****Importante:** en el directorio donde se encuentra la solución se encuentra el archivo script.sql que genera la base de datos y su contenido

Funcionalidades de la aplicación:

-La aplicación permite agregar productos (descripción, precio y stock) y los guarda en la base de datos

-permite crear Ventas. Cada venta poseerá un cliente y una cantidad de ítems sin límite(cada ítem consta de un producto y la cantidad). La aplicación guarda la venta en la base de datos para lo cual utiliza una tabla "ventas" y una tabla "itemsventa" relacionando los ítems con la venta correspondiente (relación Many-To-One).

Formulario:

En el formulario FormAgregarProducto se encontrará una sección "Agregar Producto" donde se podrán ingresar datos para la creación de un nuevo producto. A la derecha hay un dataGrid que muestra los datos de los productos disponibles.

Debajo se encuentra un boto Testear Evento cuya funcionalidad de detalla más abajo en el apartado de la clase 24

Debajo se encuentran los botones Serializar y Deserializar. El primero serializa en un archivo Xml el listado de productos y el segundo deserializa y muestra los productos en el dataGrid de la derecha.

Detalle de los conceptos plasmados por clase:

15-Excepciones: Se crearon dos clases: **DatabaseException**: lanzada por la clase **DB**. **ProductoRepetidoException:** lanzada por la clase **SuperMercado** si se trata de crear un producto que ya existe.

16-Unit Test: Se creó un proyecto llamado **UnitTests** con dos métodos que prueban la funcionalidad "Crear Producto" y la funcionalidad "Serializar y Deserializar" en este caso con un objeto de tipo Venta

17-Generics: La clase **Logger** tiene un método genérico (RegistrarEvento línea 22)que genera una entrada en un archivo de log con fecha, hora y datos del objeto. La clase **Serializador** es una clase genérica que serializa y deserializa objetos.

18-Interfaces: la Interfaz **IVerbose** incluye una propiedad y un método que deben implementar las clases que la utilicen.

19-Archivos y Serializacion: la clase **Logger** genera entradas en archivos de texto (1 distinto por dia). La clase **Serializador** genera archivos Xml para ventas y productos. Todos los archivos mencionados se encuentran en el directorio "Seif.Mariano.2D.TP4\Test\bin\Debug".

20-No se dio tema

21-SQL: En la clase **DB** se encuentran todas las querys realizadas a la base de datos (líneas: 44, 47, 67, 91, etc).

22-Base de Datos: La clase **DB** contiene todo el manejo de base de datos. Los datos para la conexión y los métodos para registro y adquisición de datos.

23-Threads: El formulario **FormAgregarProducto** crea dos hilos nuevos. El **primero** se encarga de refrescar el datagrid ubicado en la parte superior del formulario logrando que no se bloquee el formulario y siga siendo funcional. El **segundo** genera ventas (con un límite de 20) con datos aleatorios traídos de la base de datos que luego serán registradas en la base de datos.

24-Eventos y Delegados: los objetos de tipo Producto poseen un evento llamado **outOfStockEvent** que se dispara cuando el atributo "stock" del objeto es seteado en cero. Para ello se creó un delegado llamado **NotifierDelegate** (clase Producto línea 10) que aceptará métodos void con un parámetro string y un parámetro del tipo Producto.

En el formulario **FormAgregarProducto** hay un botón "Testear Evento" que seleccionará un producto al azar y lo forzará a lanzar el evento. La respuesta se verá en la consola.

25-Metodos de extensión: La clase **DateTimeExtend** extiende la clase **DateTime** con dos métodos, **LogFormatedDateTime** y **LogFileName**. El primero devuelve una cadena que será utilizada en cada entrada que se genere en los archivos de log. El segundo devuelve una cadena formateada para utilizar la creación del nombre de los archivos de log.