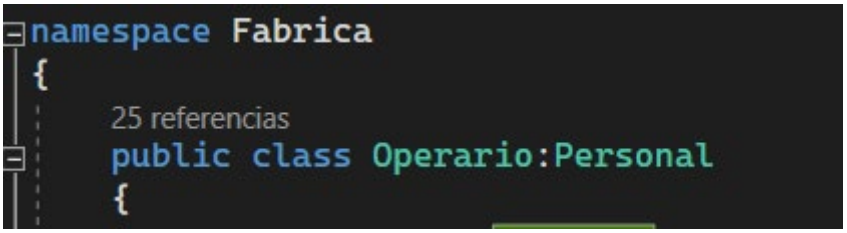
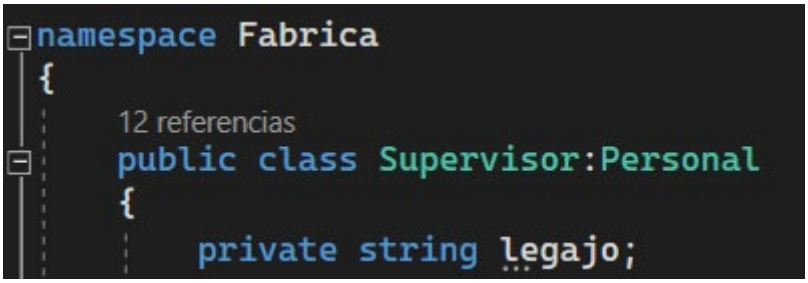
Uso de temas vistos en clase

*Voy a dar un ejemplo de casos en el código donde use los temas vistos y pedidos para el parcial*

En mi código no uso conversiones explicitas o implícitas y tampoco uso sobrecarga ya que no encontré ninguna funcionalidad para aplicarlo

1. Herencia: De una clase llamada “Personal” hay dos clases derivadas que se llaman “Supervisor” y “Operario”.




2. Polimorfismo: Uso polimorfismo en la clase “Personal” para que las clases derivadas puedan usar el método mostrar para cuando inicio sesión.

```
5 referencias
public abstract class Personal
{
    private string usuario;
    private string password;
    private Rango rango;

    2 referencias
    public Personal(string usuario, string password, Rango rango)
    {
        this.usuario = usuario;
        this.password = password;
        this.rango = rango;
    }

    4 referencias
    public string Usuario { get => usuario; set => usuario = value; }

    2 referencias
    public string Password { get => password; set => password = value; }

    2 referencias
    public Rango Rango { get => rango; set => rango = value; }

    4 referencias
    public abstract string Mostrar();
}
```

```
2 referencias
public override string Mostrar()
{
    StringBuilder sb=new StringBuilder();
    sb.AppendLine($"Nombre: {Usuario}");
    sb.AppendLine($"Legajo: {legajo}");
    sb.AppendLine($"Rango: {Rango}");
    return sb.ToString();
}
```

3. Colecciones: Use listas y diccionarios, diccionarios use para la parte donde le asigno el stock inicial a los productos y listas use por ejemplo para agregar a los supervisores y operarios a la lista de los que pueden entrar al sistema.

```
public static Dictionary<string, int> Materiales = new Dictionary<string, int>()
{
    {"Amargo", 100 },
    {"Vegano", 100 },
    {"Con mani", 100 },
    {"Chocolate Blanco", 50 },
    {"2x2", 50 },
    {"6x6", 50 },
    {"Glaseadas", 100 },
    {"Cubiertas", 100 },
    {"Con chispas", 100 },
    {"Cereza", 50 },
    {"Dulce de leche", 50 }
};
```

```
private static List<Operario> operarios = new List<Operario>();

8 referencias
public static List<Operario> ListaOperarios { get => operarios; set => operarios = value; }
```

```
Operario operario1 = new Operario("Martin", "12345", Rango.Operario, "Chacal");
Operario operario2 = new Operario("Federico", "12345", Rango.Operario, "Galarza");
Operario operario3 = new Operario("Paola", "12345", Rango.Operario, "Argento");
Operario operario4 = new Operario("Lauti", "profe", Rango.Operario, "Profe");
Operario operario5 = new Operario("Lucas", "profe", Rango.Operario, "Profe");
Operario operario6 = new Operario("Mathi", "profe", Rango.Operario, "Bustamante");

Operario.ListaOperarios.Add(operario1);
Operario.ListaOperarios.Add(operario2);
Operario.ListaOperarios.Add(operario3);
Operario.ListaOperarios.Add(operario4);
Operario.ListaOperarios.Add(operario5);
Operario.ListaOperarios.Add(operario6);
```

