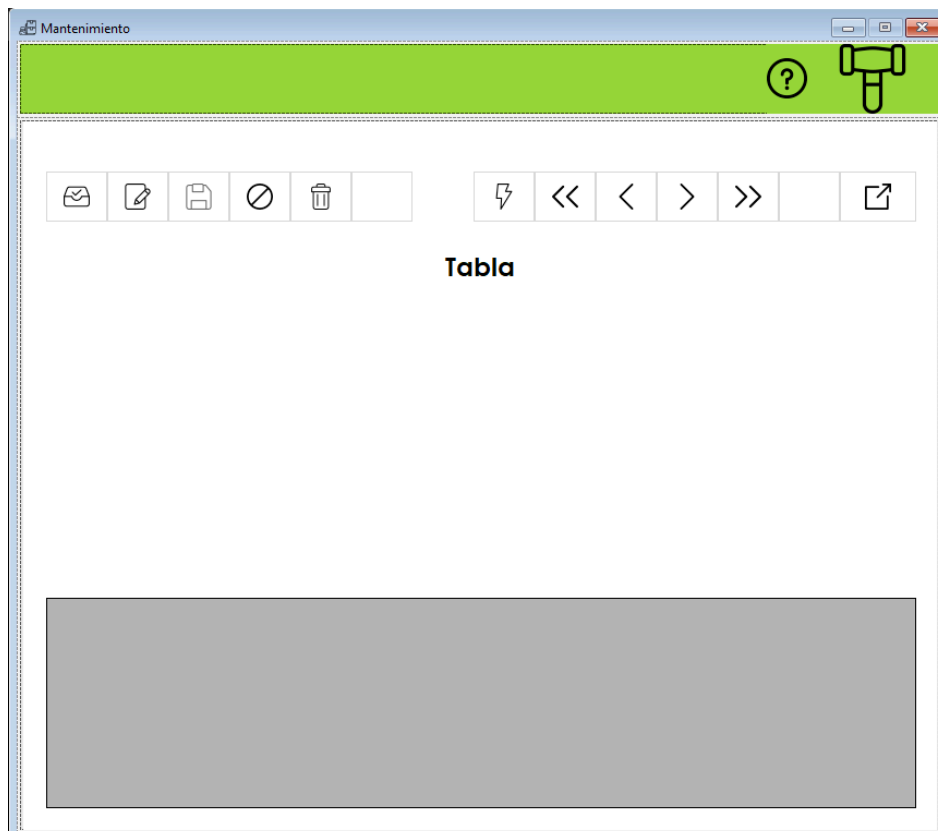
## DIAGRAMACIÓN

[DFD](#)

[ER](#)

[CASO DE USO](#)

## MANTENIMIENTOS



Para el modulo SCM se uso un solo form de mantenimientos en la capa Vista de manera que se llenan desde la capa Modelo. Se puede agregar un nuevo mantenimiento agregando una tabla a partir del id 15. Los mantenimientos

previamente ingresados se encuentran en el siguiente comentario dentro del archivo **Cls\_mantenimientos.cd** dentro de la **CapaModeloSCM**.

#### ID DE TABLAS:

- 1 = contactos
- 2 = proveedores
- 3 = tipos\_productos
- 4 = impuestos
- 5 = categorias
- 6 = acreedores
- 7 = servicios
- 8 = marcas
- 9 = lineas
- 10= presentaciones
- 11= bodegas
- 12= documentos
- 13= tipos\_movimientos
- 14= productos
- 15= inventarios

#### ORDEN DE LOS DATOS EN RETURN PARA datos:

- 1 = alias
- 2 = ayuda
- 3 = tabla
- 4 = form
- 5 = nombre
- 6 = noForaneas

#### ORDEN DE LOS DATOS EN RETURN PARA foraneas:

- 1 = tabla
- 2 = campo
- 3 = modo

Las funciones que posee son 2:

1. `public (string[], string, string, string, string, int) datos(int tabla)`
2. `public (string, string, int) foraneas(int tabla, int no)`

En la primera función se pide el id de la tabla a trabajar y se devuelven los datos a usar en el form de mantenimiento. Se devuelven los siguientes datos en el siguiente orden:

1. ALIAS (lista de nombres de los campos de la tabla)
2. AYUDAS (id de ayudas en la base de datos)
3. TABLA (el nombre de la tabla para mantenimiento)
4. FORM (el titulo que aparecerá en el form)
5. NOMBRE (el titulo que aparecerá en el form en grande con el nombre de la tabla)
6. NOFORANEAS (numero de llaves foráneas que incluye la tabla)

