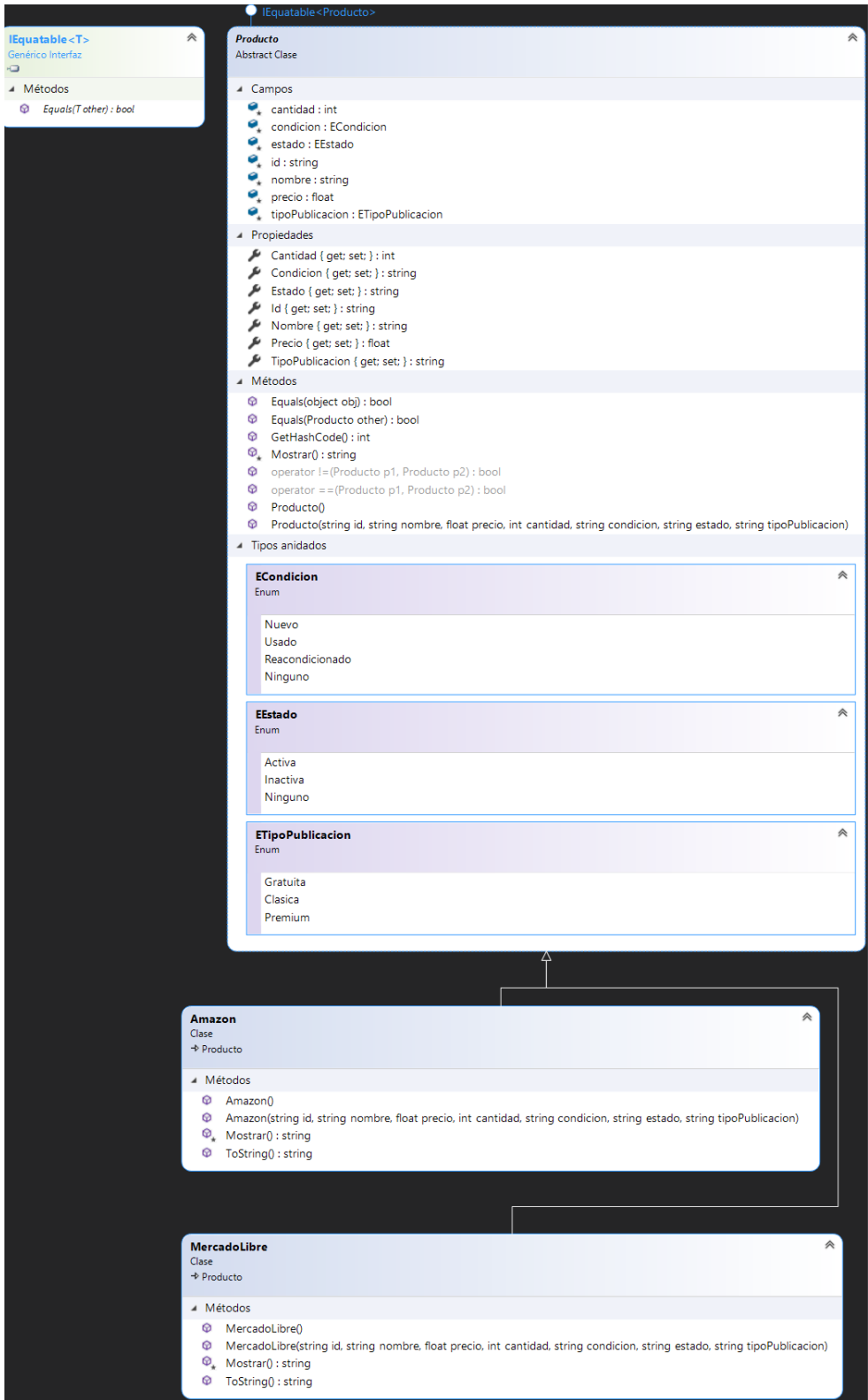


La aplicación que realice gestiona productos ya sea tanto de tipo Amazon como del tipo Mercado Libre que estarán almacenados dentro de un depósito genérico

A continuación detallo las funcionalidades de la aplicación:

- Gestiona productos: agrega, elimina, reemplaza
- Realiza control de pausadas
- Genera informes de los productos que están en el depósito
- Cargar y guardar archivos txt y serializaciones xml de los productos