4. Formulario modal: Use formularios modales para la parte de simulación de la creación de un producto, donde cada formulario modal dura 3 segundos y se van cambiando uno al otro hasta el proceso final.

```
2 referencias
private DialogResult MostrarFormularioModal()
{
    using (FormModal1 formProduct = new FormModal1())
    {
        Hide();
        var result = formProduct.ShowDialog();
        if (result == DialogResult.Cancel)
        {
            Show();
        }
        return result;
    }
}
```

```
13 referencias
public partial class FormModal1 : Form
{
    private System.Windows.Forms.Timer timer;
    5 referencias
    public FormModal1()
    {
        InitializeComponent();

        progressBar1.Maximum = 100;
        progressBar1.Value = progressBar1.Maximum;
        timer = new System.Windows.Forms.Timer();
        timer.Interval = 2000;
        timer.Tick += timer1_Tick;
        timer.Start();
    }

    2 referencias
    private void timer1_Tick(object sender, EventArgs e)
    {

        if (progressBar1.Value > 0)
        {
            progressBar1.Value--;
        }

        timer.Stop();

        FormModal2 form2 = new FormModal2();

        this.Hide();
        form2.ShowDialog();

        this.Close();
    }
```

5. Clases estáticas: Hago la clase “Produccion” estatica para que el stock sea consistente en toda la aplicación, sin importar desde dónde accedas o modifiques esos valores.

```
namespace Fabrica
{
    15 referencias
    public static class Produccion
    {
```

6. Enumeradores: Use enumeradores para elegir el rango de supervisor y operario

```
18 referencias
public enum Rango
{
    Operario,
    Supervisor
}
```

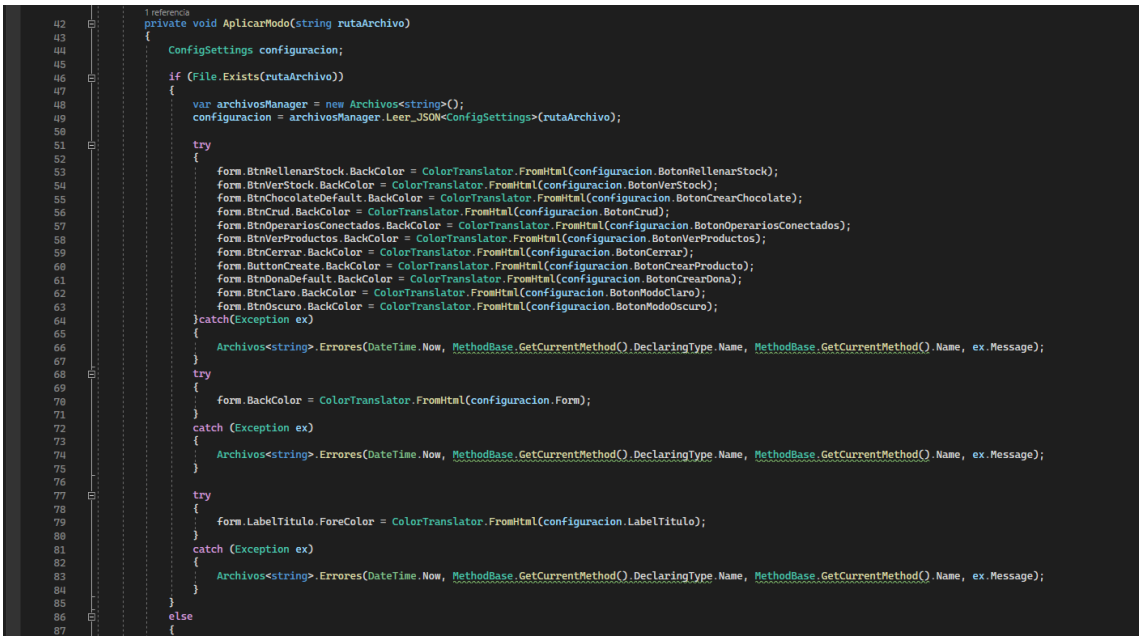
7. Propiedades: Use propiedades para las clases “Operario” y “Supervisor” para que puedan ser leídas y modificadas desde otros lugares en el código, pero al mismo tiempo tener la opción de controlar o limitar este acceso en el futuro si es necesario.

```
8 referencias
public static List<Operario> ListaOperarios { get => operarios; set => operarios = value; }
6 referencias
public static List<Supervisor> ListaSupervisor { get => supervisor; set => supervisor = value; }
```