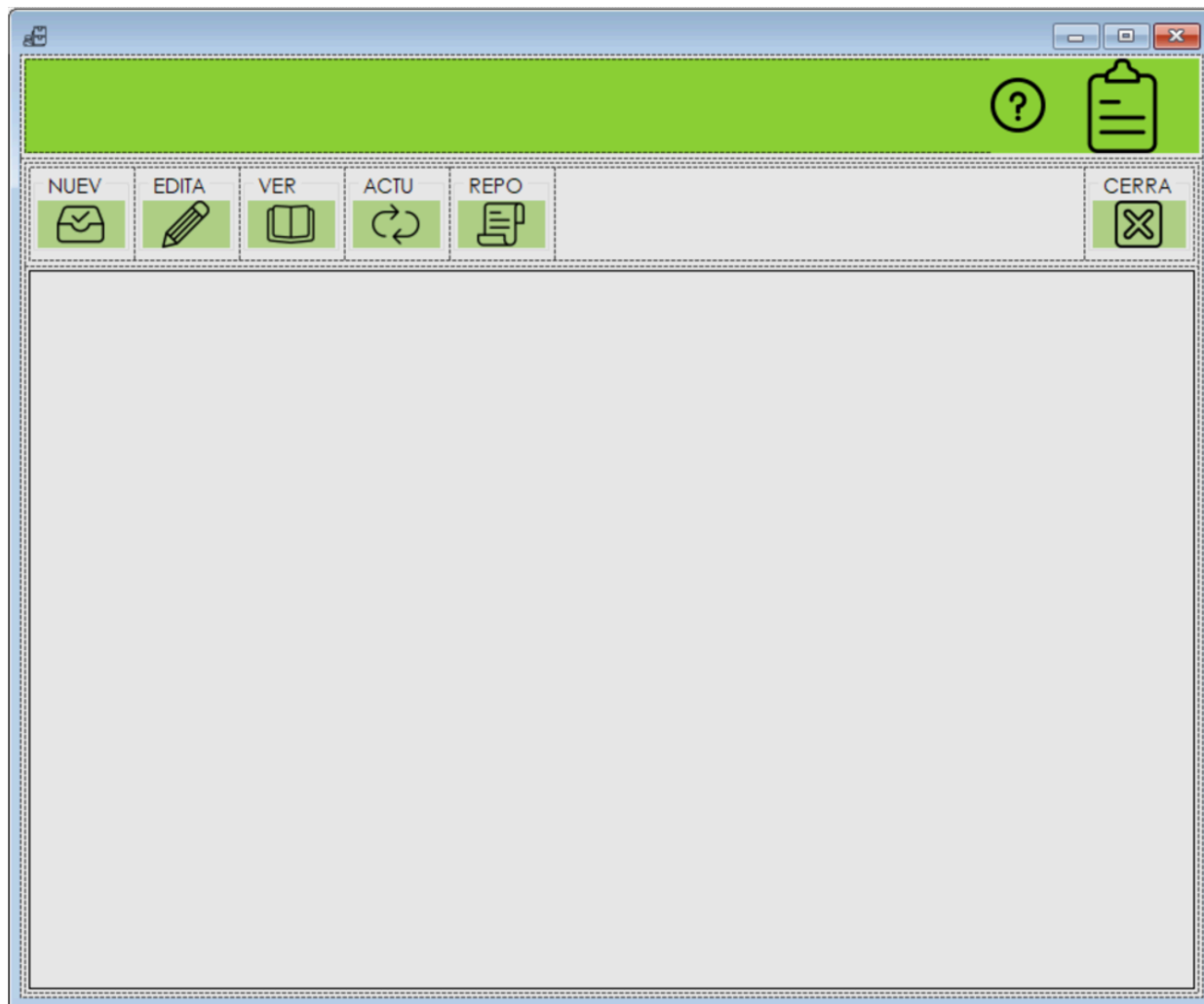
```
string[] alias1 = { "Codigo", "Nombre", "Descripcion", "Telefono", "Direccion", "Email", "Estado" };  
return (alias1, "1", "contactos", "de Contactos", "CONTACTO", 0);
```

En la segunda función se establecen los campos para las llaves foráneas, estas se obtienen dando como parámetro el id de la tabla y el numero de la llave foránea. El numero de la llave foránea es respecto al orden en que aparece en la tabla.

```
//inventarios  
case 15:  
    switch (no)  
    {  
        case 1:  
            return ("productos", "nombre_producto", 1);  
        case 2:  
            return ("bodegas", "nombre_bodega", 1);  
        default:  
            mensaje = new Mensaje("Error al identificar el mantenimiento a trabajar.");  
            mensaje.Show();  
            break;  
    }  
    break;
```

## PROCESOS

### Lista



The screenshot shows a software window titled 'Lista'. The window has a standard Windows-style title bar with minimize, maximize, and close buttons. Below the title bar is a green header bar containing a question mark icon and a clipboard icon. Underneath the header is a toolbar with six buttons: 'NUEV' (New) with an envelope icon, 'EDITA' (Edit) with a pencil icon, 'VER' (View) with a book icon, 'ACTU' (Update) with a circular arrow icon, 'REPO' (Repository) with a document icon, and 'CERRA' (Close) with a close icon. The main area of the window is a large, empty light gray rectangle, representing the list of inventory movements and purchase orders.

