



# MANUAL TÉCNICO

---

## MRP

---

**MODULO DE PRODUCCIÓN**  
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 DEDICADOS</b>	<b>2</b>
Procesos y Recetas	2
Inventarios	4
Unidad de medida	5
<b>PROCESOS</b>	<b>8</b>
Procesos al iniciar el modulo	8
MDI_MRP_Load();	9
Función cu();	9
Función productos ();	10
Función suministros ();	10
Pedidos Especiales	11
Ordenes Pendientes	13
Productos en Proceso	15
Estadística Por Fecha	18
<b>REFERENCIAS</b>	<b>19</b>
Carpetas	19
Repositorio	19

## **INTRODUCCION**

El módulo MRP tiene a cargo todos los procedimientos explícitos e implícitos de producción, los cuales serán descritos en el presente manual de manera breve.

## **DIAGRAMACIÓN**

DFD

<https://drive.google.com/open?id=1cs1A8n-BhWC5D9PRaGuNCcC3-fZJ9CqR>

ER

<https://drive.google.com/open?id=1St2eG14o3sO4Qh3Auqyem3cf9uDMbD41>

CASO DE USO

[https://drive.google.com/open?id=1YyW\\_cK92WG9EaYG5CdIs0GF4\\_STdKuem](https://drive.google.com/open?id=1YyW_cK92WG9EaYG5CdIs0GF4_STdKuem)

## **MANTENIMIENTOS DEDICADOS**

Procesos y Recetas

Un mantenimiento importante y principal la manipulación de la información de procesos y recetas de producción donde se establece la unidad de medida y el procedimiento específico de un producto.

4002.1 - Mantenimiento Proceso

### Proceso de Produccion

Proceso

no\_proceso

Producto

Proceso Para

tiempo por producto "min"

Descripción

Guardar

Eliminar

Modificar

Crear Receta

	cod_proceso	cod_producto	nombre_producto	descripcion	tiempo	estado
	2	11	TetraColor Tropi...	Harina de pesca...	0.8	1
	4	1	Dog Show	Concentrado	0.5	1
	5	13	Bolsa de frijol ro...	Bolsa de 40 kg d...	5	1
	6	14	Bolsa de maiz p...	Cada bolsa deb...	8	1
	10	20	ejemplo	ejemplo	6	1

```


1 referencia
private void Btn_Receta_Click(object sender, EventArgs e)
{
    if (dgb_produccionProceso.SelectedRows.Count == 1)
    {
        string where = Txt_idproceso.Text;
        string sql = "Select nombre from produccion_procesos where estado =1 and id_proceso = "+ where + " ";
        string razon = mo.consulta(sql);

        mantenimientoRecetas frm = new mantenimientoRecetas(where, razon); //pasarle el id del encabezado que se esta consultando
        frm.MdiParent = This.MdiParent;
        frm.Show(); }
    }
}

```

La Función principal de estos botones es la típica inserción, eliminación y modificación de datos, pero un botón a resaltar es el botón de crear receta, el cual abre el formulario correspondiente al numero de proceso que fue seleccionado.

4002.1 - Recetas



### Receta

**Receta**  
 No. Producto ...  
 No. proceso 6  
 Receta Maiz porva  
 Producto   
 Unidad de Medida   
 Cantidad Necesaria 1.0

Guardar Receta  
 Modificar  
 Agregar  
 Eliminar

Rendimiento Restante  
 Cantidad fija 0.00  
 Rendimiento 0.00

	Id detalle	No. proceso	Producto	Razon	Cantidad	rendimient R	rendimient F	Cantidad Rendimien	Costo Unitario	Unidad de Medida
▶	20	6	Semilla ...	Maiz por...	15	85.5	10	100	2.5	kg

## Inventarios

Los inventarios son un área importante del sistema, en el se manipulan las existencias de manera automática, podemos insertar 1 sola vez un producto y luego las modificaciones a las cantidades serán automáticas según la receta de cada producto y su unidad de medida.