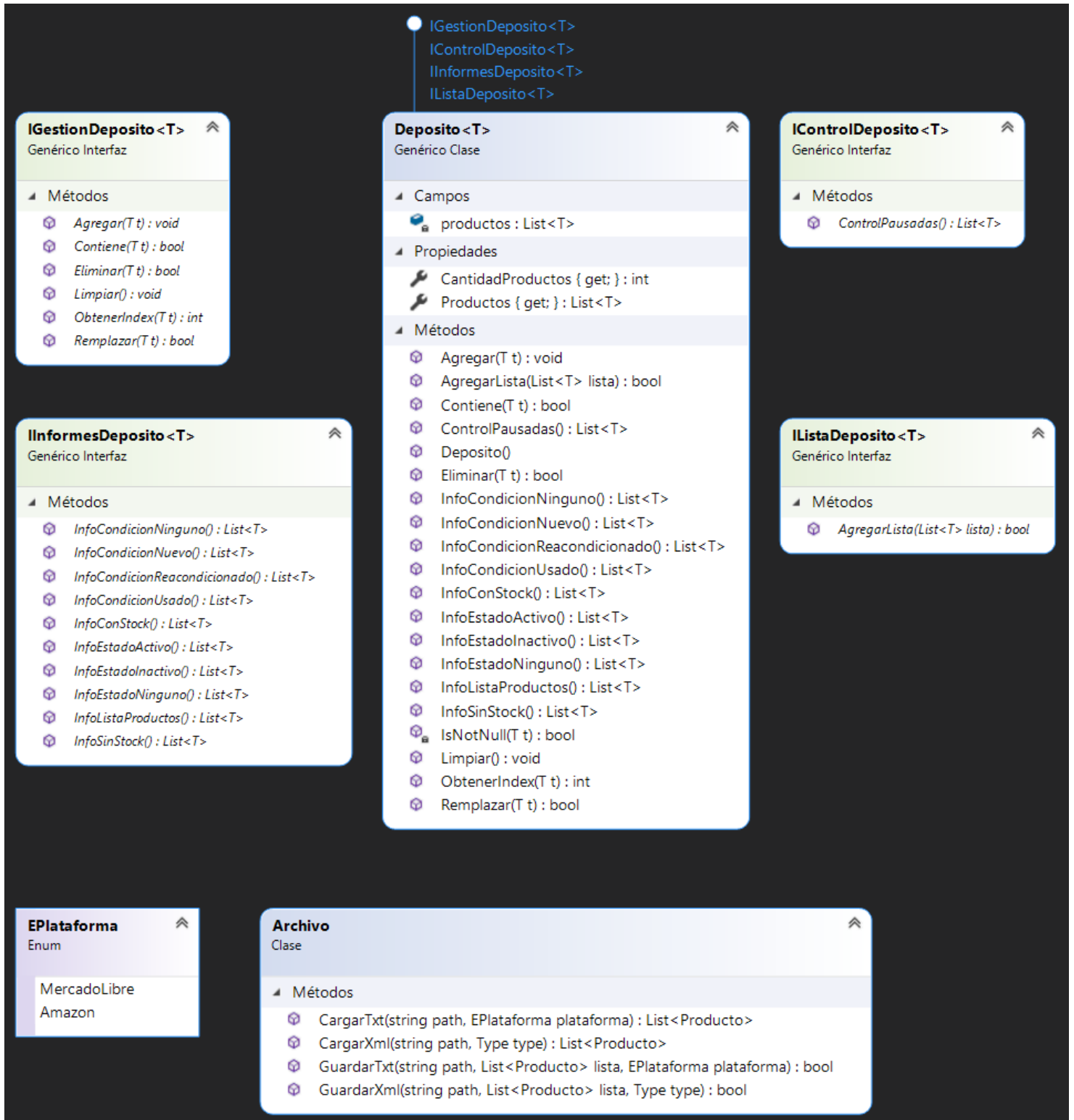
Jerarquía de Producto



- **Clase Depósito con sus interfaces y La clase Archivo**

La clase genérica Depósito almacena y gestiona un depósito de productos

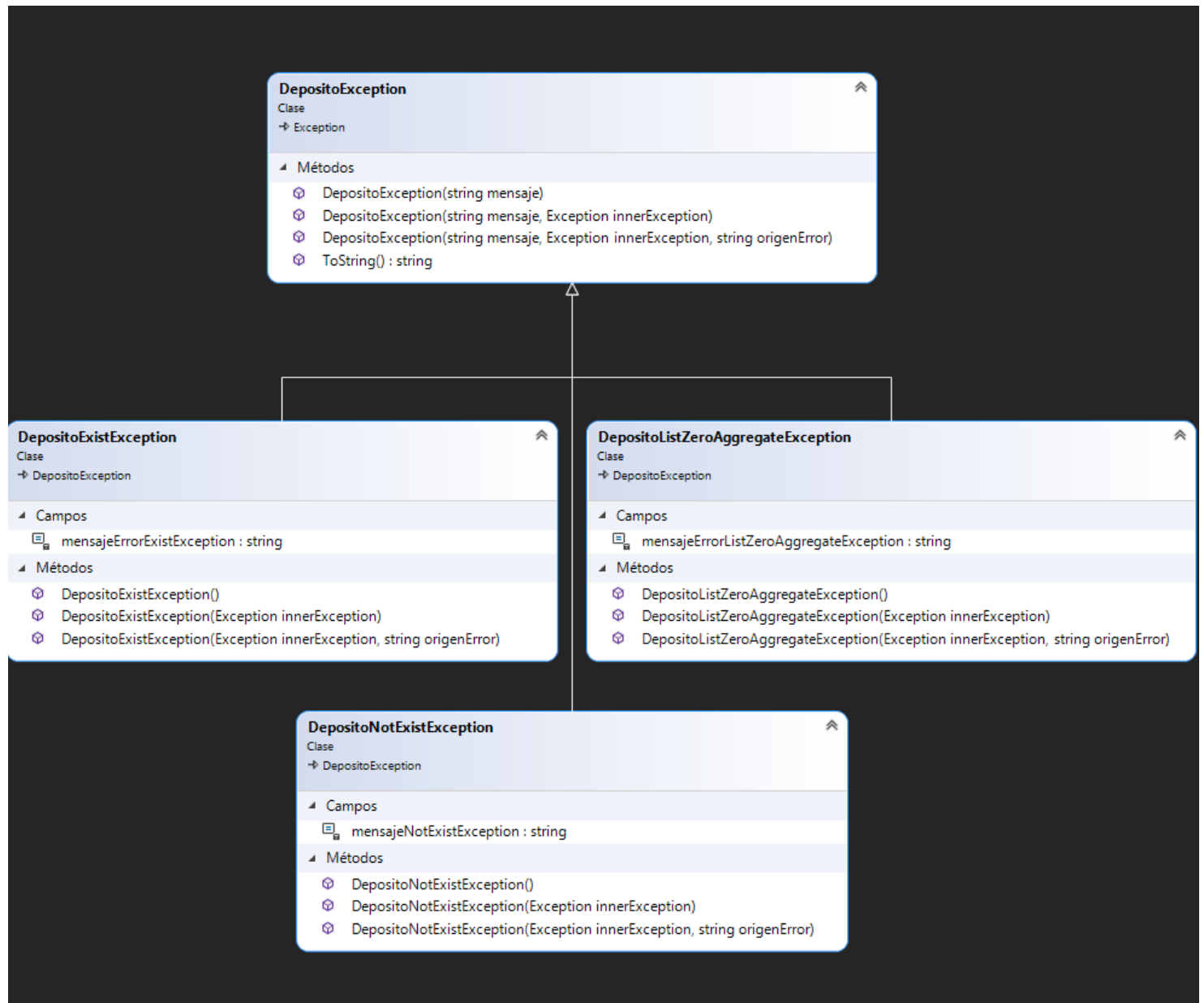
La clase Archivo es la encargada de leer y guardar archivos txt y serializaciones xml.



Jerarquía DepositoException

Excepciones del Depósito:

- **DepositoExistException:** Es cuando existe un producto en el depósito y se quiere agregar uno igual.
- **DepositoNotExistException:** Es cuando no existe un producto en el depósito y quiere por ejemplo eliminar ese producto.
- **DepositoListZeroAggregateException:** Es cuando se le pasa una lista al depósito y esta lista no tiene ningún producto.



1. Excepciones

Las excepciones las utilizó en las clases: Archivo, Depósito y en el Windows Form

- **Método CargarTxt de la clase Archivo**

Si es null el path, lanzó una excepción del tipo ArgumentException, la excepción la capturó cuando invocó al método

```
/// <summary> Carga un archivo txt de productos
10 referencias | 5/5 pasando | 0 cambios | 0 autores, 0 cambios
public static List<Producto> CargarTxt(string path, EPlataforma plataforma)
{
    List<Producto> listaAux = new List<Producto>();
    string linea;
    string[] array;

    if (path is not null)
    {
        using (StreamReader reader = new StreamReader(path, Encoding.UTF8))
        {
            while (!reader.EndOfStream)
            {
                linea = reader.ReadLine();
                array = linea.Split(',');
                float.TryParse(array[2], out float precio);
                int.TryParse(array[3], out int cantidad);

                if (plataforma == EPlataforma.MercadoLibre)
                {
                    listaAux.Add(new MercadoLibre(array[0], array[1], precio, cantidad, array[4], array[5], array[6]));
                }
                else if (plataforma == EPlataforma.Amazon)
                {
                    listaAux.Add(new Amazon(array[0], array[1], precio, cantidad, array[4], array[5], array[6]));
                }
            }
        }
    }
    else
    {
        throw new ArgumentException("Archivo", "La ruta al archivo txt es null");
    }

    return listaAux;
}
```

- **Evento mercadoLibreToolStripMenuItemCargar_Click del Formulario principal**

Este evento se encarga de obtener el path al archivo y luego cargar estos productos al depósito. Si el path es nulo lanza una excepción, capturo esa excepción y también agregó el catch de la excepción genérica por si se desencadena otra excepción, Muestro un mensaje con el MessageBox informando del error.

```
/// <summary> Obtiene la ruta al archivo txt
1 referencia | 0 cambios | 0 autores, 0 cambios
private void mercadoLibreToolStripMenuItemCargar_Click(object sender, EventArgs e)
{
    this.pgrssBarEstadoCarga.EstadoBarra = false;
    string path = this.AbrirArchivo("MercadoLibre", ".txt", out string nombreArchivo);

    try
    {
        this.lblNombreArchivo.Text = nombreArchivo;
        this.deposito.AgregarLista(Archivo.CargarTxt(path, EPlataforma.MercadoLibre));
        this.pgrssBarEstadoCarga.EstadoBarra = true;
    }
    catch (ArgumentException exception)
    {
        MessageBox.Show(exception.Message, "ERROR", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    catch (Exception exception)
    {
        MessageBox.Show(exception.Message, "ERROR", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

1.2 Excepciones Propias

DepositoExistException

- **Método Agregar de la clase Archivo**

La función agrega un producto al depósito, si este producto a ingresar ya existe retorna una excepción del tipo DepositoExistException y la capturo cuando invocó al método.