La lista es el form ubicado en la capa Vista donde se muestran todos los movimientos de inventario y ordenes de compra. Al igual que en los mantenimientos se uso un mismo form para las listas y se controla el uso de este según parámetros dentro de una clase en la capa Modelo. Llena los datos por medio de un objeto llamado **ListaData** el cual incluye el título para la lista, el título del form y la ayuda asignada.

```
public class ListaData : IEquatable<ListaData>
{
    public string titulo { get; set; }

    public string form { get; set; }

    public string ayuda { get; set; }
    public bool Equals(ListaData other)
    {
        throw new NotImplementedException();
    }
}
```

Para llenar el form de lista se requiere utilizar la clase **Cls\_listas.cs** utilizando la función **DatosLista** para poder definir los datos de la lista según el tipo de lista y se pide también el DataGridView en el que se mostrara la lista de datos. Esta función devuelve un objeto de tipo ListaData y llena el DataGridView.

```
public ListaData DatosLista(int tipoLista, DataGridView dgv)
```

#### Movimientos

Para los movimientos de inventario se posee un form el cual tiene 3 modos: **Ver**, **Editar** y **Nuevo**. Para el formulario Ver solo se muestran los datos sin opción a editar. En Editar se permite agregar o quitar productos, editar el nombre y descripción, la fecha del movimiento y el estado. En Nuevo permite crear un nuevo movimiento con todos los campos editables y con el datagrid vacío. Todo esto se establece en la función dentro del form llamada **establecerDatos()** establecido según el modo del form (este se identifica con la variable **modo** dentro de la función inicial del form), dato que viene desde el form de Lista.



```

switch (modo)
{
    case 1: // ingreso de nuevo registro
        llenarCombos();
        estadoCodigo();

        break;

    case 2: // Vista de Registro

        Grp_guardar.Visible = false;
        Grp_guardar.Enabled = false;
        Grp_cancelar.Visible = false;
        Grp_cancelar.Enabled = false;
        Chk_codigo.Visible = false;
        Chk_codigo.Enabled = false;
        Grp_producto.Visible = false;
        Grp_producto.Enabled = false;
        Pnl_datos.Enabled = false;
        Txt_descripcion.Enabled = false;
        Chk_estado.Enabled = false;
        Cbo_tipoMovimiento.Enabled = false;

        llenarEncabezado();
        llenarTotales();
        break;

    case 3: // Edicion de Registro

        Chk_codigo.Visible = false;
        Chk_codigo.Enabled = false;
        Txt_codigo.Enabled = false;
        Cbo_tipoMovimiento.Enabled = false;

        llenarCombos();

        llenarEncabezado();
        llenarTotales();
        cambioEnc = 0;
        cambioDet = 0;
        break;

    case 4:// Edicion de Registro recién ingresado

        Chk_codigo.Enabled = false;
        Chk_codigo.Visible = false;
        Txt_codigo.Enabled = false;
        Cbo_tipoMovimiento.Enabled = false;

        cambioEnc = 0;
        cambioDet = 0;

        modo = 3;
        break;
}

public Frm_MovimientosInventarios(Form form, int modo, int encabezado)
{
    InitializeComponent();

    //inicializacion de variables globales y campos segun el modo del form:
    Text = "1002 - " + Text; //Se le agrega el codigo de la aplicacion al form
    //se define el formato del dateTimePicker
    Dtp_fecha.Format = DateTimePickerFormat.Custom;
    Dtp_fecha.CustomFormat = "dd MM yyyy";
    //se inicializa la variable para el form de lista
    this.form = form;
    //se crea un string[] pra almacenar los datos del encabezado
    string[] datos;
    //se inicializa la variable global del modo del form
    this.modo = modo;
    //se indica que no se ha realizado ningun cambio de informacion
    cambioDet = 0;
    //se establece el tipo de ingreso de codigo como codigo automatico
    Chk_codigo.Checked = true;
    //Se establece el mostrar el impuesto como falso-
    //Chk_iva.Checked = false;
    //Se inicializa el numero de encabezado para modos 2 y 3
    this.idEncabezado = encabezado;

    Txt_precioTotal.Text = "0";
    Txt_costoTotal.Text = "0";


    establecerDatos();
}

```



1002 - Movimientos de Inventario



## Movimientos de Inventario


? 