4004 - Mantenimiento Inventario

### Mantenimiento de Inventario

Proceso

No\_Inventario: 15      Cantidad en inventario: 606.00

Producto: Bolsa Vitamina E 900 g      Unidad de medida: ml

configuración: 4-primavera

	cod_inventario	Nombre_Producto	Temporada	Cantidad	Unidad_de_Medida	Estado
	22	SELENIO ORGANI...	verano	662.5	gr	1
	24	Beta Glucano	verano	493	ml	1
	25	Frijol Regional	invierno	900	kg	1
▶	15	Bolsa Vitamina E ...	primavera	606	ml	1
	18	Fucoidan	primavera	767	ml	1

Guardar

Eliminar

Modificar

Inicialmente se ejecutan las funciones para cargar de datos los combo box principales.

```
InitializeComponent();
asignarAlias(alias);
actualizadatsgriew();

Txt_idinv.Enabled = false;
Cmb_prod.llenarse2("productos","id_producto","nombre_producto","id_tipo_producto","2");
cmb_config.llenarse("configuraciones_inventarios","cod_config","temporada");
//cmb_unidadmedida.llenarse("unidades_medida","id_medida","unidad");
Txt_idinv.Text = mo.idmax("inventarios_produccion","id_inventario");

Btn_actualizar.Enabled = false;
Btn_eliminar.Enabled = false;

llenarse("unidades_medida","unidad1");
```

### Unidad de medida

La unidad de medida es un valor importante ya que corresponde a la conversión de las medidas de las recetas y el descuento a los inventarios según sea la conversión realizada.

4005 - Mantenimiento Unidad de Medida

### Mantenimiento de Unidad de Medida

Unidad de medida 1

Cantidad

1


Tiene

Unidad de medida 2


Cantidad

0.0000

Guardar



Eliminar



	Id	Cantidad 1	Unidad 1	Cantidad 2	Unidad 2
▶	20	1	lb	16	oz
	21	1	oz	0.0625	lb
	22	1	lb	1	lb
	23	1	oz	1	oz
	24	1	kg	1000	gr

Al ingresar una unidad de medida se ejecuta un proceso automático que ingresa el valor contrario de esa conversión, es decir, si se ingresa LB a KG, se convierte también de KG a LB, LB a LB y de KG a KG.

```

if (um1 != "" && um2 != "" && valr != 0)
{
    string sql = "Select id_medida from unidades_medida where unidad1 = '" + um1 + "' and unidad2 = '" + um2 + "' and estado = 1 ";
    string existe = mo.consulta(sql);
    if (existe == "")
    {
        string sql2 = "INSERT INTO unidades_medida (unidad1, cantidad1, unidad2, cantidad2) VALUES ('" + um1 + "', '1', '" + um2 + "', '" + valr + "') ";
        mo.insertar(sql2);
        unidadesnew(um2, valr, um1);
        MessageBox.Show("Registro Almacenado ", "Alerta", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);
        Txt_um1.Text = "";
        Txt_um2.Text = "";
        Nud_cant.Value = 0;
        actualizardatagridview(sql3);
    }
    else
    {

