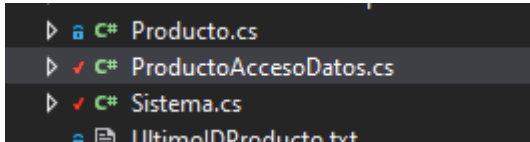


Flujo del programa:

El kiosco utiliza completamente SQL para recibir, modificar y eliminar información sobre los productos. Los registros de las ventas los realiza mediante Archivos, donde se guarda la lista de todas las ventas efectuadas. También se informa la cantidad de dinero recaudado en el día, solo se utiliza efectivo, por lo tanto al llegar a una cifra elevada da un aviso al usuario recomendado realizar un retiro del mismo.

Uso de los temas:

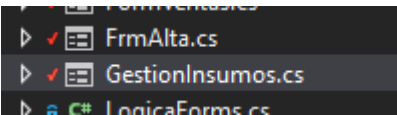
SQL:



Todo el uso y la conexión con la base de datos SQL se encuentra en la clase ProductoAccesoDatos

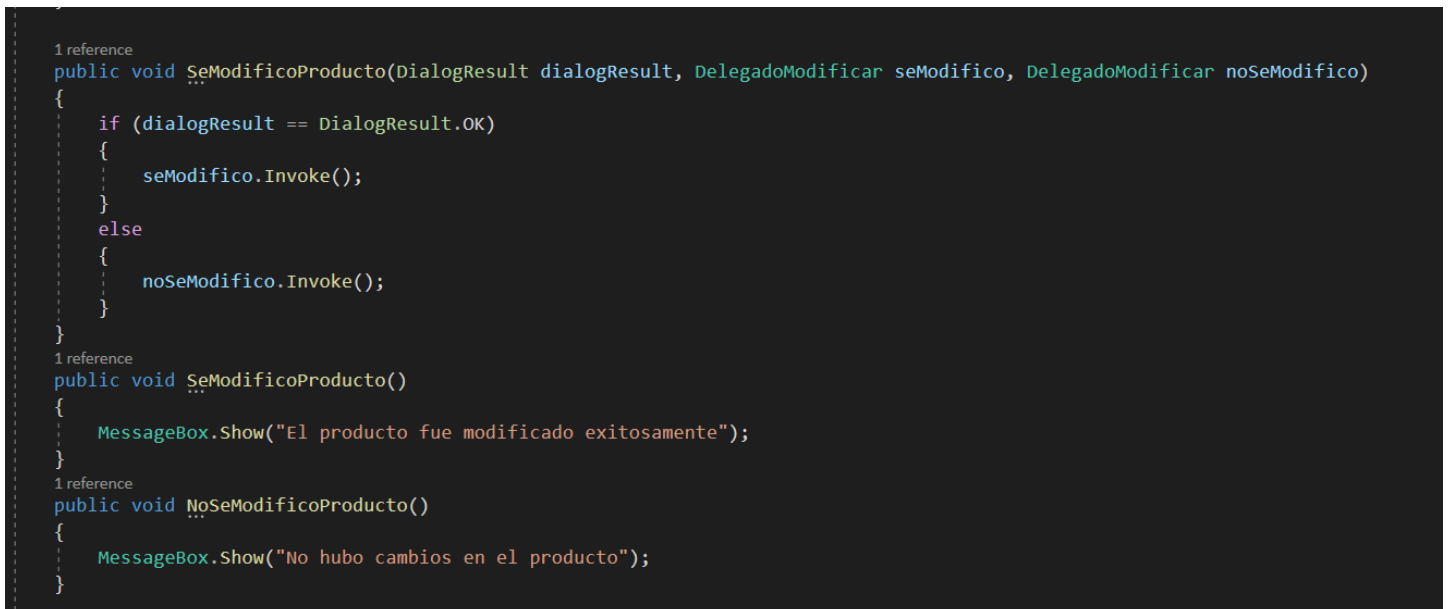
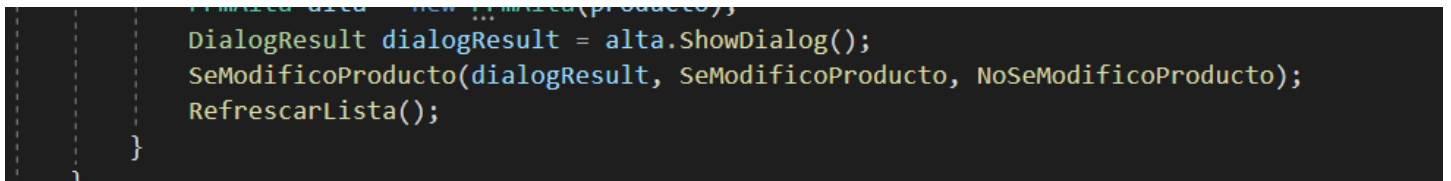
En ella se realizan las tareas de devolver una lista completa de productos, guardar un nuevo producto, eliminar un producto o modificar los datos de un producto.