8. Sobrecarga: Use sobrecarga para un método de eliminar, donde en uno lo uso para eliminar a un “Operario” de la base de datos y el otro lo sobrecargo para poder pasarle otro parámetro para eliminar al “Supervisor”

```
1 referencia
public static void Eliminar(int id)
{
    try
    {
        command.Parameters.Clear();
        connection.Open();
        command.CommandText = $"DELETE FROM OPERARIO WHERE ID = {id}";
        command.Parameters.AddWithValue("@ID", id);
        int rows = command.ExecuteNonQuery();
    }
    catch (Exception ex)
    {
        Archivos<string>.Errores(DateTime.Now, MethodBase.GetCurrentMethod().DeclaringType.Name, MethodBase.GetCurrentMethod().Name, ex.Message);
        throw;
    }
    finally
    {
        connection.Close();
    }
}
1 referencia
public static void Eliminar(int id, string labor)
{
    try
    {
        command.Parameters.Clear();
        connection.Open();
        command.CommandText = $"DELETE FROM {labor} WHERE ID = {id}";
        command.Parameters.AddWithValue("@ID", id);
        int rows = command.ExecuteNonQuery();
    }
    catch (Exception ex)
    {
        Archivos<string>.Errores(DateTime.Now, MethodBase.GetCurrentMethod().DeclaringType.Name, MethodBase.GetCurrentMethod().Name, ex.Message);

        throw;
    }
    finally
    {
        connection.Close();
    }
}
}
```

9. Excepciones: Uso excepciones en la mayor parte de mi código ya que de esta forma me aseguro tener controlado los errores que puedan a llegar a surgir en mi código. Un ejemplo es cuando lo uso para cuando se cambia algún color en el archivo de configuracion JSON para el modo oscuro.



```
1 referencia
private void AplicarModo(string rutaArchivo)
{
    ConfigSettings configuracion;

    if (File.Exists(rutaArchivo))
    {
        var archivosManager = new Archivos<string>();
        configuracion = archivosManager.Leer_JSON<ConfigSettings>(rutaArchivo);

        try
        {
            form.BtnRellenarStock.BackColor = ColorTranslator.FromHtml(configuracion.BotonRellenarStock);
            form.BtnVerStock.BackColor = ColorTranslator.FromHtml(configuracion.BotonVerStock);
            form.BtnChocolateDefault.BackColor = ColorTranslator.FromHtml(configuracion.BotonCrearChocolate);
            form.BtnCrud.BackColor = ColorTranslator.FromHtml(configuracion.BotonCrud);
            form.BtnOperariosConectados.BackColor = ColorTranslator.FromHtml(configuracion.BotonOperariosConectados);
            form.BtnVerProductos.BackColor = ColorTranslator.FromHtml(configuracion.BotonVerProductos);
            form.BtnCerrar.BackColor = ColorTranslator.FromHtml(configuracion.BotonCerrar);
            form.ButtonCreate.BackColor = ColorTranslator.FromHtml(configuracion.BotonCrearProducto);
            form.BtnDonaDefault.BackColor = ColorTranslator.FromHtml(configuracion.BotonCrearDona);
            form.BtnClaro.BackColor = ColorTranslator.FromHtml(configuracion.BotonModoClaro);
            form.BtnOscuro.BackColor = ColorTranslator.FromHtml(configuracion.BotonModoOscuro);
        }catch(Exception ex)
        {
            Archivos<string>.Errores(DateTime.Now, MethodBase.GetCurrentMethod().DeclaringType.Name, MethodBase.GetCurrentMethod().Name, ex.Message);
        }
        try
        {
            form.BackColor = ColorTranslator.FromHtml(configuracion.Form);
        }
        catch (Exception ex)
        {
            Archivos<string>.Errores(DateTime.Now, MethodBase.GetCurrentMethod().DeclaringType.Name, MethodBase.GetCurrentMethod().Name, ex.Message);
        }

        try
        {
            form.LabelTitulo.ForeColor = ColorTranslator.FromHtml(configuracion.LabelTitulo);
        }
        catch (Exception ex)
        {
            Archivos<string>.Errores(DateTime.Now, MethodBase.GetCurrentMethod().DeclaringType.Name, MethodBase.GetCurrentMethod().Name, ex.Message);
        }
    }
    else
    {
```

