

El programa creado se usa para la gestión de cortes de una fabrica textil, la cual puede fabricar solo las marcas implementadas.

(1 corte de ropa son muchas prendas, generalmente salen en curvas parejas)

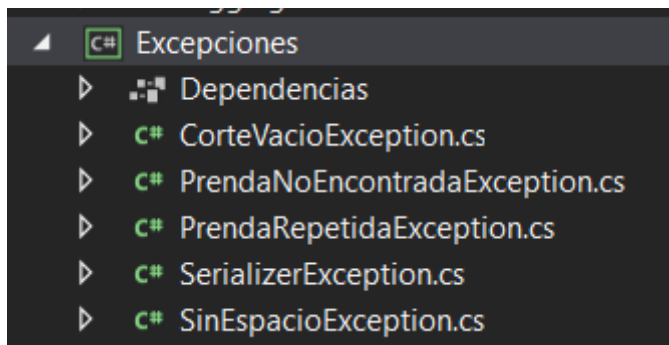
En las funcionalidades que podemos encontrar tenemos el ABM y también exportar e importar en un formato serializado, el archivo es creado y tomado desde el escritorio.

La fabrica tiene una capacidad de fabricar la cantidad de metros de tela disponible.

**\*ACLARACION:** Una vez creado el corte si se elimina el corte no afectara a la tela disponible ya que no se puede volver a usar la tela que ya se confecciono.

Temas de clases implementados:

**Clase 15 – Excepciones:** Posee 5 excepciones las cuales se detallan a continuación, también atrapa excepciones no controladas por si ocurre algún error no esperado.



**Clase 16 – Test Unitarios:** Se realizaron 3 pruebas diferentes y corrieron con éxito

- 1) Que se instancien las listas de la fabrica
- 2) Al agregar un corte nuevo superar el limite y que lance una excepción
- 3) Que agregue bien un corte

**Clase 17 – Tipos genéricos:** En la clase para guardar el archivo se implemento ya que puede recibir un tipo de dato genérico para luego generalizar el que sea.

```
4 referencias
public class ImpExpXML<T> : ISerializer<T>
{
    // ...
}
```

**Clase 18 – Interfaces:** En la clase indumentaria y en el proyecto de serilizer se implementó este tema, indicándole la estructura que debe usar

```

13 referencias
interface IPrenda
{
    int Id { get; set; }
    9 referencias
    int S { get; set; }
    9 referencias
    int M { get; set; }
    9 referencias
    int L { get; set; }
    9 referencias
    int XL { get; set; }
    9 referencias
    int XXL { get; set; }

    3 referencias
    enum Prenda Tipo { get; set; }

    4 referencias
    enum Marca Marca { get; set; }

    4 referencias
    string Modelo { get; set; }

    8 referencias
    string Variante { get; set; }
}

1 referencia
interface ISerializer<T>
{
    2 referencias
    bool Guardar(string path, T datos);

    2 referencias
    bool Leer(string path, out T datos);
}

```

**Clase 19 – Archivos y serialización:** En el proyecto Serializer esta implementado todo en formato XML

```

4 referencias
public bool Guardar(string path, T data)
{
    XmlTextWriter writer = null;
    XmlSerializer serializer = null;
    try
    {
        writer = new XmlTextWriter(path, Encoding.UTF8);
        writer.Formatting = Formatting.Indented;
        serializer = new XmlSerializer(typeof(T));
        serializer.Serialize(writer, data);
        return true;
    }
    catch (Exception ex)
    {
        throw new SerializerException(ex);
    }
    finally
    {
        if (writer != null)
        {
            writer.Close();
        }
    }
}

3 referencias
public bool Leer(string path, out T data)
{
    XmlTextReader reader = null;
    XmlSerializer serializer = null;
    try
    {
        reader = new XmlTextReader(path);
        serializer = new XmlSerializer(typeof(T));
        data = (T)serializer.Deserialize(reader);
        return true;
    }
    catch (Exception ex)
    {
        throw new SerializerException(ex);
    }
    finally
    {
        if (reader != null)
        {
            reader.Close();
        }
    }
}

```

**Clase 21/22 – Bases de datos:** En la clase acceso a datos.

```

public List<Indumentaria> LoadFromBD()
{
    List<Indumentaria> lista = new List<Indumentaria>();

    try
    {
        this.comando = new SqlCommand();
        this.comando.CommandType = CommandType.Text;
        this.comando.Connection = this.conexion;
        comando.CommandText = "SELECT * FROM Cortes ORDER BY Id";
        this.conexion.Open();
        SqlDataReader odr = comando.ExecuteReader();

        while (odr.Read())
        {
            if (odr["Tipo"].ToString() == "Joggings")
            {
                int Id = odr.GetInt32(0);
                int S = odr.GetInt32(1);

                Joggings corte1 = new Joggings(odr.GetInt32(0), odr.GetInt32(1), odr.GetInt32(2), odr.GetInt32(3), odr.GetInt32(4));
                if (odr["Variante"].ToString() == "Con punio")
            }
        }
    }
}