```



```

Inferencia
void unidadesnew(string un2, double cant2, string un1)
{
    double resultado = 1 / cant2;
    double resultador = Math.Round(resultado, 4);
    string sql = " INSERT INTO unidades_medida (unidad1, cantidad1, unidad2 , cantidad2) VALUES ('" + un2 + "' , " + 1 + " , '" + un1 + "' , " + resultado + " ) ";
    mo.inserter(sql);

    string consultar5 = "Select id_medida from unidades_medida where unidad1 = '" + un1 + "' and unidad2 = '" + un1 + "' and estado = 1 ";
    string ex = mo.consulta(consultar5);
    if (ex == " ")
    {
        string sql6 = " INSERT INTO unidades_medida (unidad1, cantidad1, unidad2 , cantidad2) VALUES ('" + un1 + "' , " + 1 + " , '" + un1 + "' , " + 1 + " ) ";
        mo.inserter(sql6);
    }
    string consultar7 = "Select id_medida from unidades_medida where unidad1 = '" + un2 + "' and unidad2 = '" + un2 + "' and estado = 1 ";
    string ex2 = mo.consulta(consultar7);
    if (ex2 == " ")
    {
        string sql7 = " INSERT INTO unidades_medida (unidad1, cantidad1, unidad2 , cantidad2) VALUES ('" + un2 + "' , " + 1 + " , '" + un2 + "' , " + 1 + " ) ";
        mo.inserter(sql7);
    }
}
}

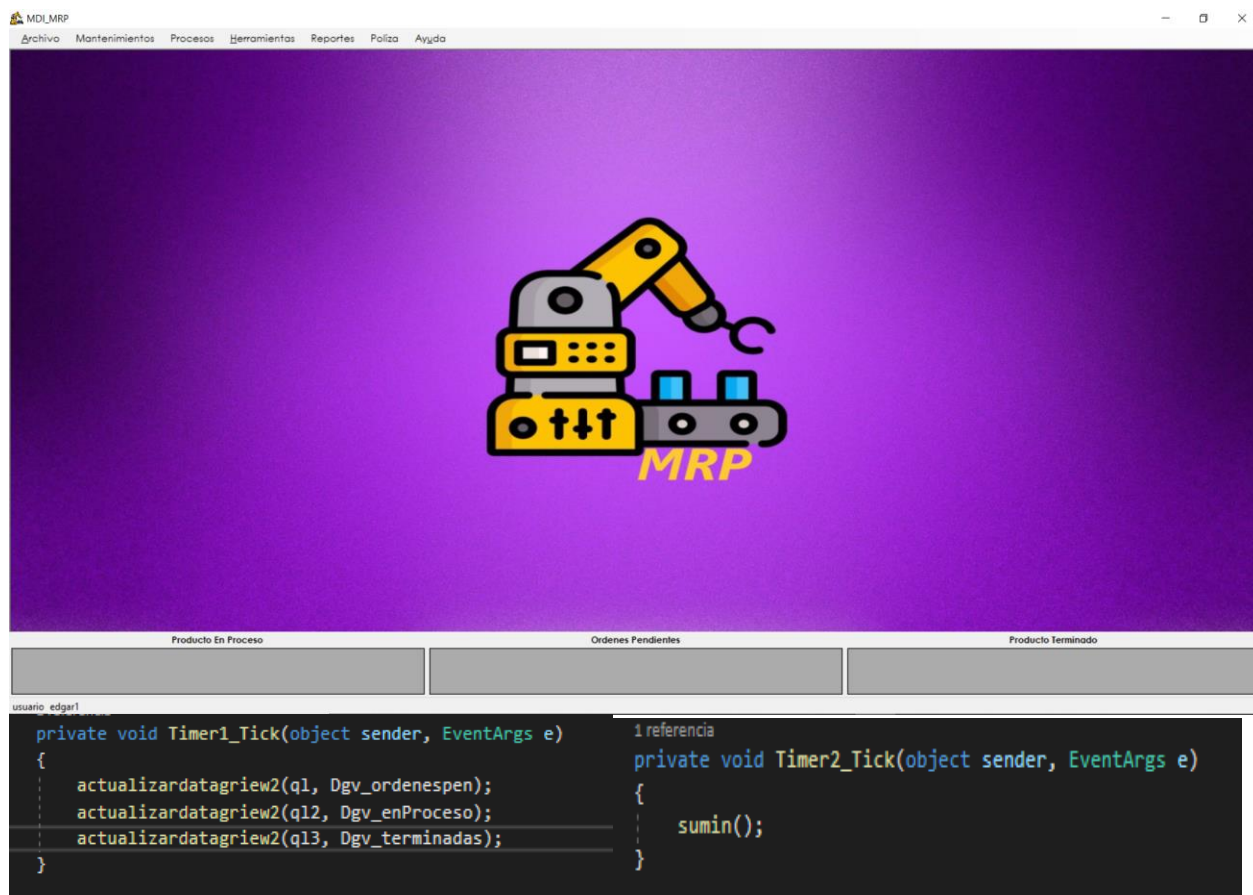
```



## PROCESOS

### Procesos al iniciar el modulo

Como procesos iniciales al cargar el modulo se ejecutarán 2 timers uno de 15 segundos encargado de recargar el menú inferior y el segundo de 1 minuto encargado de verificar la necesidad de suministros en el inventario.



También se ejecutan funciones encargadas de llenar tablas de la base de datos relacionada a los productos, costos y suministros por primera vez.

