Guía técnica

**Alejandro Mihai y Tomás Praga**

# 

# 

# 

# 

# 

1.-Definición y planteamiento del problema 3

2.-Diseño modular 4

3.-Diseño de programas 5

3.-Juego de pruebas 6

# **1.-Definición, planteamiento del problema y solución**

El objetivo de esta aplicación es ser una herramienta para el vendedor en tienda a la hora de gestionar una venta a un cliente, también conocida como TPV (Terminal de Punto de Venta).

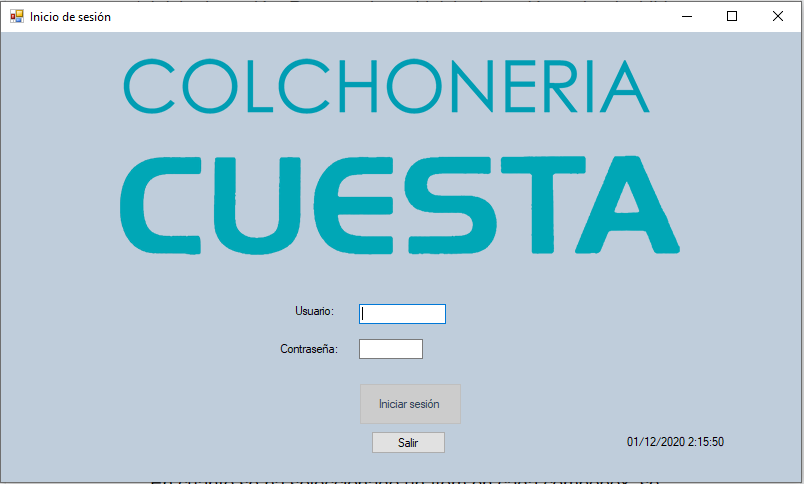
A este fin, la aplicación contiene manejo de datos de productos, de usuarios que usen la aplicación y de ventas diarias y cobros.

La empresa que nos ha encargado la TPV es la “Colchonería CUESTA” y quiere que la TPV nos permita gestionar sus productos, los precios de venta, los usuarios y poder realizar ventas (con su correspondiente ticket) guardando el valor recaudado por día.

## 1.1 Solución

### Inicio de sesión: Para resolver el inicio de sesión se ha decidido usar el evento que comprueba el campo text de los textbox cuando se introducen datos, con métodos que se encuentran en una librería específica para ello.

Por lo tanto si el usuario introduce datos en el campo de usuario o contraseña que no son correctos, se le avisará y se limpiarán los campos tanto para facilitarle el trabajo al usuario como para que no pueda cometer errores.



### Gestión de ventas: Se ha decidido usar tres objetos combobox, un listbox y dos botones.

-Visualización y selección de productos: El combobox que muestra las categorías de los productos, al ser seleccionado un ítem, permite desplegar el nombre del producto, y este a su vez, una vez elegido, permite que se elija el del tamaño.

Si no hay una categoría elegida, los demás combobox no permiten elegir items. Es un efecto cascada.

En cuanto se ha seleccionado un ítem en cada combobox, se puede proseguir a añadir el producto al carrito.

-Añadir al carrito: La forma en que se añade un item al Listbox (el carrito) es la siguiente: se toman los valores de la estructura producto (categoría, nombre, etc.) y se guardan y concatenan en un string.

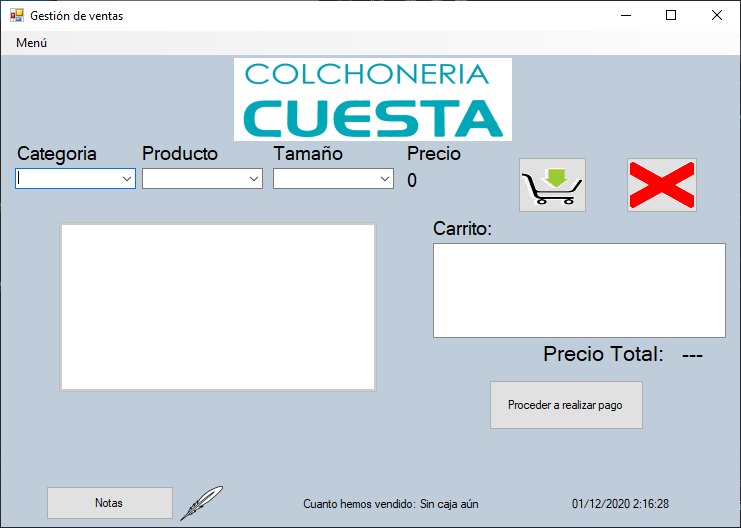
De tal modo que primero se guarda en el string la categoría, luego se concatena el nombre, luego el tamaño y luego el precio.

Una vez concatenados todos los campos, se añade el item al listbox, y así se puede ver en pantalla.

-Visualización en pantalla del precio total y botón eliminar del carrito: Como necesitaremos el precio de cada producto para el cobro, al momento de concatenar estos string también guardamos el precio de cada producto en un ArrayList de forma paralela. De forma tal que el primer item del listbox tiene su precio en la primera posición del array list.

Si se elimina del carrito el ítem (en el listbox) número 3, también se eliminará su precio del ArrayList.

El precio total que se ve en pantalla en un label es simplemente una suma de todos los elementos del ArrayList, que es efectivamente el carrito de la compra.