```

**Clase 23 – Hilos:** En la clase del frmPrincipal se aplica un hilo para refrescar cada 500 ms los datos.

```

public void ActualizarGrilla()
{
    try
    {
        while (this.pauseThread == false)
        {
            Thread.Sleep(500);

            if (this.dtGdViewCortes.SelectedRows.Count > 0)
            {
                rowSelected = this.dtGdViewCortes.SelectedRows[0].Index;
            }

            if (this.dtGdViewCortes.InvokeRequired)
            {
                this.dtGdViewCortes.BeginInvoke((MethodInvoker)delegate ()
                {
                    dtGdViewCortes.DataSource = null;
                    this.dtInd = _acceso.LoadDTFromBD();

                    dtGdViewCortes.DataSource = this.dtInd;
                    configurarGrilla();

                });
            }
        }
    }
    catch (Exception)
    {
    }
}

```

**Clase 24 – Eventos y delegados:** En la clase AccesoDatos tiene un evento para corroborar la tela disponible, en caso de que no sea suficiente lanza una excepcion

```

private void SinTela(object sender, EventArgs e)
{
    if (sender is double)
    {
        if (_acceso.TelaDisponibleBD() < (double)sender)
        {
            throw new TelaInsuficienteException();
        }
    }
}

```

```

21 private DataTable dtInd;
22 private AccesoDatos _acceso;
23 Thread hiloActualizador;
24 private bool pauseThread = false;
25 private int rowSelected = -1;
26 // <summary>
27 // Constructor por defecto
28 // </summary>
29 // referencia
30 public frmPrincipal()
31 {
32     InitializeComponent();
33     this.FB = new Fabrica();
34     this._acceso = new AccesoDatos();
35     this.dtInd = new DataTable();
36     this.dtInd = _acceso.LoadDTFromBD();
37     this.lstInd = new List<Indumentaria>();
38     this.btnImport.Click += new System.EventHandler(this.Importar);
39     this.FormClosing += new System.Windows.Forms.FormClosingEventHandler(this.Cerrar);
40
41     //asocio el evento
42     this._acceso.SinTelaEvent += SinTela;
43
44     this.hiloActualizador = new Thread(this.ActualizarGrilla);

```

```

public delegate void Delegado(object sender, EventArgs e);
public event Delegado SinTelaEvent;

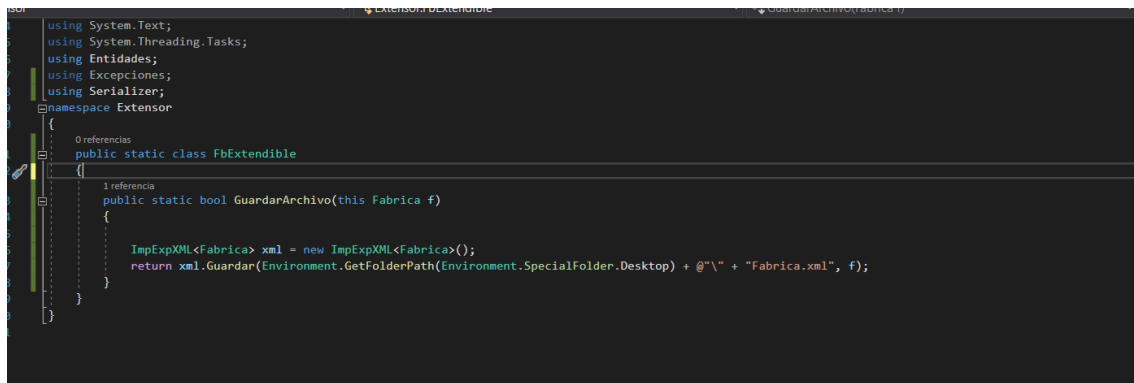
```

```

if (this.SinTelaEvent != null)
{
    this.SinTelaEvent(metrosTela, EventArgs.Empty);
}

```

**Clase 25 – Métodos de extensión:** En el proyecto extensor, se extiende la clase fabrica para guardar el archivo .



```
using System.Text;
using System.Threading.Tasks;
using Entidades;
using Excepciones;
using Serializers;
namespace Extensor
{
    0 referencias
    public static class FbExtendible
    {
        1 referencia
        public static bool GuardarArchivo(this Fabrica f)
        {
            ImpExpXML<Fabrica> xml = new ImpExpXML<Fabrica>();
            return xml.Guardar(Environment.GetFolderPath(Environment.SpecialFolder.Desktop) + @"\" + "Fabrica.xml", f);
        }
    }
}
```

Para crear la BD se debe ejecutar las siguientes instrucciones:

```
USE [Santoliquido_TP4_BD]
```

```
GO
```

```
/** Object: Table [dbo].[Cortes]  Script Date: 9/7/2021 18:29:56 **/
```

```
SET ANSI_NULLS ON
```

```
GO
```

```
SET QUOTED_IDENTIFIER ON
```

```
GO
```

```
CREATE TABLE [dbo].[Cortes](
    [Id] [bigint] IDENTITY(1,1) NOT NULL,
    [S] [int] NULL,
    [M] [int] NULL,
    [L] [int] NULL,
    [XL] [int] NULL,
    [XXL] [int] NULL,
    [Tipo] [varchar](50) NULL,
    [Marca] [varchar](50) NULL,
    [Modelo] [varchar](50) NULL,
    [Variante] [varchar](50) NULL,
```

```
CONSTRAINT [PK_Cortes] PRIMARY KEY CLUSTERED
(
    [Id] 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
```