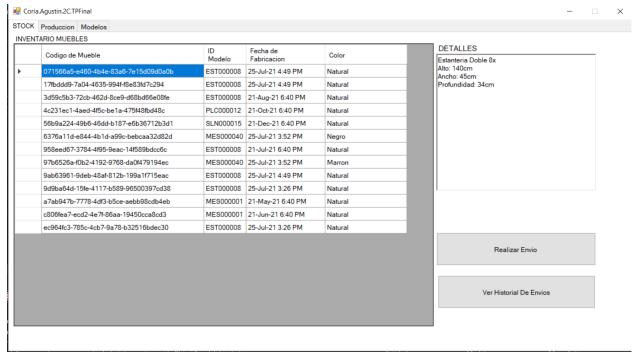
## EXPLICACIÓN DE LA APLICACIÓN:

El proyecto es una fábrica de muebles.

Mi fábrica tiene varias partes que interactúan entre sí y puedo manejar todas ellas desde mis formularios.

Antes que nada levantar la base de datos desde el archivo .bak , al final de este doc está el script de ser necesario.

## 1) Dale start



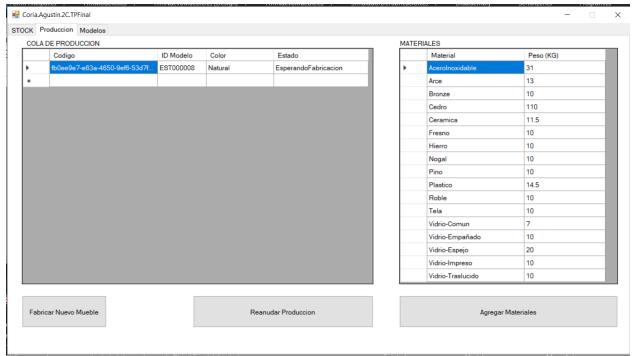
Todos los muebles que aparecen allí están fabricados. En detalles tenés más detalles que son levantados desde la base de datos que guarda todos los modelos. Está diseñado así porque puedo eliminar modelos pero seguir teniendo muebles que dependen de estos.

El **botón Realizar Envío** realizará un envío, se quitará al mueble de mi fábrica y se lo guardará en la base de datos.

El **botón Ver Historial De Envios** te llevará a un form con los detalles de TODOS los muebles enviados en el pasado. Deshabilitará el botón para que no se puedan abrir 2 al mismo tiempo. **FrmHistorialEnvios:** Podés ordenar los datos por columna y ascendiente/descendiente. Accede directamente a la base de datos.

 Al hacer un envío con el FrmHistorialEnvios abierto saltará una alerta y se actualizarán ambos formularios. TablaMuebleListener es la clase que se encarga de ver si se agregó un mueble a la base de datos y avisa con eventos cuando sucede.

## 2) Ahora vamos a la TAB de producción:



A la izquierda podemos ver todos los muebles que están en cola de producción (queue mueblesEnProduccion de mi fábrica).

Por defecto, al iniciar la aplicación, está pausada la producción.

El **botón Reanudar Produccion / Pausar Producción** inicializará o abortará el Thread encargado de la fabricación manejada por la clase **SimuladorDeFabricacion**.

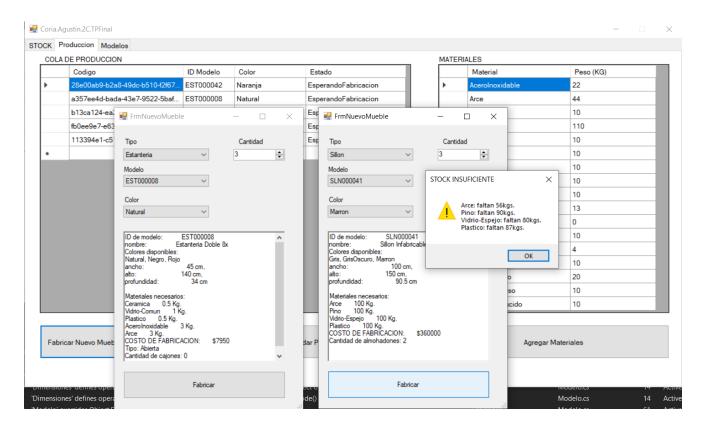
Este Thread utilizará un intervalo establecido en el archivo Globals.cs que representa milisegundos de espera. En 10 segundos cambiará el estado de "Esperando Fabricación" a "En Produccion", cuando pasen 10 segundos nuevamente se quitará el mueble de la queue y se lo pasará a la lista de stock, aparecerá en el tab STOCK.

El botón Agregar materiales me llevará al form **FrmAgregarMateriales**.

