Nicolás Martín Vega Gangemi
División 2C
Trabajo Práctico Final Nº4

# Trabajo Practico Nº4

## Índice del PDF


- Explicación de los proyectos creados (pág. 1)
- Errores corregidos del TP3 (pág. 2)
- Donde se utilizan los temas nuevos en el trabajo (pág. 3)
- Diagrama de clases (pág. 4 - 5)
- Objetivo Buscado (pág. 6)

Nicolás Martín Vega Gangemi
División 2C
Trabajo Práctico Final Nº4

# Explicación de los proyectos creados

La Clase Abstracta MateriaPrima de la cual heredaran Arrabio y Reciclado.
Dicha clase abstracta posee las propiedades de todos los atributos que luego se utilizaran a lo largo del ejercicio.

La clase MateriaPrimaDao es la encargada de interactuar con la Base de Datos.
El connectionString esta creado en el constructor statico de MateriaPrimaDAO y es indispensable para este TP:

```
static MateriaPrimaDAO()
{
    MateriaPrimaDAO.conexion = new SqlConnection(@"Data Source = .\SQLEXPRESS; Initial Catalog = Fabricacion;Integrated security = true");
    MateriaPrimaDAO.comando = new SqlCommand();

    MateriaPrimaDAO.comando.CommandType = CommandType.Text;
    comando.Connection = MateriaPrimaDAO.conexion;

}
```

Hay 6 Enumerados, que son los encargados de rellenar los comboBox del formulario.

Dentro de la carpeta Excepciones, encontrara 4 excepciones custom.

Hay un ExtensionMethod creado el cual otorgará un dato extra de información en el TP.

Cree una interfaz generica en la cual coloque dos métodos CalcularGanancia y EsValioso el cual cada Material deberá resolver de distintas maneras.

La otra clase creada el Stock en la cual se guardaran los materiales creados en una lista.

Dentro del proyecto Serializable, se encuentran las clases Mensaje y MensajeCargado que serán las encargadas de realizar el guardado en formato .txt y .xml.

En ConsolaTP3 se encuentra un código para probar la función las entidades entre sí y también se le agrego en este TP4 la capacidad de leer los archivos XML.

UnitTestTPFinal, se encuentran 4 metodos que retornan bool, verificados.

Por último, en FormPrincipal, se encuentra el formulario que permitirá al usuario agregar, editar, eliminar, filtrar, exportar la información que se almacene en dicha ejecución y ver en tiempo de ejecución información mediante un RichBoxText

Nicolás Martín Vega Gangemi
División 2C
Trabajo Práctico Final Nº4

# Errores corregidos del TP Nº3

1. Combo Calidad no se borra al Agregar.
    a. Honestamente no me sucedió nunca.
    Intenté forzar el error de todas las maneras que se me ocurrían, pero no sucedió.
    Revise el código y el *btnAgregar_Click*, utiliza el método *Limpiar()*;
    El cuál coloca todos los campos del formulario en vacíos.

```
/// <summary>
/// Establece los campos del formulario en vacios
/// </summary>
2 references
private void Limpiar()
{
    this.cmbProceso.SelectedIndex = -1;
    this.cmbMaterial.SelectedIndex = -1;
    this.txtOrigen.Text = string.Empty;
    this.txtCantidad.Text = string.Empty;
    this.cmbCalidad.SelectedIndex = -1;
    this.cmbColor.SelectedIndex = -1;
}
```

2. Los comboBox me dejan escribir lo que yo quiera saliendo de los parámetros prestablecidos.
    a. Cambie la propiedad de los comboBox: DropDownStyle y la coloque en DropDownList, lo que provoca que no se pueda escribir.

3. Editar solo me deja editar los nuevos
    a. Corregido. Ahora al proceder la información del DataSource de la base de datos, ya no se provoca ese problema.

4. No se lee nunca el XML
    a. En una primera instancia no sabía a que te referías. Al preguntarle a un compañero, me dijo que “Mi programa nunca lo lee”. Por esa razón en la consola hice un foreach que cargara unos archivos xml, que están en su carpeta:
    *TP4_FABRICACION/ConsolaTP3/bin/Debug/*
    b. Disculpe la desprolijidad de este método.

5. Mensaje no controla excepciones.
    a. Cree 2 excepciones que las aplico en el guardar de TXT y XML

6. No hay excepciones propias
    a. Cree 5 excepciones. 4 de ellas las llegue a implementar en el trabajo.

7. Hacer algún test que pruebe excepciones, archivos, cosas permeables a error.
    a. Hice 2 UnitTest que prueban el guardado del archivo TXT.