```
//IMPLEMENTACION INTERFAZ IgestionDeposito<T>
/// <summary>
/// Agrega un producto al deposito
/// </summary>
/// <param name="t">Producto</param>
/// <exception cref="ArgumentNullException">Si el producto es null</exception>
/// <exception cref="DepositoExistException">Si el producto ya existe en el deposito</exception>
24 referencias | 12/12 pasando | 0 cambios | 0 autores, 0 cambios
public void Agregar(T t)
{
    if (this.Contiene(t))
    {
        throw new DepositoExistException();
    }
    else
    {
        this.productos.Add(t);
    }
}
```

- **Evento btn_Agregar_Click del Formulario principal**

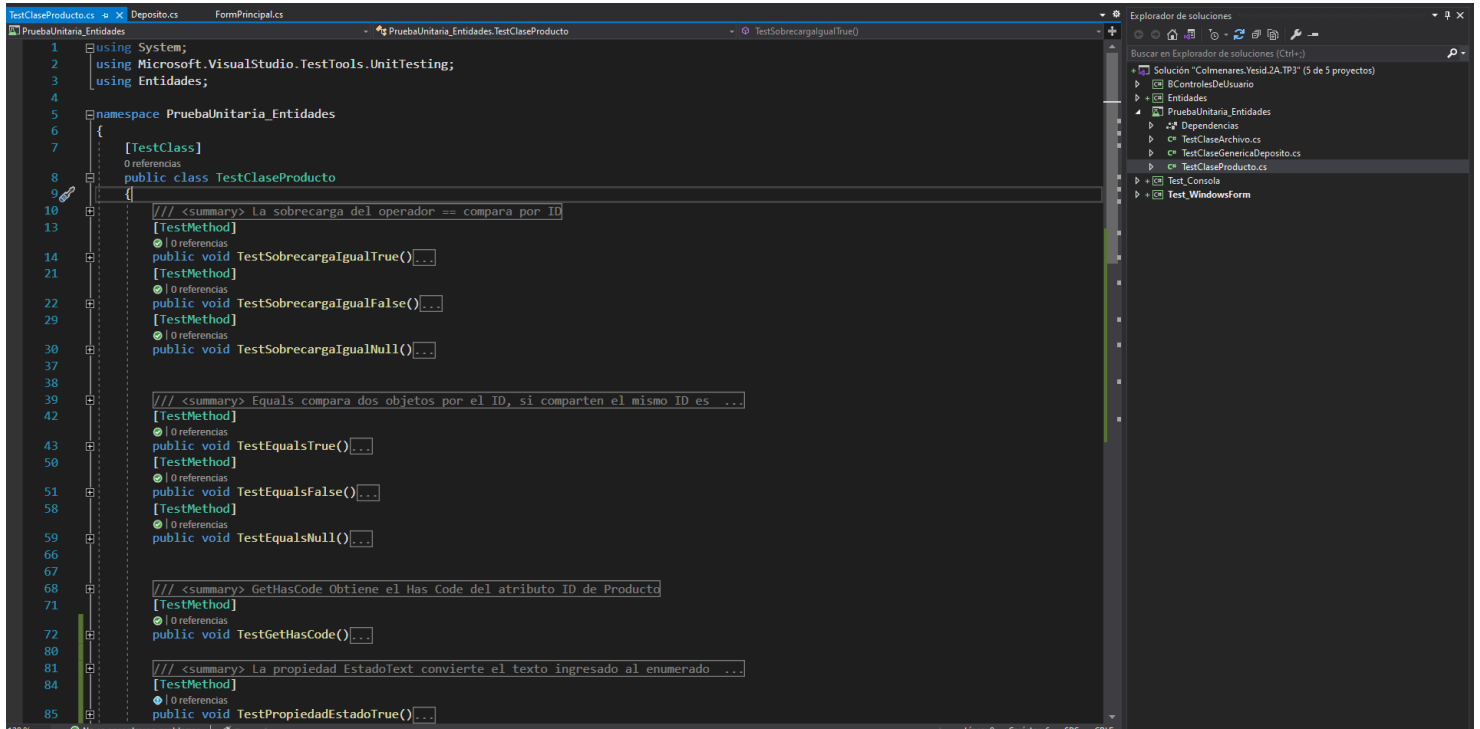
Si el producto a agregar es repetido capturo la excepción y muestro un MessageBox con el mensaje de error.

```
/// <summary> Agrega un producto al deposito y se guarda en el archivo txt
1 referencia | 0 cambios | 0 autores, 0 cambios
private void btnAgregar_Click(object sender, EventArgs e)
{
    FormEditor editar = new FormEditor("Agregar Producto", this.PlataformaActual);

    if (editar.ShowDialog() == DialogResult.OK)
    {
        try
        {
            this.Deposito.Agregar(editar.Producto);
        }
        catch (DepositoExistException exception)
        {
            MessageBox.Show($"{exception.Message}", "ERROR", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }
}
```

2. Pruebas Unitarias

- Clase **TestClaseArchivo.cs**: Pruebo las funciones de la clase Archivo, que es cargar y guardar archivos txt y xml.
- Clase **TestClaseGenericaDeposito.cs**: Pruebo algunas de las funciones de la clase Depósito.
- Clase **TestClaseProducto.cs**: Pruebo algunas de las funciones de la clase Producto, MercadoLibre y Amazon.



3. Tipos Genéricos

Implemento 4 interfaces genéricas. IGestionDeposito, IControlDeposito, IInformesDeposito, IListaDeposito

```
TestClaseProducto.cs | Deposito.cs | FormPrincipal.cs
Entidades
1 using System;
2 using System.Collections.Generic;
3
4 namespace Entidades
5 {
6     58 referencias | 0 cambios | 0 autores, 0 cambios
7     public class Deposito<T> : IGestionDeposito<T>, IControlDeposito<T>, IInformesDeposito<T>, IListaDeposito<T>
8     where T : Producto
9     {
10         //ATRIBUTOS
11         /// <summary> Lista donde guardaremos los productos
12         private List<T> productos;
13
14
15
16         //CONSTRUCTOR
17         /// <summary> Constructor por defecto que inicializa la lista de productos
18         21 referencias | 15/19 pasando | 0 cambios | 0 autores, 0 cambios
19         public Deposito()...
20
21
22         //METODOS
23         /// <summary> Lanza una excepcion del tipo ArgumentException, si el tipo T p...
24         2 referencias | 0 cambios | 0 autores, 0 cambios
25         private bool IsNotNull(T t)...
```

4. Interfaces

- IGestionDeposito

Gestiona los productos del depósito