// Allí podremos elegir todos los materiales que queramos agregar a nuestro stock.

Con el botón Guardar y salir confirmás que se agregarán todos los materiales a nuestra fábrica.

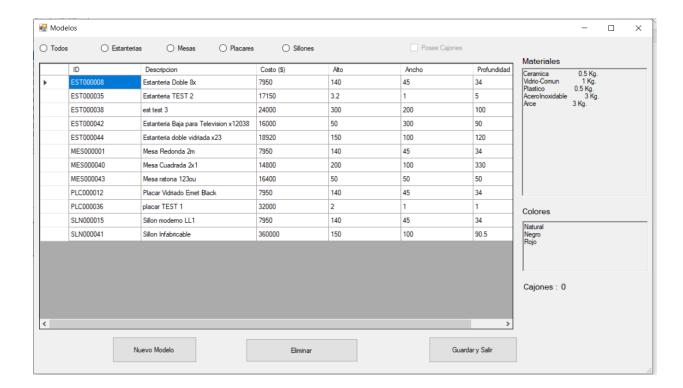
El **botón Fabricar Nuevo Mueble** nos dejará elejir un modelo, color y cantidad de mueble a fabricar. De no haber materiales en stock no se agregará a la queue y nos avisará mediante MessageBox cuántos materiales nos faltan.



## 3) TAB de Modelos:

Solo se mostrarán los modelos que existen en mi fábrica.

El botón Administrar Modelos nos abrirá un nuevo form:



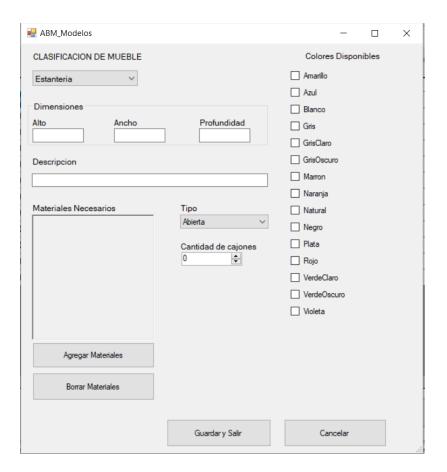
Los 5 radio buttons nos dejan filtrar modelos por clase.

El checkbox de cajones nos deja filtrar Estanterías y Placares para aquellos que tengan cajones.

Se pueden eliminar o agregar modelos.

Si no se sale del form con el botón Guardar y Salir NO SE GUARDARÁN LOS DATOS. Hecho de esta manera por si se elimina un modelo incorrecto.

El botón Nuevo Modelo nos abrirá un nuevo form: FrmNuevoMueble



Con el botón agregar materiales abriremos nuevamente un form de materiales.

Puedo borrar Materiales.

# Algunos valores se validan desde el form:

Que se haya seleccionado algún color.

Que los valores en Alto, Ancho, Profundidad sean numéricos.

Que la lista de materiales no esté vacía.

## Otros al querer construir el modelo:

Que las dimensiones sean mayores a 0 y menores a 10000

Que la descripción no esté vacía ni supere los 100 caracteres.

Estos últimos lanzarán excepciones cuyos mensajes se mostrarán por MessageBox.

## Información adicional:

- En el archivo Enumerados tendremos todos los enumerados utilizados.
- En la clase Globals se guardan los precios de los materiales, los paths de carpetas a usar para la serialización, el path a la base de datos y los prefijos utilizados para identificar las clases de modelos (Estantería, Mesa, Sillón, Placar).
  - Pueden modificarse los precios para ver los cambios

## **Utilización de Threads:**

- 1) El form de Historial de Envíos utiliza un thread para escuchar a la base de datos constantemente. Cuando esto sucede se avisa por MessageBox y se actualiza la vista de la datagridview.
- 2) Form principal, Thread nuevoMuebleThread se encarga de llamar al formulario de nuevo mueble y maneja los eventos cuando se agrega un mueble y cuando se cierra el formulario de nuevo mueble.

Agrega los muebles creados a la queue.

3) Form principal, Thread fabricandoMueblesThread . Constantemente pregunta si hay elementos en la cola de producción. De ser así, se lo fabricará y finalmente se lo moverá de la queue a la lista de stock de muebles de la fábrica.