Producto En Proceso			Ordenes Pendientes			Producto Terminado	
codigo orden	Fecha Inicio	Fecha Limite	cod_pendiente	codigo orden	Fecha Limite	cod_orden	costo_total
1	25/5/2020	31/5/2020				4	539.798

MDI\_MRP\_Load();

```
1 referencia
private void MDI_MRP_Load(object sender, EventArgs e)
{
    frm_login login = new frm_login();
    login.ShowDialog();

    lblUsuario.Text = login.obtenerNombreUsuario();
    usuarioact = lblUsuario.Text;
    productos();
    cu();
    suministros();
    timer1.Start();
    timer2.Start();
}
```

Función cu();

```
public void cu()
{
    string sql = "select dr.id_detalle, (p.precio_producto/ dr.rendimiento_fijo) as 'costo_unitario' from productos p";
    DataTable dt = mo.consultaLogica2(sql);
    int i = 0;

    if (dt.Rows.Count > 0)
    {
        foreach (DataRow row in dt.Rows)
        {
            var id = dt.Rows[i]["id_detalle"].ToString();
            var costo = dt.Rows[i]["costo_unitario"].ToString();

            mo.updateestados("detalles_recetas", "costo_unitario", costo, "id_detalle", id);

            i++;
        }
    }
}
```

### Función productos ();

```
public void productos()
{
    string sql = "SELECT id_producto FROM productos p where p.id_tipo_producto = 1 and p.estado = 1 ";
    DataTable dt = mo.consultaLogica2(sql);
    int i = 0;

    string fecha1 = DateTime.Now.ToString("yyyy-MM-dd");

    if (dt.Rows.Count > 0)
    {
        foreach (DataRow row in dt.Rows)
        {
            var id = dt.Rows[i]["id_producto"].ToString();

            string sql2 = "SELECT id_producto FROM costos_produccion where id_producto = " + id + " and estado =1;";

            string resultado = mo.consulta(sql2);

            if (resultado == " ")
            {
                string insertar = "INSERT INTO costos_produccion (id_producto,fecha) values (" + id + "," + fecha1 + ")";
                mo.insertar(insertar);
            }

            i++;
        }
    }
}
```

### Función suministros ();

```
public void suministros()
{
    DataTable dt = mo.inventarios();

    if (dt.Rows.Count != 0)
    {
        int i = 0;

        string idmax = mo.getIdmax("solicitudes_encabezados", "cod_solicitud");

        foreach (DataRow row in dt.Rows)
        {
            string fecha = DateTime.Now.ToString("yyyy-MM-dd");
            var dato = dt.Rows[i]["id_producto"].ToString();

            var maximo = dt.Rows[i]["maximo"].ToString();
            var cantidad = dt.Rows[i]["cantidad_total"].ToString();

            string existe = mo.ObtenerSimple3("solicitudes_detalle", "id_producto", "id_producto", dato);

            if (existe == "")
            {
                double max = Convert.ToDouble(maximo);
                double cant = Convert.ToDouble(cantidad);

                double resultado = max - cant;

                string cadena = "INSERT INTO solicitudes_encabezados (cod_solicitud, fecha_solicitud , prioridad) VALUES(NULL, '" + fecha + "', 'Alta' )";
                mo.insertar(cadena);
                string detalle = "INSERT INTO solicitudes_detalle (cod_solicitud, id_producto, cantidad ) VALUES (" + idmax + " , '" + dato + "', '" + resultado + "')";
                mo.insertar(detalle);
            }

            i++;
        }
    }
}
```

## Pedidos Especiales

Las Ordenes de producción especiales corresponden a una orden proveniente de el modulo de CRM donde se ingresan los datos de la orden a producir.

2101 - Pedidos Especiales

## Pedidos Especiales

Encabezado

No. Orden

5

Generar Orden

Cliente

Cliente1

Fecha

2020-05-25

Fecha Limite

2020-06-01

Detalle

Producto

Agregar

Eliminar

Enviar Orden

Cantidad

1

Producto	Cantidad	Fecha	Fecha Limite

El procedimiento de asignación de fecha se valido para dar un lapsus de tiempo correspondiente a 7 días, sin embargo, posteriormente la orden se puede producir antes de la fecha.

**Ciente1**

**2020-05-25**

**2020-06-01**

junio de 2020						
lun.	mar.	mié.	jue.	vie.	sáb.	dom.
	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	1	2	3	4	5

Hoy: 25/5/2020

**Generar Orden**

**Eliminar**

**Enviar Orden**

El procedimiento de búsqueda en los combos es variante entre id del producto o nombre de este.