10. Archivos y serialización: Uso archivos para crear un JSON, para crear un XML y para guardar un archivo con los errores que se encuentran en las excepciones.

```
Archivos<string>.Errores(DateTime.Now, MethodBase.GetCurrentMethod().DeclaringType.Name, MethodBase.GetCurrentMethod().Name, ex.Message);
```

```
configuracion = new ConfigSettings
{
    BotonRellenarStock = "",
    BotonCrearChocolate = "",
    BotonVerStock = "",
    BotonCrud = "",
    BotonOperariosConectados = "",
    BotonVerProductos = "",
    BotonCerrar = "",
    BotonCrearProducto = "",
    BotonCrearDona = "",
    Form = "",
    LabelTitulo = "",
    BotonModoClaro = "",
    BotonModoOscuro = "",
};

string configuracionString = JsonSerializer.Serialize(configuracion);
var archivosManager = new Archivos<string>();
archivosManager.EscribirJson(rutaArchivo, configuracionString);



MessageBox.Show("Se creo el archivo de configuración.", "Archivo JSON", MessageBoxButtons.OK, MessageBoxIcon.Information);
```

```
if (datos != null && datos.Count > 0)
{
    var archivosManager = new Archivos<List<ComponenteCantidad>>();
    archivosManager.Escribir_XML(directorioPath, datos);
    MessageBox.Show("El stock actual fue guardado en un archivo XML", "Archivo guardado", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
}
else
{
    MessageBox.Show("La lista de datos está vacía o nula", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
}

atch (Exception ex)

Archivos<string>.Errores(DateTime.Now, MethodBase.GetCurrentMethod().DeclaringType.Name, MethodBase.GetCurrentMethod().Name, ex.Message);
```

11. Generics: Uso generics para mi clase de “Archivos” ya que los distintos métodos a la hora de usarlos les tengo que pasar un tipo distinto.

```
{
    36 referencias
    public class Archivos<T> : Iarchivos<T>
    {

        2 referencias
        public bool Escribir_TXT(string path, string datos)
        {
            try
            {
                using (StreamWriter sw = new StreamWriter(path, true))
                {
                    sw.WriteLine(datos);
                }
                return true;
            }
            catch (Exception ex)
            {

                Console.WriteLine(ex.Message);
                return false;
            }
        }

        2 referencias
        public void EscribirJson(string path, T objeto)
        {
            string directorio = Path.GetDirectoryName(path);

            if (!Directory.Exists(directorio))
            {
                Directory.CreateDirectory(directorio);
            }

            JsonSerializerOptions options = new JsonSerializerOptions();
            options.WriteIndented = true;

            string jsonString = JsonSerializer.Serialize(objeto, options);

            File.WriteAllText(path, jsonString);
        }
```

12. Interfaces: Uso interfaces para todos los métodos de archivos, ya que los uso en varias partes de mi código.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Fabrica
{
    1 referencia
    public interface Iarchivos<T>
    {
        2 referencias
        bool Escribir_TXT(string path, string datos);
        2 referencias
        void EscribirJson(string path, T objeto);
        2 referencias
        void Escribir_XML(string path, T objeto);
        1 referencia
        T LeerXml(string path);

        3 referencias
        T Leer_JSON<T>(string path);
    }
}
```

13. Test unitario: Uso test unitario para hacer distintas pruebas en métodos de mi código, por ejemplo para validar el ingreso de un operario.