# Donde se utilizan los temas nuevos en el trabajo

- SQL: Dentro de la clase MateriaPrimaDao se encuentra todos los métodos que interactúan con la base de datos.
  A lo largo del formulario, estos métodos son llamados por los botones para interactuar.

- Hilos / Threads: En el formulario Principal “FrmPrincipal”, se créan 2 hilos.
  - Uno de ellos llamado **threadIniciador,** se instanciara en el Load del formulario y se encargará de utilizar un método, sin parametros, para estar constantemente actualizando la información de la Base de datos, y con esta información, iniciar el otro thread.
  - Este otro thread llamado **thread,** se instanciará con ParameterizedThreadStart y se le pasará un string de parámetro para que con la información del otro thread se vaya actualizando y se pueda utilizar.
    - Dentro del camino que toma este Thread, se utiliza un Callback.

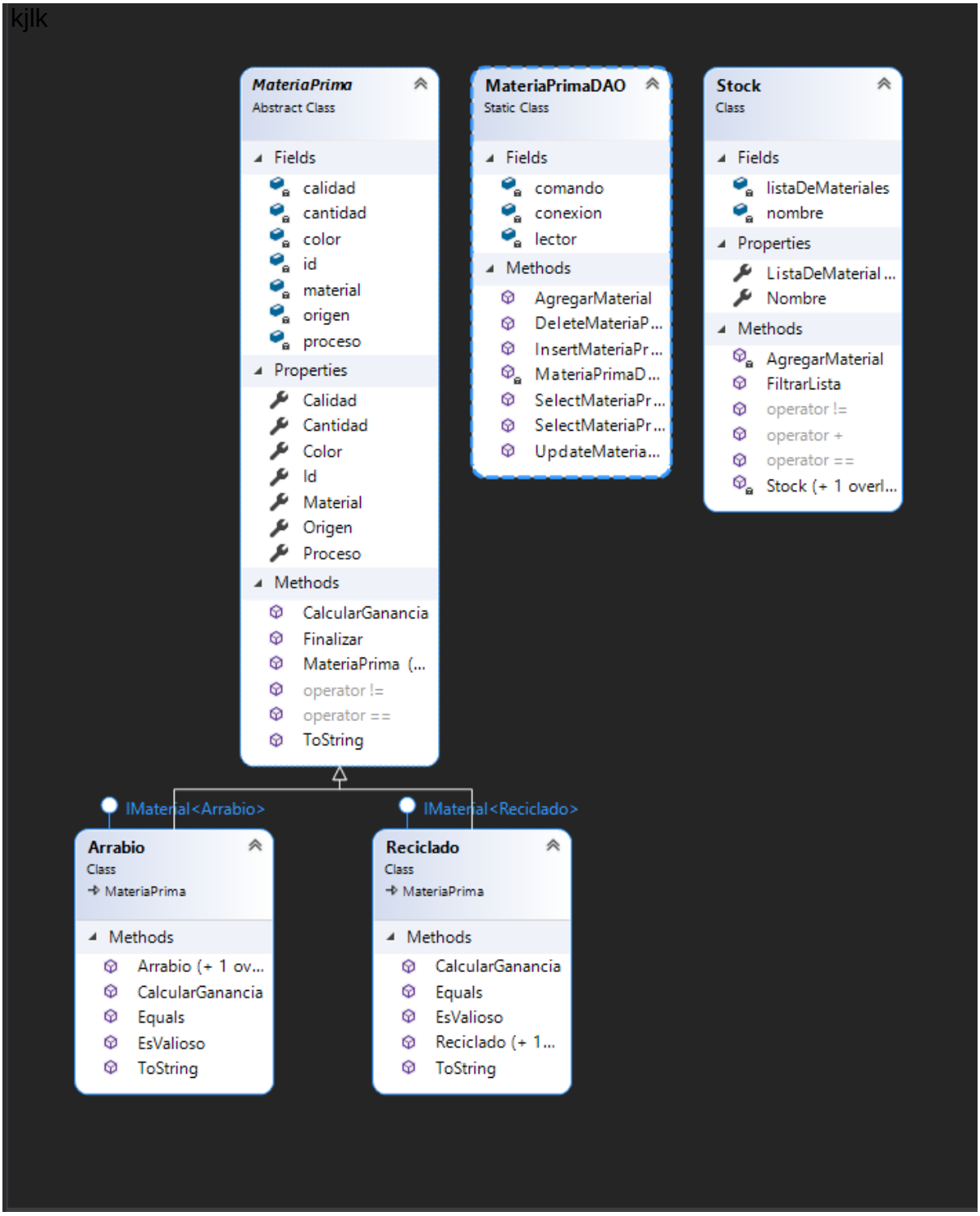
- Eventos y Delegados: ya vimos uno en la explicación del thread anterior.
  - Otro delegado creado también en el FrmPrincipal es el de Botones
  - Este juego de eventos se verá en la ejecución de Editar algún material.
  - Uno de ellos deshabilitará todos los botones menos el OkEditar y el otro realizará la función opuesta.