DATOS DE MOVIMIENTO DE INVENTARIO

Datos Nota

CODIGO  ☒ Código automatico NOMBRE   
 TIPO MOV.   
 FECHA  ☒ Activo

GUARDAR  CANCELAR 

PRODUCTOS  BUSCAR   
 PRODUCTO  PRECIO COSTO CANTIDAD

AGREGAR  ELIMINAR 

DETALLE DE MOVIMIENTO

Codigo	Producto	Nombre	Cantidad	Costo	Precio

PRESIO TOTAL 0 COSTO TOTAL 0

### Ordenes de Compra

Para la orden de compra se posee un form especial que funciona en base a 3 modos específicos: **Ver**, **Editar** y **Nuevo**. De manera el modo del form viene desde el form de lista. En el modo Ver solo permite la visualización de los datos. En el modo Editar solo permite cambiar el nombre y descripción, la fecha de entrega estimada, fecha de emisión, es estado, si ya ha sido entregado y agregar o eliminar productos. En el modo Nuevo todos los campos quedan habilitados para poder ingresar una nueva orden de compra.

```
//nueva ordent de compra
case 1:
    tipoCodigo();

    llenarCombos();

    entregado = 0;

    cambioEnc = 0;
    break;

//ver una orden de compra
case 2:
    Grp_guardar.Enabled = false;
    Grp_guardar.Visible = false;
    Grp_cancelar.Enabled = false;
    Grp_cancelar.Visible = false;
    Grp_agregar.Enabled = false;
    Grp_agregar.Visible = false;
    Grp_producto.Enabled = false;
    Grp_producto.Visible = false;
    Txt_codigo.Enabled = false;
    Txt_nombre.Enabled = false;
    Txt_descripcion.Enabled = false;
    Dtp_emision.Enabled = false;
    Dtp_entrega.Enabled = false;
    Cbo_proveedor.Enabled = false;
    Dgv_ordenCompraDetalle.Enabled = false;
    Grp_BuscarProv.Visible = false;
    Grp_BuscarProv.Enabled = false;
    Dtp_emision.Enabled = false;
    Dtp_entrega.Enabled = false;

    llenarEncabezado();
    break;

//editar una orden de compra
case 3:
    Chk_codigo.Enabled = false;
    Chk_codigo.Visible = false;

    llenarCombos();

    Grp_BuscarProv.Visible = false;
    Grp_BuscarProv.Enabled = false;
    Cbo_proveedor.Visible = false;
    Cbo_proveedor.Enabled = false;
    Dtp_emision.Enabled = false;
    Dtp_entrega.Enabled = false;

    llenarEncabezado();
    cambioEnc = 0;
    entregado = 0;
    break;

public Frm_OrdenCompra(Form form, int modo, int encabezado)
{
    InitializeComponent();

    //inicializacion de variables globales y campos segun el modo del form:
    Text = "1003 - " + Text; //Se le agrega el codigo de la aplicacion al form
    //se define el formato del dateTimePicker
    DateTimePicker Dtp = new DateTimePicker();
    Dtp.Format = DateTimePickerFormat.Custom;
    Dtp.CustomFormat = "dd MM yyyy";
    Dtp_entrega = Dtp;
    Dtp_emision = Dtp;

    //se inicializa la variable para el form de lista
    this.form = form;
    //se inicializa la variable global del modo del form
    this.modo = modo;
    //Se inicializa el numero de encabezado para modos 2 y 3
    codigoSig = encabezado;
    //se indica que no se ha realizado ningun cambio de informacion
    cambioDet = 0;
    //se establece el tipo de ingreso de codigo como codigo automatico
    Chk_codigo.Checked = true;

    idEncabezado = encabezado;

    Chk_codigo.Checked = true;

    Chk_estado.Checked = true;

    Txt_precioTotal.Text = "0";
```

1003 - Orden de Compra

## ORDEN DE COMPRA

DATOS DE ORDEN DE COMPRA

Datos Nota

CODIGO  ☒ Codigo automatico NOMBRE

PROVEEDOR  BUSCAR PROV

FECHA ENTREGA  FECHA EMISION

☐ Entregado ☒ Activo

PRODUCTOS

PRODUCTO  BUSCAR

CANTIDAD

DETALLE DE ORDEN DE COMPRA

No.	Producto	Nombre	Cantidad	Precio

PRECIO TOTAL 0

## REFERENCIAS

[Carpetas](#)

[Repositorio](#)

## AUTORIA

AUTORIA	
CREADO POR	LIGIA ABRIL
CREADO PARA	GRUPO BIENESTAR
CREADO EN	MARZO-MAYO 2020