### **Excepciones:**

- 1) ArchivolnvalidoException: Se disparará si al querer deserializar la fábrica alguno de los directorios, cuyo path se encuentra en la clase "Globals", este no existe.
- ErrorAlGuardarFabricaException: Se disparará si ocurre un error al serializar la fábrica en archivos.
- 3) ExceptionColorInvalido: Se disparará en el constructor parametrizado de Mueble si el color con el cual se lo quiere crear no pertenece a los colores disponibles de su modelo. Nunca debería poder suceder desde los formularios ya que los combobox correspondientes a los colores dependen del modelo seleccionado. Posee un método de prueba en el Unit test de Excepciones.
- **4) ExceptionDimensionesInvalidas:** Se dispara cuando cualquiera de los campos de Dimensiones supera 10.000 (cm).

Posee un método de prueba en el Unit test de Excepciones.

Se la llama en el botón Guardar del formulario NuevoModelo

Se puede probar ingresando datos inválidos al crear nuevo modelo

5) **ExceptionNombreModeloInvalido:** se dispara cuando se quiere crear un modelo sin descripción o una descripción muy larga (>100 caracteres).

Posee un método de prueba en el Unit test de Excepciones.

Se la llama en el botón Guardar del formulario NuevoModelo

Se puede probar ingresando descripción inválida al crear nuevo modelo

**6) ExceptionSillonSinAlmohadones:** Se dispara al querer crear un modelo de sillón sin almohadones.

Posee un método de prueba en el Unit Test de Excepciones

## Interfaz ICajonera:

Los modelos que la implementan tienen propiedad de get y set para su cantidad de cajones así como un método que indica si el modelo tiene más de 0 cajones.

Se utiliza en el FrmModelos que administra los modelos para mostrar los cajones de cada modelo y filtrar por aquellos modelos que tienen cajones.

#### Materiales:

## - Clase Abstracta Material:

Posee un atributo que guarda el Peso de un material.

<u>Propiedad Key</u>: Devuelve un string identificador del material. Si tengo dos materiales de la clase Vidrio pero sus tipos son distintos (Espejo vs Común por ej.) los podré distinguir. <u>Propiedad Costo Total</u>: hará peso \* precio del material (stored in Globals.cs)

El método <u>PuedeGastar()</u> me asegura que puedo restar peso de un material. Es útil para asegurarme, antes de fabricar un mueble, que todos los materiales en su lista de materiales, existen en mi fábrica y tengo suficiente stock.

Los métodos <u>Gastar</u>, <u>Setear y Agregar Peso ()</u>, manipularán el atributo de pesoEnKg. El override al <u>método equals</u> me dirá si dos materiales son de la misma clase y además del mismo tipo, utilizando la propiedad Key.

### Clases derivadas de Material

### Madera, Metal, Vidrio, Ceramica, Tela, Plastico

Harán override el método Key para identificarse.

Harán override del método CostoTotal para utilizar cada una la variable global que corresponda a su precio.

El método ToString() estará sobreescrito para devolver Key.

Más información sobre los tipos en archivo Enumerados.cs

#### **Estructura Dimensiones**

Posee Alto, ancho y profundidad.

Dos dimensiones serán iguales si todos sus atributos coinciden.

### Clase abstracta Modelo

- Tiene un <u>atributo estático uniqueID</u> que se levanta de un archivo cada vez que se inicia la aplicación.
- El atributo Identificador será seteado en cada clase derivada al construir un modelo.
- Posee propiedades públicas para sus atributos. No se podrá crear un modelo con dimensiones inválidas (< 1 || > 10000).
- La <u>propiedad CostoDeFabricación</u> no será serializada. Se calcula en el momento que se la precisa ya que depende de variables globales de precios.

- No se podrá crear un modelo con un nombre vacío, null, o de longitud mayor a 100 caracteres.
- El <u>método ValidarColor</u> valida que el color recibido existe como uno de los colores disponibles del modelo.
- El método genérico ContienenMismosObjetos comparará dos array de objetos y verá que no sean null y que cada uno contenga los elementos del otro. Ignora el orden de los objetos.
- El <u>método Equals</u> me dirá si dos modelos son iguales comparando el Type, las dimensiones, los colores y los materiales necesarios.
- El método ToString devolverá solo el Identificador único.
- El <u>método Mostrar()</u> mostrará todos los detalles del modelo en particular. Será sobreescrito para agregar algunos atributos por las clases derivadas.

#### Clases derivadas de Modelo:

En los constructores de cada una se asigna un id único que usara UniqueID, lo incrementará en 1, lo convertirá a un string de 6 caracteres (utiliza PadLeft) anteponiendo ceros y agregará el prefijo correspondiente.

Los métodos Mostrar y Equals añadirán a la funcionalidad del Equals de Modelo, y agregarán validación o muestra de los atributos que posean.

### ModeloEstanteria:

Implementa ICajonera

Tiene tipo EEstantería (Ver Enumerados.cs)

Tiene Cajones, por defecto en 0.