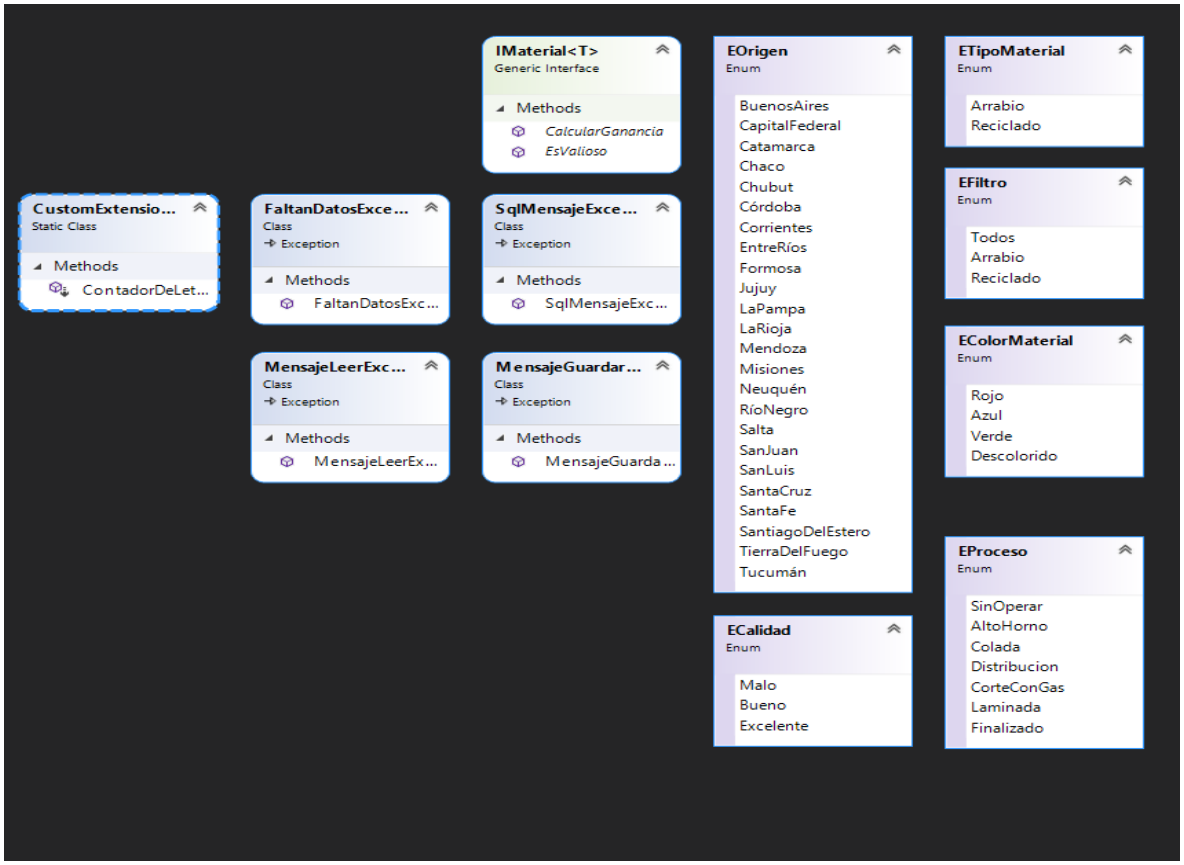
```
public delegate void Botones();
public event Botones okEditarEventHandler;
public event Botones okEditarDeshabilitarEventHandler;
```