```
namespace Pruebas
{
    [TestClass]
    0 referencias
    public class UnitTest2
    {
        [TestMethod]
        | 0 referencias
        public void TestBuscarPorUsuarioYRango_UsuarioYRangoValidos_DeberiaEncontrarOperario()
        {
            // Arrange
            var operarioEsperado = new Operario("ricardo", "hoy", Rango.Operario, "fort", 11);
            var listaPersonal = new List<Operario>
            {
                operarioEsperado,
            };

            // Act
            var resultado = Operario.BuscarPorUsuarioYRango("ricardo", "hoy", Rango.Operario, listaPersonal);

            // Assert
            Assert.AreEqual(operarioEsperado.Usuario, resultado.Usuario);
            Assert.AreEqual(operarioEsperado.Password, resultado.Password);
            Assert.AreEqual(operarioEsperado.Rango, resultado.Rango);
            Assert.AreEqual(operarioEsperado.Apellido, resultado.Apellido);
            Assert.AreEqual(operarioEsperado.Id, resultado.Id);

        }


    }
}
```

14. Base de datos: Uso base de datos en un montón de lugares, ya que es la base donde se alojan tanto los operarios como los supervisores, tengo una clase de base de datos y en la parte de la fabrica del supervisor tiene la opción de ejecutar consultas de tipo “Crud”

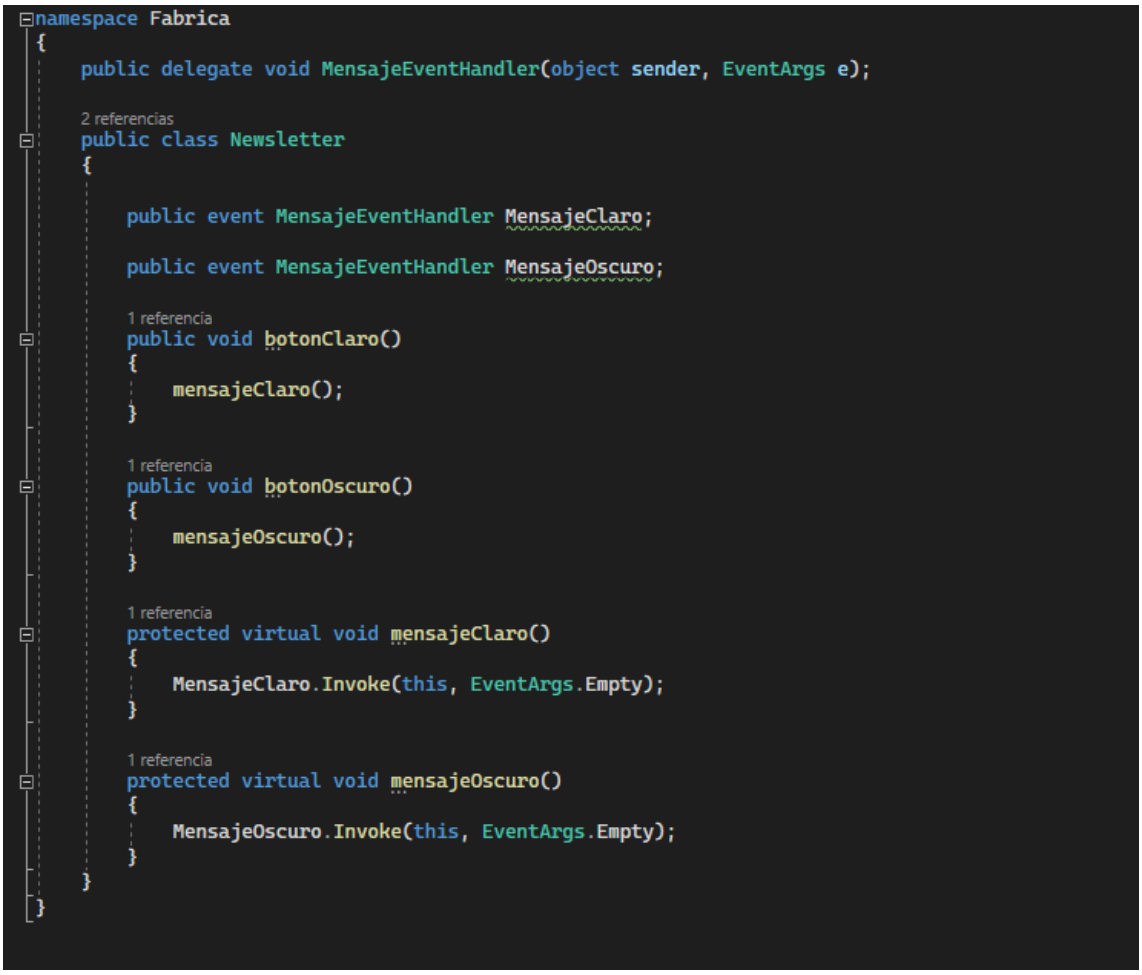
```
16 referencias
public class BaseDeDatosDAO
{
    static string connectionString;
    static SqlCommand command;
    static SqlConnection connection;
    const string NOMBRE_TABLA = "NOMBRE";
    const string APELLIDO_TABLA = "APELLIDO";
    const string PASSWORD_TABLA = "CONTRASEÑA";
    const string LEGAJO_TABLA = "LEGAJO";
    const string ID_TABLA = "ID";