```
1 referencia | 0 cambios | 0 autores, 0 cambios
public interface IGestionDeposito<T>
{
    where T : Producto

    /// <summary> Retorna la posicion del producto en el deposito
    5 referencias | 3/3 pasando | 0 cambios | 0 autores, 0 cambios
    public int ObtenerIndex(T t);

    /// <summary> Agrega un producto al deposito
    24 referencias | 12/12 pasando | 0 cambios | 0 autores, 0 cambios
    public void Agregar(T t);

    /// <summary> Elimina un producto del deposito
    7 referencias | 3/3 pasando | 0 cambios | 0 autores, 0 cambios
    public bool Eliminar(T t);

    /// <summary> Verifica si existe el producto en el deposito
    7 referencias | 2/2 pasando | 0 cambios | 0 autores, 0 cambios
    public bool Contiene(T t);

    /// <summary> Reemplaza un producto del deposito
    5 referencias | 2/2 pasando | 0 cambios | 0 autores, 0 cambios
    public bool Reemplazar(T t);

    /// <summary> Limpia el deposito
    4 referencias | 1/1 pasando | 0 cambios | 0 autores, 0 cambios
    public void Limpiar();
}
```

- **IControlDeposito**

Controla que los productos estén activos o inactivos

```
namespace Entidades
{
    1 referencia | - cambios | -autores, - cambios
    public interface IControlDeposito<T> where T : Producto
    {
        /// <summary> Controla que los productos esten activos o inactivos
        4 referencias | - cambios | -autores, - cambios
        public List<T> ControlPausadas();
    }
}
```

- **IInformesDeposito**

Informa sobre los productos que tengo en el depósito

```
1 referencia | 0 cambios | 0 autores, 0 cambios
public interface IInformesDeposito<T> where T : Producto
{
    /// <summary> Genera un informe con el contenido del deposito
    2 referencias | 0 cambios | 0 autores, 0 cambios
    public List<T> InfoListaProductos();

    /// <summary> Genera un informe de los productos sin stock
    3 referencias | 0 cambios | 0 autores, 0 cambios
    public List<T> InfoSinStock();

    /// <summary> Genera un informe de los productos con stock
    2 referencias | 0 cambios | 0 autores, 0 cambios
    public List<T> InfoConStock();

    /// <summary> Genera un informe de los productos con estado activo
    2 referencias | 0 cambios | 0 autores, 0 cambios
    public List<T> InfoEstadoActivo();

    /// <summary> Genera un informe de los productos con estado inactivo
    3 referencias | 0 cambios | 0 autores, 0 cambios
    public List<T> InfoEstadoInactivo();

    /// <summary> Genera un informe de los productos con ningun estado
    2 referencias | 0 cambios | 0 autores, 0 cambios
    public List<T> InfoEstadoNinguno();

    /// <summary> Genera un informe de los productos con condicion nuevo
    3 referencias | 0 cambios | 0 autores, 0 cambios
    public List<T> InfoCondicionNuevo();

    /// <summary> Genera un informe de los productos con condicion usado
    2 referencias | 0 cambios | 0 autores, 0 cambios
    public List<T> InfoCondicionUsado();

    /// <summary> Genera un informe de los productos con condicion reacondicionado
    2 referencias | 0 cambios | 0 autores, 0 cambios
    public List<T> InfoCondicionReacondicionado();

    /// <summary> Genera un informe de los productos no tengan ninguna condicion
    2 referencias | 0 cambios | 0 autores, 0 cambios
    public List<T> InfoCondicionNinguno();
}
```


- **IListaDeposito**

Permite la funcionalidad de agregar listas al deposito

```
public interface IListaDeposito<T> where T : Producto
{
    /// <summary> Agrega una lista de productos al deposito
    public bool AgregarLista(List<T> lista);
}
```

5. Archivos y Serialización

La Clase Archivo es la encargada de leer y guardar archivos txt y serializaciones xml.

- **Método CargarTxt de la clase Archivo**

Recibe como parámetro el path la ruta al archivo y un enumerado que contiene la plataforma a cargar, si es de Mercado Libre o Amazon.

Retorna una lista con los productos cargados del archivo txt

```
/// <summary> Carga un archivo txt de productos
10 referencias | 5/5 pasando | 0 cambios | 0 autores, 0 cambios
public static List<Producto> CargarTxt(string path, EPlataforma plataforma)
{
    List<Producto> listaAux = new List<Producto>();
    string linea;
    string[] array;

    if (path is not null)
    {
        using (StreamReader reader = new StreamReader(path, Encoding.UTF8))
        {
            while (!reader.EndOfStream)
            {
                linea = reader.ReadLine();
                array = linea.Split(',');
                float.TryParse(array[2], out float precio);
                int.TryParse(array[3], out int cantidad);

                if (plataforma == EPlataforma.MercadoLibre)
                {
                    listaAux.Add(new MercadoLibre(array[0], array[1], precio, cantidad, array[4], array[5], array[6]));
                }
                else if (plataforma == EPlataforma.Amazon)
                {
                    listaAux.Add(new Amazon(array[0], array[1], precio, cantidad, array[4], array[5], array[6]));
                }
            }
        }
    }
    else
    {
        throw new ArgumentNullException("Archivo", "La ruta al archivo txt es null");
    }

    return listaAux;
}
```

- **Método GuardarTxt de la clase Archivo**

Recibe como parámetro el path la ruta al archivo, la lista de productos y la plataforma del producto a guardar.

Retorna un true si se guardo correctamente

```
/// <summary> Guarda una lista de productos en un archivo txt
6 referencias | 0/2 pasando | 0 cambios | 0 autores, 0 cambios
public static bool GuardarTxt(string path, List<Producto> lista, EPlataforma plataforma)
{
    bool valor = false;

    if (path is not null)
    {
        using (StreamWriter writer = new StreamWriter(path, false, Encoding.UTF8))
        {
            foreach (Producto item in lista)
            {
                if (item.GetType().Name == plataforma.ToString())
                {
                    writer.WriteLine($"{item.Id},{item.Nombre},{item.Precio},{item.Cantidad},{item.Condicion}," +
                        $"{item.Estado}, {item.TipoPublicacion}");
                }
            }
            valor = true;
        }
    }
    else
    {
        throw new ArgumentNullException("Archivo", "La ruta al archivo txt es null");
    }

    return valor;
}
```

- **Método GuardarXml de la clase Archivo**

Recibe como parámetro el path la ruta al archivo, la lista de productos y el typeof del tipo a guardar, en este caso sería el typeof de la lista de productos.

Retorna un true si se guardo correctamente

```
/// <summary> Guarda y realiza una serializacion de una lista de productos
3 referencias | 0/1 pasando | 0 cambios | 0 autores, 0 cambios
public static bool GuardarXml(string path, List<Producto> lista, Type type)
{
    bool valor = false;

    if (path is not null)
    {
        using (XmlTextWriter writer = new XmlTextWriter(path, Encoding.UTF8))
        {
            XmlSerializer ser = new XmlSerializer(type);

            ser.Serialize(writer, lista);
            valor = true;
        }
    }
    else
    {
        throw new ArgumentNullException("Archivo", "La ruta al archivo xml es null");
    }

    return valor;
}
```

- **Método CargarXml de la clase Archivo**

Recibe como parámetro el path la ruta al archivo y typeof del tipo a retornar, en este caso sería de la lista de productos

Retorna una lista con los productos cargados del archivo serializado xml