#### ModeloMesa:

Tiene tipo EEspacio (Ver Enumerados.cs)

#### ModeloPlacar:

Implementa ICajonera

Tiene Cajones, por defecto en 0.

#### ModeloSillon:

Tiene atributo cantidadAlmohadones

En la propiedad get, set CantidadAlmohadones se lanza una excepción si se quiere crear un modelo de sillón SIN almohadones.

## **Clase Mueble**

- El constructor parametrizado recibe un Modelo y guarda su ID. Recibe una fecha y la guarda. Recibe un color y llama al método ValidarColor de modelo. Lanza ExceptionColorInvalido si el color no corresponde.
- Si el constructor no recibe color se setea en EColor.Natural.

- Dos muebles serán iguales si coincide su Guid.
- Tendrán estados: EsperandoFabricacion, EnProduccion, Terminado, Enviado

### Clase Fabrica

Guarda un Stock de materiales Guarda una lista de modelos disponibles Guarda una lista de Muebles Fabricados Guarda una queue de Muebles por construir

Al querer agregar un modelo a la fábrica siempre se verifica que tenga bien sus datos y que no exista ya en mi fábrica.

Agregar stock de materiales es sencillo, gastarlos no tanto. Al construir un mueble se verificará que de cada material que este precisa para ser fabricado haya stock. De lo contrario no se gastará nada.

Los muebles cambiarán al estado EsperandoFabricacion al ingresar a la queue de fabricación. Cambiarán de estado a EnProduccion sin moverse de la queue.

Cambiarán a estado Terminado cuando finalize su producción y serán movidos a la lista de stock de muebles.

Podrán ser "enviados" desde el formulario principal. Se removerán de mi fabrica pero quedarán back-upeados en la base de datos. Podré verlos desde el form principal.

Posee métodos de Leer y Guardar que utilizan serialización y deserialización XML y Binaria.

## UnitTesting:

Se prueban comparaciones de materiales y modelos para asegurarme de que no se añaden objetos que no corresponden a la fábrica.

Se prueban Excepciones, algunas que nunca ocurrirán desde los formularios.

#### Prueba En Consola:

Se puede eliminar del proyecto si se quiere.

Fuí probando agregar modelos, materiales y serialización de mi fábrica cuando aún no tenía mis formularios armados.

A partir de la linea 222 empieza la serialización y deserialización.

#### Archivos y serialización:

Clase Genérica Serializer. Recibe la Clase de los archivos a serializar y deserializar.

Usa un delegado para darle nombre a cada objeto que serializa, así puedo personalizar el título del archivo.

Utilizada en Métodos Guardar y Leer de Fábrica.

### Base de Datos:

Existe un archivo .bak. en la carpeta principal

```
De todas maneras aquí está el script:
USE [master]
GO
/***** Object: Database [Fabrica TP4 AC] Script Date: 25-Jul-21 11:25:06 PM ******/
CREATE DATABASE [Fabrica TP4 AC]
CONTAINMENT = NONE
ON PRIMARY
( NAME = N'Fabrica TP4 AC', FILENAME = N'C:\Program Files\Microsoft SQL
Server\MSSQL15.MSSQLSERVER\MSSQL\DATA\Fabrica TP4 AC.mdf', SIZE = 8192KB,
MAXSIZE = UNLIMITED, FILEGROWTH = 65536KB)
LOG ON
( NAME = N'Fabrica_TP4_AC_log', FILENAME = N'C:\Program Files\Microsoft SQL
Server\MSSQL15.MSSQLSERVER\MSSQL\DATA\Fabrica_TP4_AC_log.ldf', SIZE = 8192KB,
MAXSIZE = 2048GB, FILEGROWTH = 65536KB)
WITH CATALOG COLLATION = DATABASE DEFAULT
GO
IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))
begin
EXEC [Fabrica TP4 AC].[dbo].[sp fulltext database] @action = 'enable'
end
GO
ALTER DATABASE [Fabrica_TP4_AC] SET ANSI_NULL_DEFAULT OFF
GO
ALTER DATABASE [Fabrica_TP4_AC] SET ANSI_NULLS OFF
GO
ALTER DATABASE [Fabrica_TP4_AC] SET ANSI_PADDING OFF
GO
ALTER DATABASE [Fabrica TP4 AC] SET ANSI WARNINGS OFF
ALTER DATABASE [Fabrica_TP4_AC] SET ARITHABORT OFF
ALTER DATABASE [Fabrica_TP4_AC] SET AUTO_CLOSE OFF
GO
```