    0 referencias
    static BaseDeDatosDAO()
    {
        connectionString = @"Data Source=.;Initial Catalog=MISTICAMOUSSE;Integrated Security=True";
        command = new SqlCommand();
        connection = new SqlConnection(connectionString);
        command.CommandType = System.Data.CommandType.Text;
        command.Connection = connection;
    }

    1 referencia
    public static void GuardarOperario(string nombre, string apellido, string password, string rango)
    {
        try
        {
            command.Parameters.Clear();
            connection.Open();
            command.CommandText = $"INSERT INTO OPERARIO (NOMBRE,APELLIDO,RANGO,CONTRASEÑA) VALUES (@NOMBRE,@APELLIDO,@RANGO,@CONTRASEÑA)";
            command.Parameters.AddWithValue("@NOMBRE", nombre);
            command.Parameters.AddWithValue("@APELLIDO", apellido);
            command.Parameters.AddWithValue("@RANGO", rango);
            command.Parameters.AddWithValue("@CONTRASEÑA", password);
            int rows = command.ExecuteNonQuery();
        }
        catch (Exception ex)
        {
            Archivos<string>.Errores(DateTime.Now, MethodBase.GetCurrentMethod().DeclaringType.Name, MethodBase.GetCurrentMethod().Name, ex.Message);
            throw;
        }
        finally { connection.Close(); }
    }
    1 referencia
    public static void GuardarSupervisor(string nombre, string password, string legajo, string rango)
    {
        try
        {
            command.Parameters.Clear();
```

```
1 referencia
private void btnEliminar_Click(object sender, EventArgs e)
{
    try
    {
        if (dgOperario.SelectedRows.Count > 0)
        {
            Operario operario = (Operario)dgOperario.CurrentRow.DataBoundItem;
            BaseDeDatosDAO.Eliminar(operario.Id);
            Refrescar();

            MessageBox.Show("Operario eliminado correctamente", "Operario Eliminado", MessageBoxButtons.OK, MessageBoxIcon.Information);
        }
    }catch (Exception ex)
    {
        Archivos<string>.Errores(DateTime.Now, MethodBase.GetCurrentMethod().DeclaringType.Name, MethodBase.GetCurrentMethod().Name, ex.Message);
    }
}