```
/// <summary> Cargar una serializacion
3 referencias | 0/1 pasando | 0 cambios | 0 autores, 0 cambios
public static List<Producto> CargarXml(string path, Type type)
{
    List<Producto> lista;

    if (path is not null)
    {
        using (XmlTextReader reader = new XmlTextReader(path))
        {
            XmlSerializer ser = new XmlSerializer(type);

            lista = (List<Producto>)ser.Deserialize(reader);
        }
    }
    else
    {
        throw new ArgumentNullException("Archivo", "La ruta al archivo txt es null");
    }

    return lista;
}
```

Interfaz gráfica - Windows Form

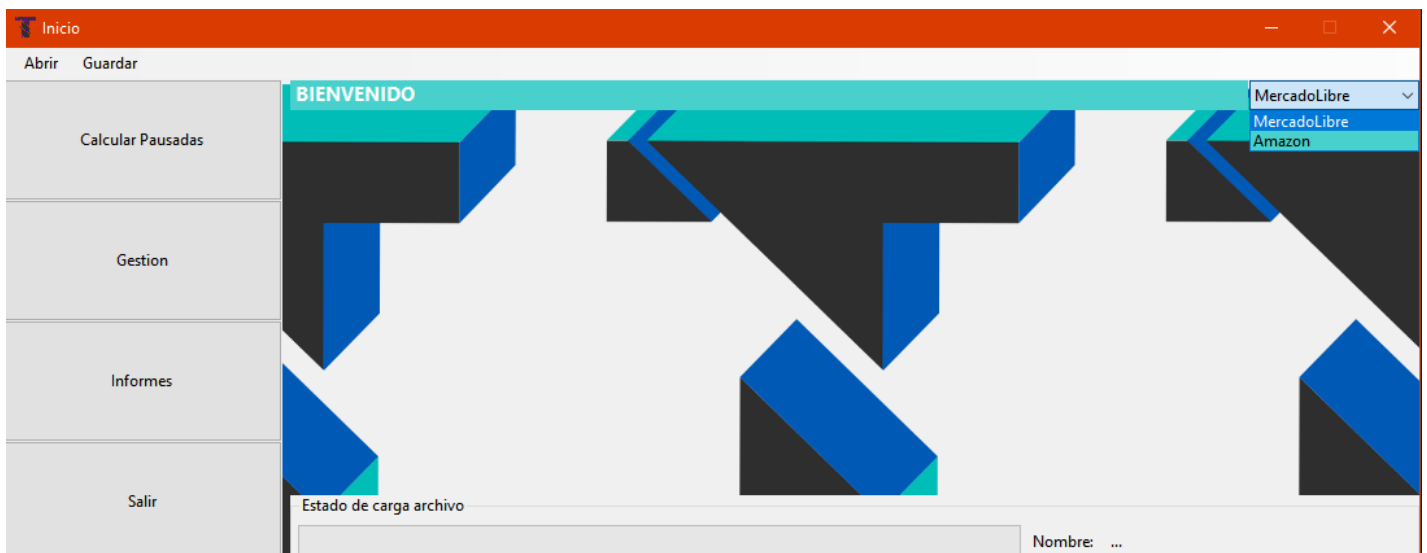
1. Abrir y Guardar

Con estas dos opciones puedes agregar y guardar archivos TXT que tengan productos de Mercado libre, Amazon o una serialización XML.



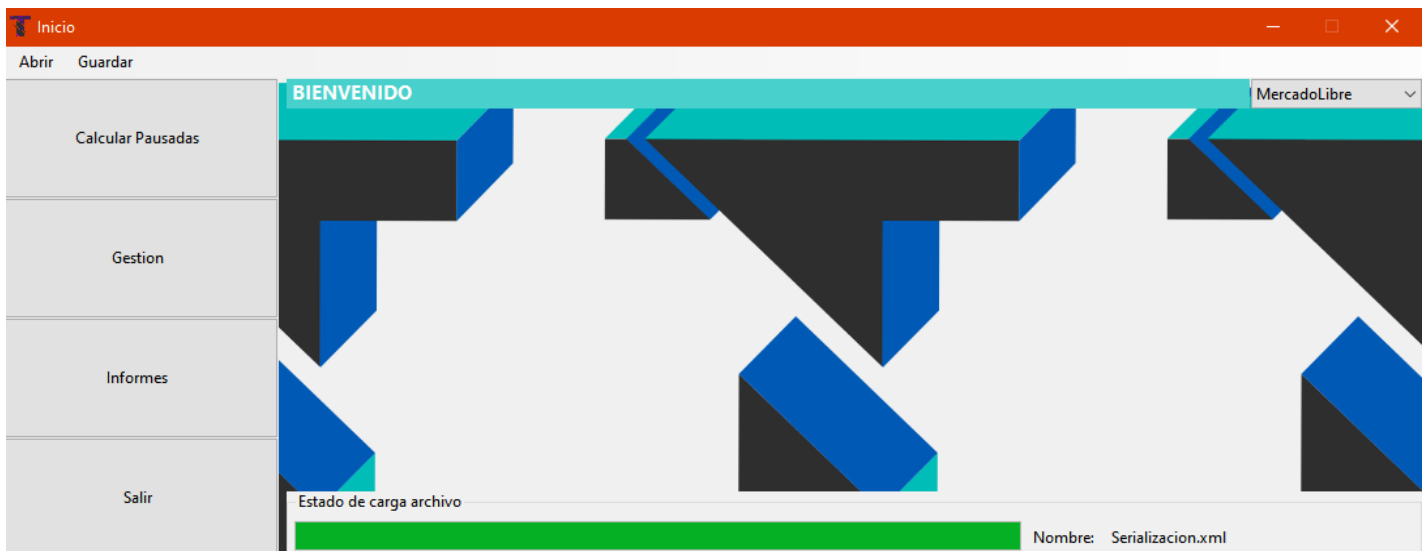
2. Combo Box Tipo de plataforma

Tiene dos opciones, la primera gestionar archivos de Mercadolibre o gestionar los archivos de Amazon que están en el depósito actual.



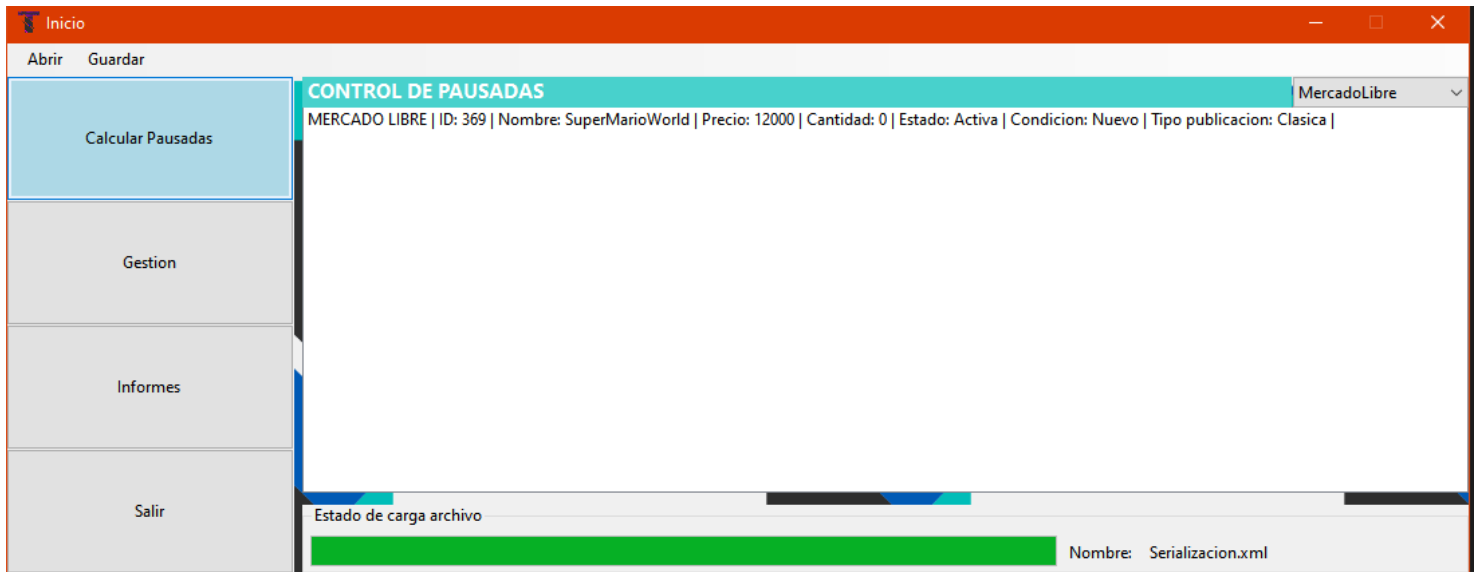
3. ProgressBar que indica el estado de carga del archivo

Si se cargó correctamente se llena la barra, y al lado de la barra se pone el nombre del archivo que se cargó.