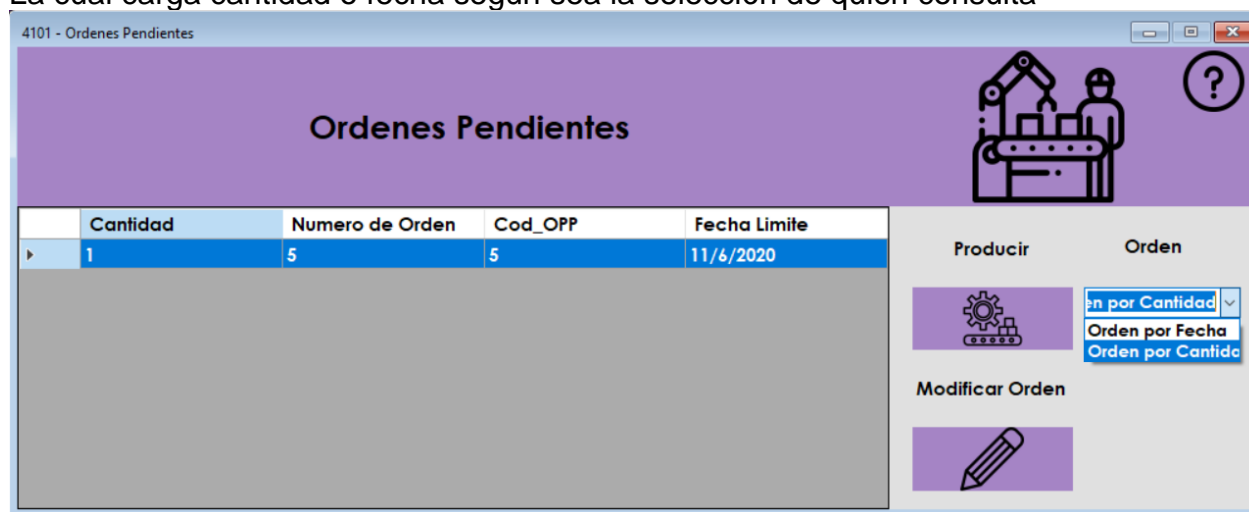
Agregar		Eliminar		Enviar Orden	
Producto	camaronina-6				
Cantidad					

Cuando tenemos la orden generada, podemos recorrer la tabla con una función específica y se ingresaran los detalles correspondientes a la orden en cuestión.

```
private void Button1_Click_1(object sender, EventArgs e)
{
    string idmax = lbl_noOrden.Text;
    if (Dgb_pedidosEspeciales.Rows.Count > 0)
    {
        foreach (DataGridViewRow row in Dgb_pedidosEspeciales.Rows)
        {
            string cadena = "INSERT INTO produccion_detalle (id_detalle, cod_orden, id_producto, cantidad_producto, estado) VALUES(NULL," + idmax + ",";
            mo.insertar(cadena);
        }
        MessageBox.Show("Orden Generada");
        this.Close();
    }
    else {
        MessageBox.Show("Agregue los productos a la orden ", "Alerta", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);
    }
}
```

### Ordenes Pendientes

Para verificar la cantidad de Ordenes pendientes se muestra la siguiente ventana. La cual carga cantidad o fecha según sea la selección de quien consulta



La manera de ordenar el ordenamiento de la tabla es la siguiente  
 Fecha o cantidad

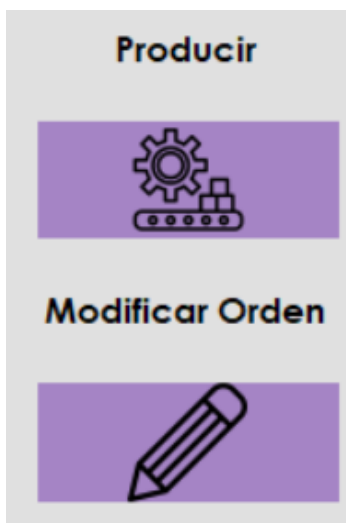


Las funciones correspondientes al ordenamiento seleccionado son las siguientes.

```
private void Cmb_Orden_SelectedIndexChanged(object sender, EventArgs e)
{
    if (Cmb_Orden.Text == "Orden por Fecha")
    {
        limpiarddb();
        asignarAlias(alias);
        actualizardatagridview();

        if (Dgb_ordenesPendientes.RowCount > 0)
        {
            Btn_producir.Enabled = true;
            Btn_cancelar.Enabled = true;
        }
    }
    if (Cmb_Orden.Text == "Orden por Cantidad")
    {
        limpiarddb();
        asignarAlias2(alias2);
        actualizarfatagridorden();
        if (Dgb_ordenesPendientes.RowCount > 0)
        {
            Btn_producir.Enabled = true;
            Btn_cancelar.Enabled = true;
        }
    }
}
}
```