1 referencia
private void btnModificar_Click(object sender, EventArgs e)
{
    try
    {
        if (dgOperario.SelectedRows.Count > 0)
        {
            Operario operario = (Operario)dgOperario.CurrentRow.DataBoundItem;
            FormActualizarOperario formActualizarOperario = new FormActualizarOperario(operario.Id);
            formActualizarOperario.ShowDialog();
            Refrescar();
        }
    }
    catch(Exception ex)
    {
        Archivos<string>.Errores(DateTime.Now, MethodBase.GetCurrentMethod().DeclaringType.Name, MethodBase.GetCurrentMethod().Name, ex.Message);
    }
```

15. Delegados: Uso delegados para externalizar esa una lógica en especifica, además de utilizar delegados para compartir elementos entre formularios. Por ejemplo uso delegados en mi lógica de crear productos ya que de esta forma se simplifica mucho mi código.

```
private delegate bool VerificarStockDelegate(string tipo, string detalle);
private delegate bool ProductoOperationDelegate(string tipo, string detalle);


2 referencias
public FormCrearProducto()
{
    InitializeComponent();
}


2 referencias
private void CrearProducto(GroupBox tipoGroupBox, GroupBox detalleGroupBox, VerificarStockDelegate verificarStock, ProductoOperationDelegate agregarProductoAListaa)
{
    try
    {
        string tipo = ObtenerSeleccion(tipoGroupBox);
        string detalle = ObtenerSeleccion(detalleGroupBox);

        if (!verificarStock(tipo, detalle))
        {
            MessageBox.Show("No queda stock", "Advertencia", MessageBoxButtons.OK, MessageBoxIcon.Warning);
            return;
        }

        MostrarFormularioModal();

        agregarProductoAListaa(tipo, detalle);

        Produccion.Stock(tipo, detalle);
    }
    catch (Exception ex)
    {
        Archivos<string>.Errores(DateTime.Now, MethodBase.GetCurrentMethod().DeclaringType.Name, MethodBase.GetCurrentMethod().Name, ex.Message);
    }
}
```

16. Eventos: Uso eventos para poder manejar distintas situaciones en conjunto y para poder mostrar mensajes usando event handler. Por ejemplo uso eventos para poder a través de código hacer que el botón de modo oscuro y modo claro muestren un mensaje especificando lo que se hizo.



```
namespace Fabrica
{
    public delegate void MensajeEventHandler(object sender, EventArgs e);

    2 referencias
    public class Newsletter
    {

        public event MensajeEventHandler MensajeClaro;

        public event MensajeEventHandler MensajeOscuro;

        1 referencia
        public void botonClaro()
        {
            mensajeClaro();
        }

        1 referencia
        public void botonOscuro()
        {
            mensajeOscuro();
        }

        1 referencia
        protected virtual void mensajeClaro()
        {
            MensajeClaro.Invoke(this, EventArgs.Empty);
        }

        1 referencia
        protected virtual void mensajeOscuro()
        {
            MensajeOscuro.Invoke(this, EventArgs.Empty);
        }
    }
}
```

```
private void btnOscuro_Click(object sender, EventArgs e)
{
    tema = Color.Gray;
    cambiarColor(this);
    newsletter.botonOscuro();
}

1 referencia
private void btnClaro_Click(object sender, EventArgs e)
{
    tema = Color.AliceBlue;
    cambiarColor(this);
    newsletter.botonClaro();
}

1 referencia
private void MostrarMensajeClaro(object sender, EventArgs e)
{
    MessageBox.Show("Color oscuro aplicado", "Color Cambiado");
}

1 referencia
private void MostrarMensajeOscuro(object sender, EventArgs e)
{
    MessageBox.Show("Color oscuro aplicado", "Color Cambiado");
}

1 referencia
private void FormCrud_Load(object sender, EventArgs e)
{
    newsletter = new Newsletter();

    newsletter.MensajeClaro += MostrarMensajeClaro;
    newsletter.MensajeOscuro += MostrarMensajeOscuro;
}
```