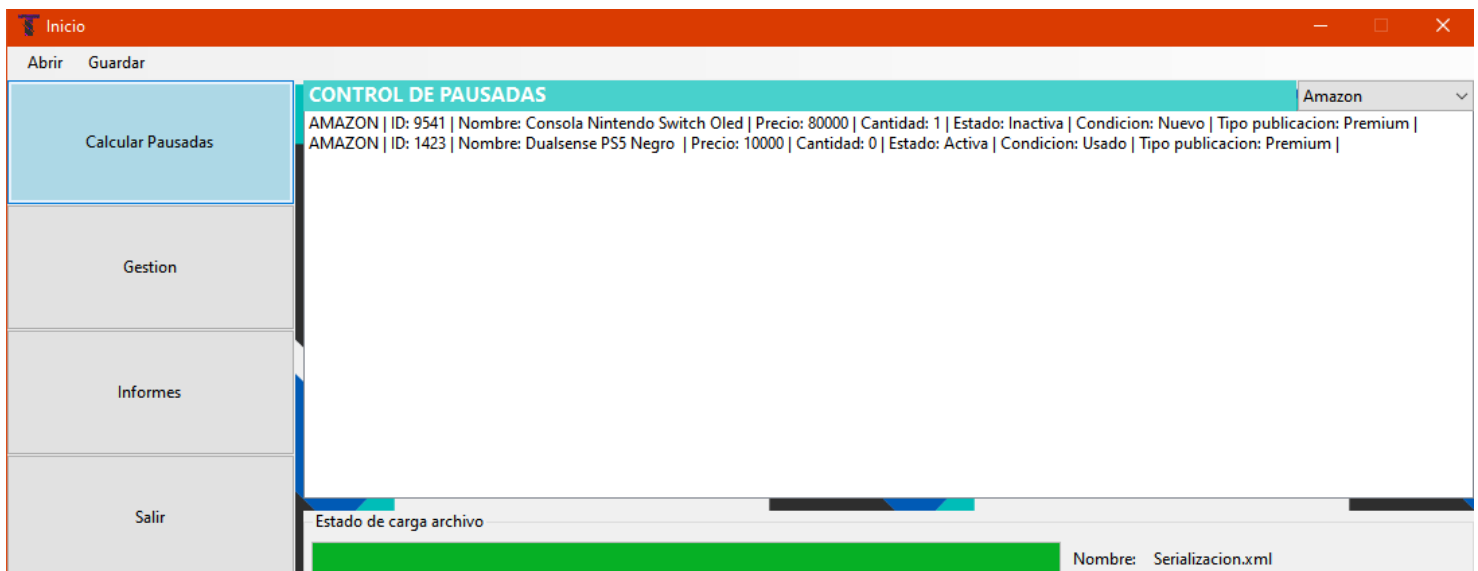


4. Botón Calcular Pausadas

Realiza el control de pausadas de los productos de Mercado Libre en este caso por que el combo box de plataforma esta en Mercado Libre



Si por el contrario establecemos la plataforma en Amazon, calculará las pausadas de los productos de Amazon.



5. Botón Gestion

Tiene un panel que contiene 4 botones, agregar, modificar, eliminar y limpiar depósito



5.1 Botón Agregar

Agrega un producto al depósito, otra vez dependiendo en qué plataforma está, por ejemplo, ahora está en Amazon, entonces agrega el producto del tipo Amazon al depósito.

El ID debe ser un valor numérico menor a 11 dígitos, es obligatorio ingresar el id y el nombre por lo menos para poder agregar un producto al depósito.

The screenshot shows a window titled 'Editar' with a sub-header 'Agregar Producto'. The form contains the following fields:

- ID: A text input field.
- Nombre: A text input field.
- Precio: A numeric input field with a value of 0,00 and a spinner control.
- Cantidad: A numeric input field with a value of 0 and a spinner control.
- Condicion: A dropdown menu with 'Nuevo' selected.