Encargadas de actualizar la tabla según la opción elegida



El botón producir desencadena una serie de funciones dedicadas a lo siguiente:

1. Verificar suministros
2. Productos ingresados a costos
3. Costo unitario
4. Costo mano de obra
5. Costo materia prima
6. Costo CIF



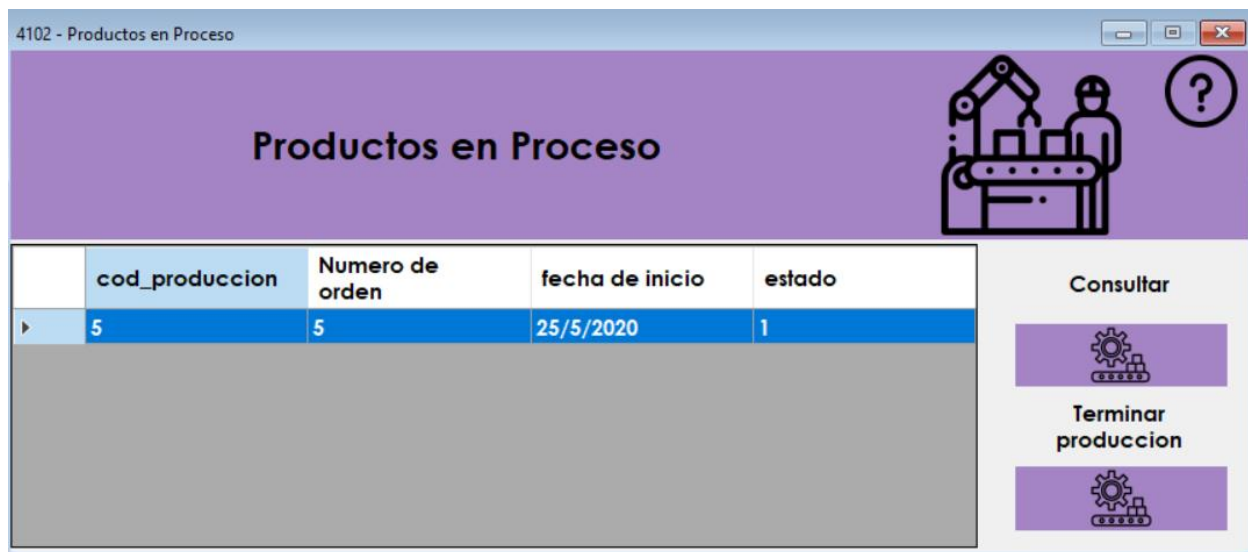
## 7. Costo total de producción

```
suministros();  
actualizadatabgrid();  
productos();  
cu();  
cutmo();  
cutmp();  
cif();  
ctp();
```

### Productos en Proceso

Dentro del Formulario de producto en proceso podemos verificar el producto pendiente pero ya en línea de producción, sin embargo, las ordenes tienen un lapso para terminarse y actualizarse hasta una fecha determinada, función que se ejecuta al abrir este formulario únicamente si hay filas u ordenes que actualizar

```
public productosProceso()  
{  
    InitializeComponent();  
  
    assignerAlias(alias);  
    actualizadatabgrid();  
  
    if (dgb_productosProceso.Rows.Count > 1) {  
        actualizarod();  
        cord();  
    }  
}
```



Podremos consultar la orden o verificar ordenes que aun sigan en la lista pero que ya estén terminadas y terminarlas manualmente.



En el botón consultar podremos ver el encabezado y el detalle de la orden seleccionada  
 Y el timer activo actualizará las ordenes terminadas.