Delegados:

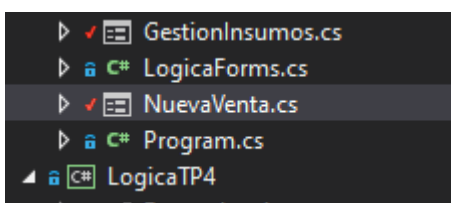


El uso de delegados se aplica dentro del form GestionInsumos.

El cual advierte al usuario sobre el cambio realizado en un producto según el DialogResult del formulario de Modificaciones recibiendo 2 métodos diferentes.



Task:



Se utiliza dentro del Form NuevaVenta, en el método Load, para hacer la tarea de la lectura de la base de datos asincrónica.

```

1 reference
private async void NuevaVenta_Load(object sender, EventArgs e)
{
    try
    {
        await Task.Run(() => { CargarLista(); });
        ListarProductosParaLaVenta();
        lblSaldoTotal.Text = "Saldo: $0";
    }
    catch (Exception ex)
    {
        LogicaForms.MostrarExcepciones(ex);
    }
}

```

Eventos:

```

10 references
public class Sistema
{
    public static event DelegadoVerificarSaldo saldoAlcanzado;
    public static List<Venta> listaVentas;
    private static float saldoTotalEnCaja;
}

```

▶ C# Producto.cs
 ▶ ✓ C# ProductoAccesoDatos.cs
 ▶ ✓ C# Sistema.cs
 ▶ UltimoIDProducto.txt
 ▶ UltimoIDVenta.txt
 ▶ ✓ C# Venta.cs

El delegado está declarado dentro de la clase Sistema y es utilizado dentro del Form FormPrincipal

```

ProbarConexionABaseDeDatos();
Sistema.saldoAlcanzado += MostrarQueHayMuchaPlataEnLaCaja;
}
catch (Exception ex)
{
    LogicaForms.MostrarExcepciones(ex);
}

```

▶ Dependencies
 ▶ + FormHacerRetiro.cs
 ▶ ✓ FormPrincipal.cs
 ▶ ✓ FormVentas.cs
 ▶ ✓ FrmAlta.cs
 ▶ ✓ GestionInsumos.cs

```

1 reference
public void MostrarQueHayMuchaPlataEnLaCaja()
{
    MessageBox.Show("Hay mucho dinero en caja, haga un retiro por favor");
}
1 reference

```

Metodos de extension:

Para el método de extensión se desarrolló la clase IntExtension la cual extiende la clase int.

0 references

```
public static class IntExtension
```

```
{
```

1 reference

```
public static Venta VentaPorId(this int id, List<Venta> lista)
```

```
{
```

```
    foreach (Venta item in lista)
```

```
    {
```

```
        if (item.IdVenta == id)
```


```
            return item;
```


```
    }
```


```
    return null;
```

```
}
```


```
}
```


▷  **C#** DatosInvalidoException.cs


▷  **C#** GestorDeArchivoTexto.cs


▷  **C#** GestorSerializacion.cs


▷  **C#** IId.cs


▷ +  **C#** IntExtension.cs


 listaProductos.xml


 listaVentas.xml

▷  **C#** ParametrosVaciosException

▷  **C#** Producto.cs

▷  **C#** ProductoAccesoDatos.cs

▷  **C#** Sistema.cs

 UltimoIDProducto.txt

Properties