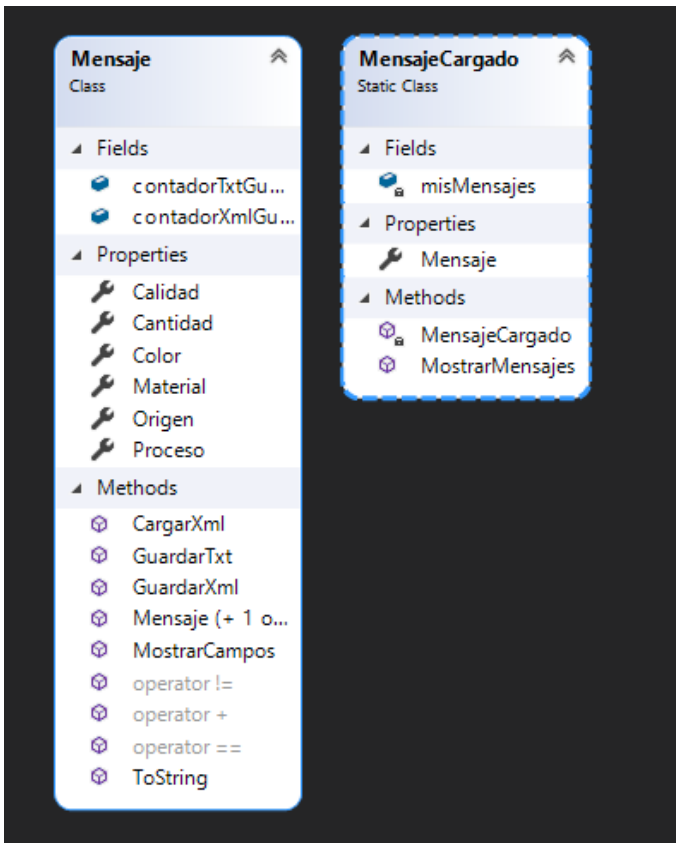
```
this.okEditarEventHandler += DeshabilitarBotonesMenosOkEditar;
this.okEditarDeshabilitarEventHandler += ActivarBotonesMenosOkEditar;
```

- Métodos de extensión: Se creó uno solo donde contará la cantidad de letras que tiene un string para devolverlo como un Int y de esta manera utilizar esa información en el TP

# Diagrama de clases




IMaterial<T>
Generic Interface
Methods
CalcularGanancia
EsValioso
EOrigen
Enum
BuenosAires
CapitalFederal
Catamarca
Chaco
Chubut
Córdoba
Corrientes
EntreRíos
Formosa
Jujuy
LaPampa
LaRioja
Mendoza
Misiones
Neuquén
RíoNegro
Salta
SanJuan
SanLuis
SantaCruz
SantaFe
SantiagoDelEstero
TierraDelFuego
Tucumán
ETipoMaterial
Enum
Arrabio
Reciclado
EFiltro
Enum
Todos
Arrabio
Reciclado
EColorMaterial
Enum
Rojo
Azul
Verde
Descolorido
EProceso
Enum
SinOperar
AltoHorno
Colada
Distribucion
CorteConGas
Laminada
Finalizado
ECalidad
Enum
Malo
Bueno
Excelente
CustomExtensio...
Static Class
Methods
ContadorDeLet...
FaltanDatosExce...
Class
Exception
Methods
FaltanDatosExc...
SqlMensajeExce...
Class
Exception
Methods
SqlMensajeExc...
MensajeLeerExc...
Class
Exception
Methods
MensajeLeerEx...
MensajeGuardar...
Class
Exception
Methods
MensajeGuarda...


Mensaje
Class
Fields
contadorTxtGu...
contadorXmlGu...
Properties
Calidad
Cantidad
Color
Material
Origen
Proceso
Methods
CargarXml
GuardarTxt
GuardarXml
Mensaje (+ 1 o...
MostrarCampos
operator !=
operator +
operator ==
ToString
MensajeCargado
Static Class
Fields
misMensajes
Properties
Mensaje
Methods
MensajeCargado
MostrarMensajes

# Objetivo Buscado

Mediante la ejecución del programa el usuario podrá agregar al DataGridView materiales por medio de rellenar los campos y presionar el botón “Agregar”.

Al presionar el botón “Editar” antes deberá seleccionar la fila deseada y así podrá editarla a su gusto, luego para terminar la edición deberá presionar el botón Ok editar (único habilitado).

Del mismo modo, luego de seleccionar una fila podrá tocar el botón “Eliminar” y dicho material desaparecerá.

Hay un comboBox que permitirá filtrar los datos del DataGrid según su Material.

A la derecha habrá un RichTextBox que cada 5 segundos, actualizará la información de la base de datos con otros mñetodos que actuarán en su proceso, por ejemplo calcular la ganancia por los materiales según su calidad y si esta finalizado o no su proceso.

Por último, al presionar “Exportar” todos los materiales que se hayan argegado al DataGridView durante la ejecución crearan un archivo txt y xml.