4102.1 Detalle Productos En Proceso

## Productos en Proceso



**Encabezado**

	Numero de Orden	fecha orden	fecha limite	tipo de produccion	Estado
▶	5	25/5/2020	11/6/2020	Especial	0

**Detalle**

	Nombre de Producto	Cantidad Total	Numero de detalle	Numero de Orden	Estado
▶	TetraColor Tr...	1	10	5	1

Actualización en menú inferior

Producto terminado

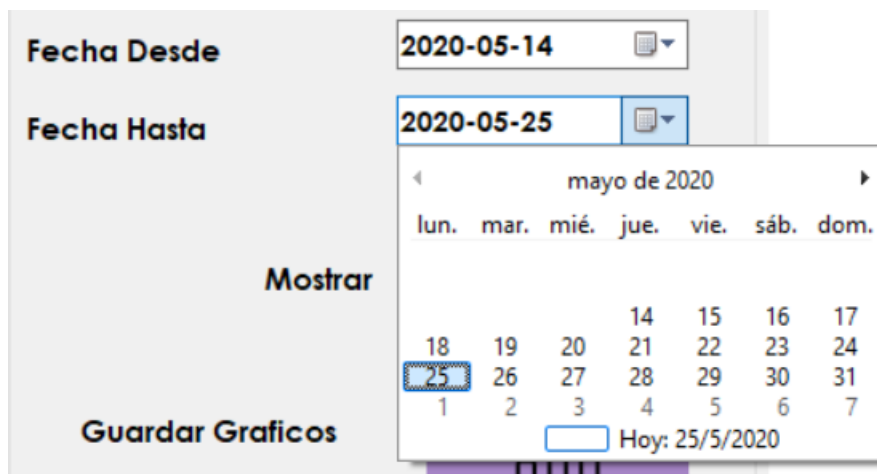
	cod_orden	costo_total
▶	5	115.958

### Estadística Por Fecha

El Formulario Muestra las estadísticas del nivel de producción mas alto y los costos de cada producto en un rango de fecha específico.



Las fechas son validadas para no permitir buscar en fechas equivocadas



The screenshot shows the date selection interface with 'Fecha Desde' set to 2020-05-14 and 'Fecha Hasta' set to 2020-05-25. A calendar for May 2020 is displayed, with the 25th highlighted. The calendar shows the days of the week (lun., mar., mié., jue., vie., sáb., dom.) and the dates (18 to 31). Below the calendar, there is a text box with 'Hoy: 25/5/2020'.

El programa también permite almacenar estas graficas para un análisis posterior, y la función correspondiente es la siguiente:

```
1 referencia
private void Btn_imagen_Click(object sender, EventArgs e)
{
    Process proceso = new Process();
    string fecha1 = DateTime.Now.ToString("yyyy-MM-dd");

    string nombre = InputDialog.mostrar("Nombre con el que desea guardar ");

    if (nombre != "")
    {
        chartprod1.SaveImage("graficos/bars"+nombre+"-"+fecha1+".png", ChartImageFormat.Png);
        chartprod.SaveImage("graficos/pie"+nombre+"-"+fecha1+".png", ChartImageFormat.Png);
        MessageBox.Show("Imagen almacenada con exito", "OK", MessageBoxButtons.OK, MessageBoxIcon.Information);

        proceso.StartInfo.FileName = "C:/Users/edgar/Documents/GitHub/MRP-Edgar/MRP/MDI_MRP/MDI_MRP/bin/Debug/graficos/pie"+nombre+"-"+fecha1+".png";
        proceso.Start();
        proceso.StartInfo.FileName = "C:/Users/edgar/Documents/GitHub/MRP-Edgar/MRP/MDI_MRP/MDI_MRP/bin/Debug/graficos/bars"+nombre+"-"+fecha1+".png";
        proceso.Start();
    }
    else {
        MessageBox.Show("Debe escribir un nombre para almacenar su imagen", "ERROR", MessageBoxButtons.OK, MessageBoxIcon.Information);
    }
}
```

## REFERENCIAS

### Carpetas

[https://drive.google.com/open?id=1aazJ6hcEbPxnxbm\\_zQGzc5U9ZwaWFwbT](https://drive.google.com/open?id=1aazJ6hcEbPxnxbm_zQGzc5U9ZwaWFwbT)

### Repositorio

<https://github.com/Eddcas01/MRP-Edgar.git>

## AUTORIA

AUTORIA	
CREADO POR	Edgar Casasola
CREADO PARA	GRUPO BIENESTAR
CREADO EN	MARZO-MAYO 2020