- Estado: A dropdown menu with 'Activa' selected.
- Tipo Publicacion: A dropdown menu with 'Gratuita' selected.
- A 'Guardar' button at the bottom.

5.2 Botón Modificar

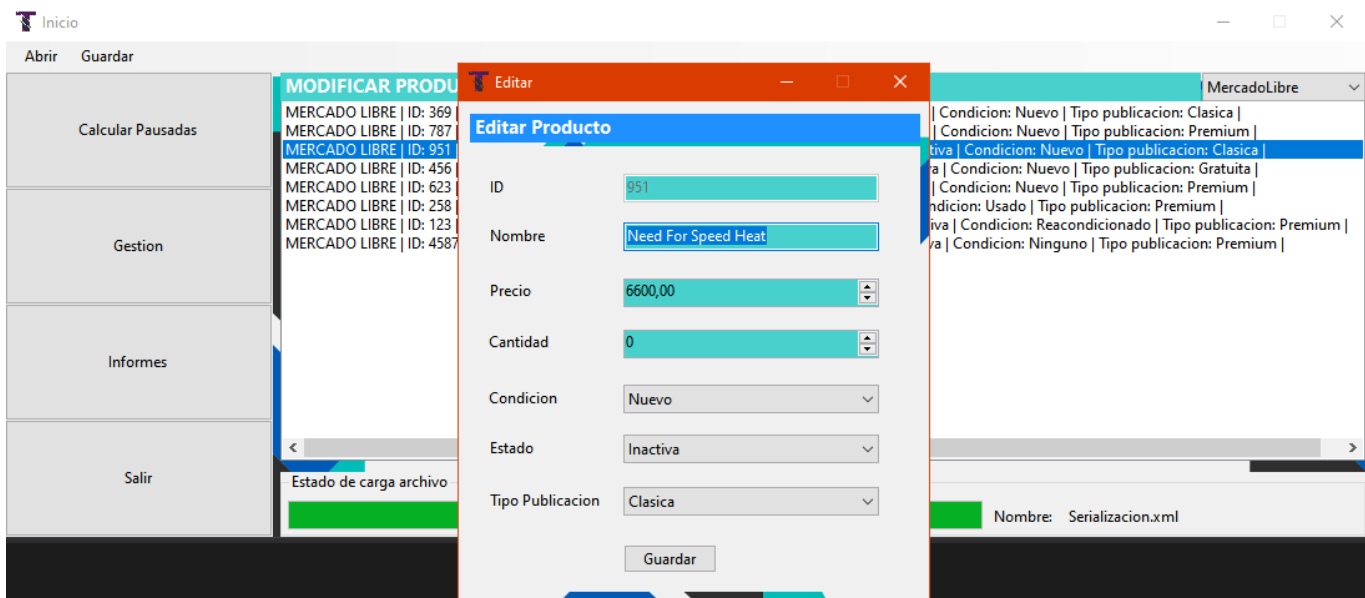
Genera la lista de productos que hay en el depósito y los muestra en un list TextBox

The screenshot shows a window titled 'Inicio' with a sidebar menu on the left and a main content area. The sidebar menu includes 'Abrir', 'Guardar', 'Calcular Pausadas', 'Gestion', 'Informes', and 'Salir'. The main content area has a header 'MODIFICAR PRODUCTOS' and a dropdown menu for 'Amazon'. Below the header is a list of products with the following details:

ID	Nombre	Precio	Cantidad	Estado	Condicion	Tipo publicacion
154487	Consola Xbox Series X	100000	2	Activa	Nuevo	Gratuita
4564	Consola PS5 Fisica	135000	5	Activa	Nuevo	Gratuita
9541	Consola Nintendo Switch Oled	80000	1	Inactiva	Nuevo	Premium
6243	Joystick Xbox Series Rojo	12500	7	Activa	Nuevo	Gratuita
59875	Headset Pulse 3D Sony PS5	21000	10	Activa	Reacondicionado	Clasica
1423	Dualsense PS5 Negro	10000	0	Activa	Usado	Premium
598745	Mouse HyperX Dark Wired	6500	5	Activa	Ninguno	Gratuita

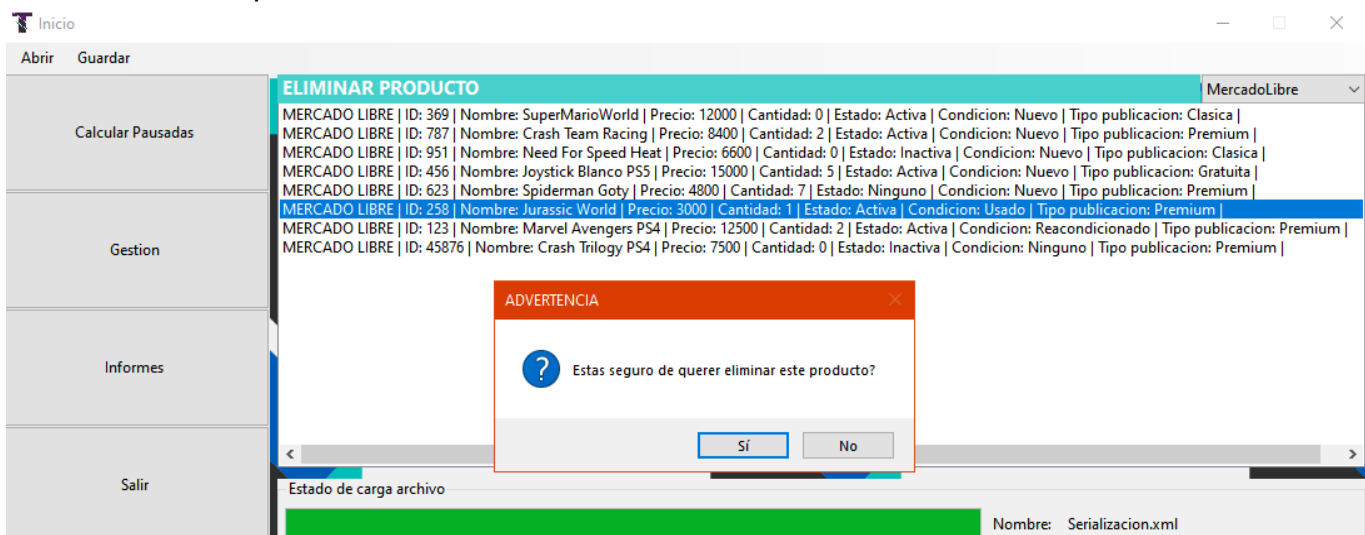