ALTER DATABASE [Fabrica\_TP4\_AC] SET AUTO\_SHRINK OFF GO

ALTER DATABASE [Fabrica\_TP4\_AC] SET AUTO\_UPDATE\_STATISTICS ON GO

ALTER DATABASE [Fabrica\_TP4\_AC] SET CURSOR\_CLOSE\_ON\_COMMIT OFF GO

ALTER DATABASE [Fabrica\_TP4\_AC] SET CURSOR\_DEFAULT GLOBAL GO

ALTER DATABASE [Fabrica\_TP4\_AC] SET CONCAT\_NULL\_YIELDS\_NULL OFF GO

ALTER DATABASE [Fabrica\_TP4\_AC] SET NUMERIC\_ROUNDABORT OFF GO

ALTER DATABASE [Fabrica\_TP4\_AC] SET QUOTED\_IDENTIFIER OFF GO

ALTER DATABASE [Fabrica\_TP4\_AC] SET RECURSIVE\_TRIGGERS OFF GO

ALTER DATABASE [Fabrica\_TP4\_AC] SET DISABLE\_BROKER GO

ALTER DATABASE [Fabrica\_TP4\_AC] SET AUTO\_UPDATE\_STATISTICS\_ASYNC OFF GO

ALTER DATABASE [Fabrica\_TP4\_AC] SET DATE\_CORRELATION\_OPTIMIZATION OFF GO

ALTER DATABASE [Fabrica\_TP4\_AC] SET TRUSTWORTHY OFF GO

ALTER DATABASE [Fabrica\_TP4\_AC] SET ALLOW\_SNAPSHOT\_ISOLATION OFF GO

ALTER DATABASE [Fabrica\_TP4\_AC] SET PARAMETERIZATION SIMPLE GO

ALTER DATABASE [Fabrica\_TP4\_AC] SET READ\_COMMITTED\_SNAPSHOT OFF GO

```
ALTER DATABASE [Fabrica_TP4_AC] SET HONOR_BROKER_PRIORITY OFF
GO
ALTER DATABASE [Fabrica_TP4_AC] SET RECOVERY FULL
GO
ALTER DATABASE [Fabrica_TP4_AC] SET MULTI_USER
GO
ALTER DATABASE [Fabrica TP4 AC] SET PAGE VERIFY CHECKSUM
GO
ALTER DATABASE [Fabrica_TP4_AC] SET DB_CHAINING OFF
GO
ALTER DATABASE [Fabrica_TP4_AC] SET FILESTREAM( NON_TRANSACTED_ACCESS =
OFF)
GO
ALTER DATABASE [Fabrica TP4 AC] SET TARGET RECOVERY TIME = 60 SECONDS
GO
ALTER DATABASE [Fabrica TP4 AC] SET DELAYED DURABILITY = DISABLED
GO
ALTER DATABASE [Fabrica_TP4_AC] SET ACCELERATED_DATABASE_RECOVERY = OFF
GO
ALTER DATABASE [Fabrica_TP4_AC] SET QUERY_STORE = OFF
GO
ALTER DATABASE [Fabrica_TP4_AC] SET READ_WRITE
GO
USE [Fabrica TP4 AC]
GO
SET ANSI NULLS ON
GO
SET QUOTED IDENTIFIER ON
GO
```

```
CREATE TABLE [dbo].[Modelo](
      [Identificador] [varchar](50) NOT NULL,
     [Descripcion] [varchar](300) NULL,
     [Alto] [float] NOT NULL,
     [Ancho] [float] NOT NULL,
     [Profundidad] [float] NOT NULL,
PRIMARY KEY CLUSTERED
     [Identificador] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY =
OFF, ALLOW ROW LOCKS = ON, ALLOW PAGE LOCKS = ON,
OPTIMIZE FOR SEQUENTIAL KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO
USE [Fabrica_TP4_AC]
GO
SET ANSI NULLS ON
GO
SET QUOTED IDENTIFIER ON
GO
CREATE TABLE [dbo].[Mueble](
     [Guid] [varchar](50) NOT NULL,
     [IdModelo] [varchar](50) NOT NULL,
     [FechaDeFabricacion] [datetime] NULL,
     [FechaDeEnvio] [datetime] NULL,
     [Color] [varchar](50) NOT NULL,
CONSTRAINT [PK_Mueble] PRIMARY KEY CLUSTERED
      [Guid] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY =
OFF, ALLOW ROW LOCKS = ON, ALLOW PAGE LOCKS = ON,
OPTIMIZE FOR SEQUENTIAL KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO
ALTER TABLE [dbo].[Mueble] WITH CHECK ADD FOREIGN KEY([IdModelo])
REFERENCES [dbo].[Modelo] ([Identificador])
GO
```