At the bottom of the window, there is a status bar with the text 'Estado de carga archivo' and a green progress bar. The file name 'Nombre: Serializacion.xml' is displayed on the right.

Si seleccionamos con el mouse un elemento se abre una ventana para poder corregir y/o modificar los datos del producto



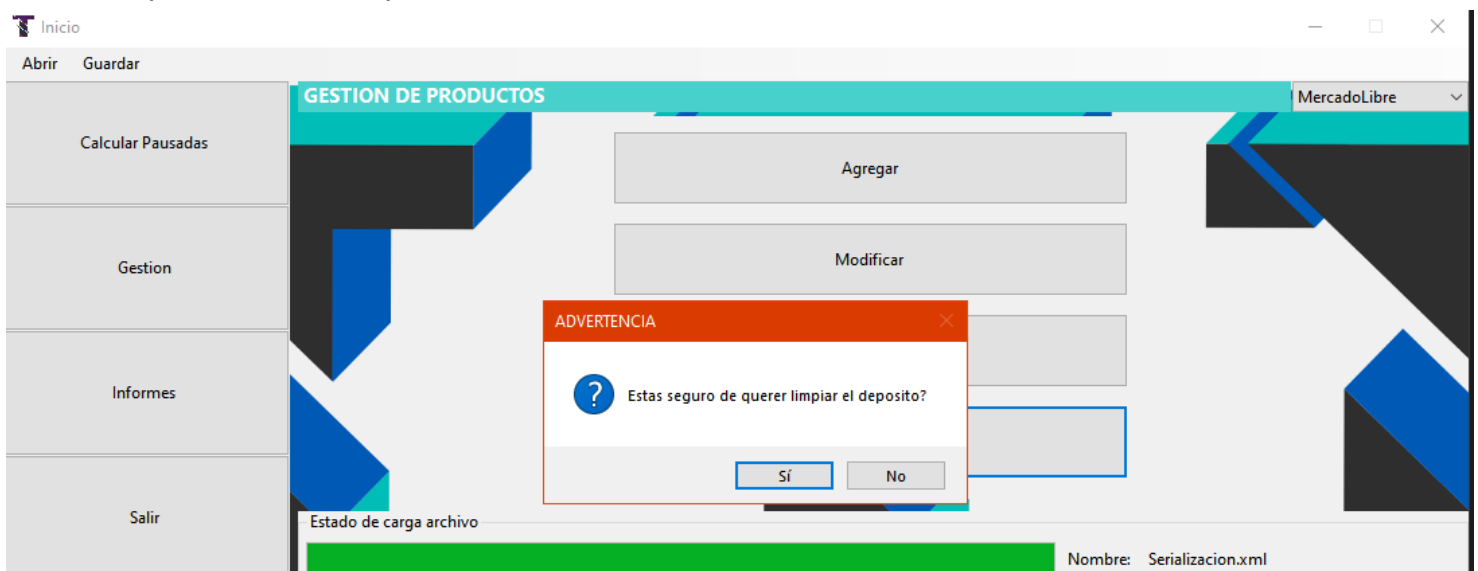
5.3 Botón Eliminar

Genera la lista de productos que hay en el depósito y los muestra en un list TextBox, si realizamos clic en alguno de los productos, nos salta una ventana confirmando que se quiere eliminar el producto



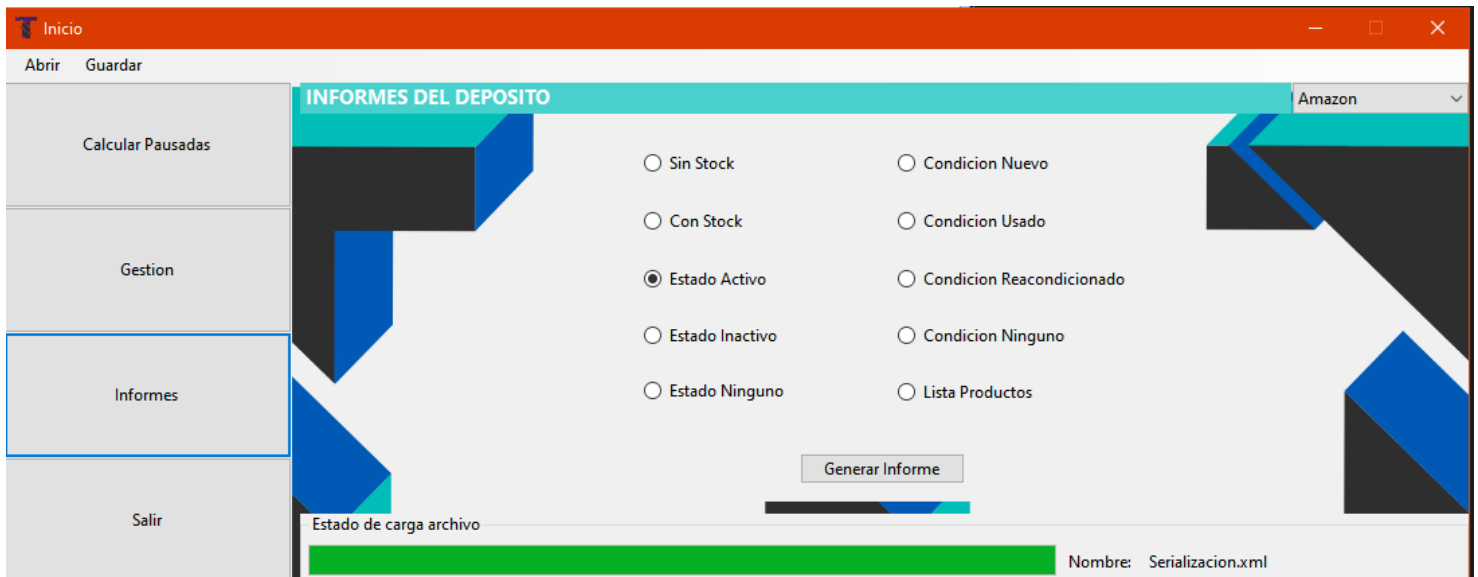
5.4 Botón Limpiar Deposito

Elimina todos los productos del depósito, salta una ventana confirmando si se quiere eliminar los productos del depósito.

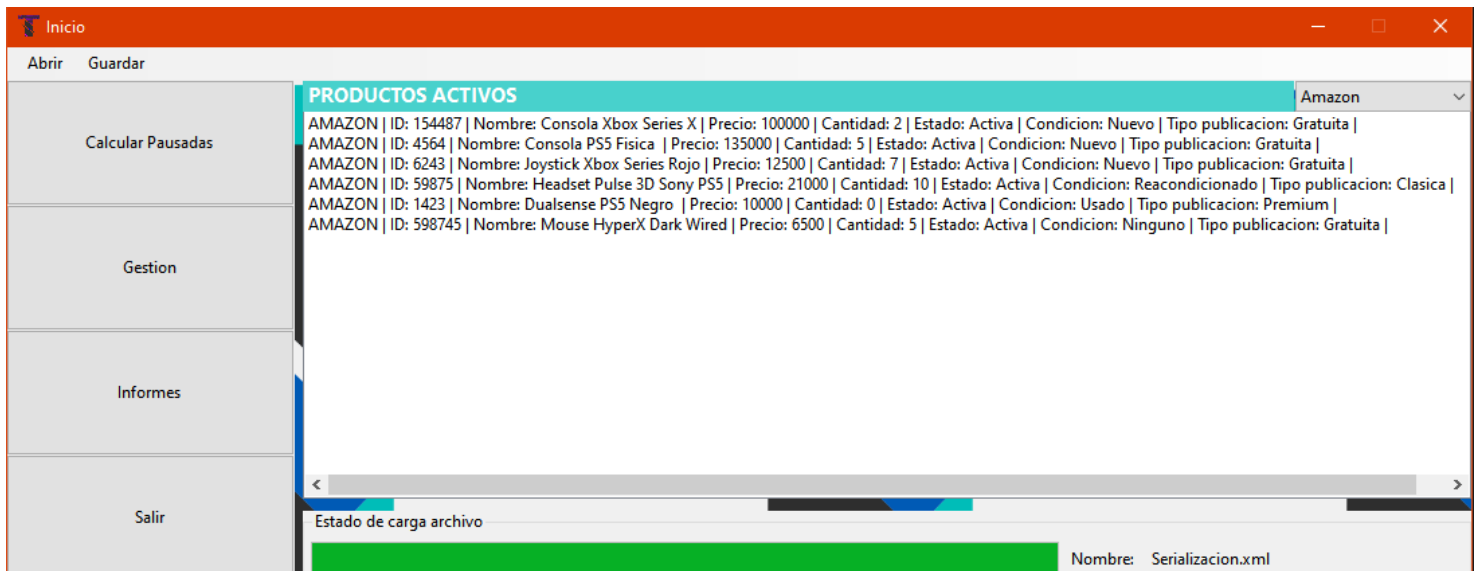


6. Botón Informes

Tiene un panel que contiene varios radiobutton, son los distintos informes que genera del depósito



Luego que seleccionemos la opción y le demos en el botón Generar Informe, nos lleva a un listBox que contiene los productos con ese filtro que seleccione antes.



Si cambiamos la plataforma y seleccionamos Mercado Libre, nos mostrará el filtrado solo para los productos de Mercado Libre