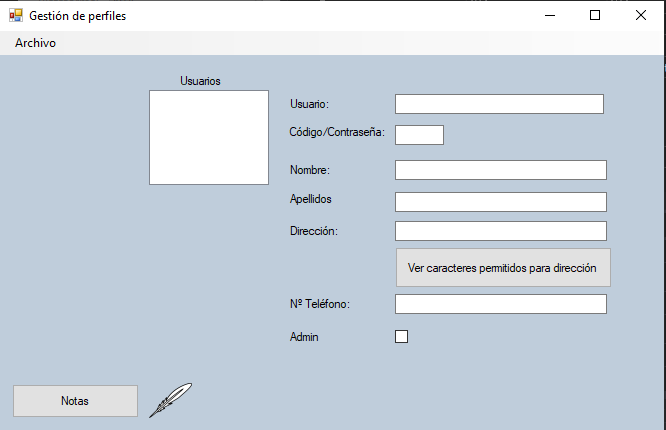


### Gestión de usuarios: Para la gestión de usuarios se ha decidido usar un fichero de acceso secuencial. Los motivos son dos: los campos pueden tener cualquier longitud, al contrario que en los ficheros de acceso directo donde los registros deben ser de un tamaño fijo. El otro motivo es por la variedad del ejercicio, dado que la gestión de productos/precios se realiza con un fichero de acceso aleatorio.

Para la gestión de usuarios/perfiles el usuario tiene a su disposición un menuStrip (barra de herramientas) donde puede limpiar todos los campos (textbox), abrir el fichero de usuarios, guardar datos nuevos, eliminar un perfil, o volver a la pantalla de ventas.

La pantalla tiene varios campos (textbox) con restricciones que se validan y que el usuario debe rellenar (todos) para crear un nuevo usuario.

Para saber si un perfil es de administrador o no, se utiliza un checkbox. Si esta “checked” el perfil es de administrador y tendrá privilegios para gestionar productos y usuarios. Si no, se trata de un empleado que solo usará la aplicación para vender.

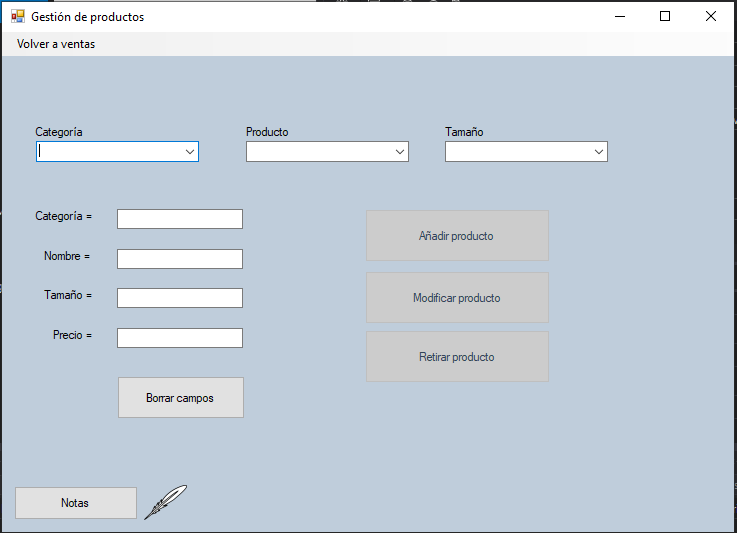


### Gestión de precios/productos: Para la gestión de precios/productos se ha usado un fichero de acceso aleatorio.

Se permite mediante botones crear un producto nuevo, modificar uno existente, y borrar.

Se utilizan comboBox para, del mismo modo que en la pantalla de ventas, a modo de cascada, se vayan eligiendo categoría, luego nombre y finalmente tamaño (al elegir este último el textBox precio se rellena solo).

Cuando se desea crear un producto nuevo, se itera por el fichero de acceso aleatorio primero para comprobar que los registros no coinciden con el que está introducido por el usuario. De coincidir, se le informa al usuario de que el producto ya existe, y se limpian los campos.



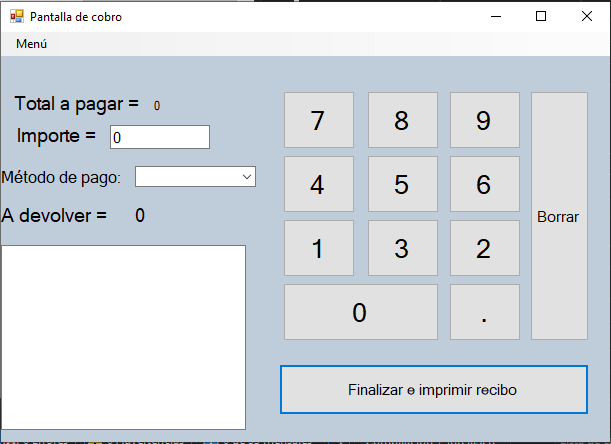
1. Cobro: El cobro se realiza mediante el uso de un teclado que se encuentra en pantalla (pudiendo usarlo con el ratón o el teclado).

La cesta (datos del listBox del formulario ventas) y el precio total se traspasan a este formulario y son utilizados para la creación de un ticket.

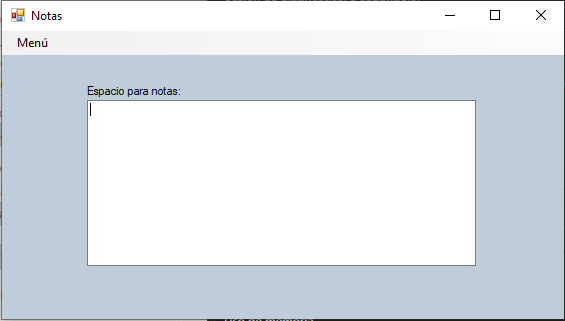
Antes de crear el ticket el usuario debe elegir método de pago (seleccionado a través de un comboBox) e introducir el importe (en un textBox) a pagar (solo al pagar con efectivo!).

Hay habilitado un label que muestra la cantidad que debe devolver el vendedor al cliente.

Al pulsar el botón de finalizar e imprimir recibo, se produce una factura con los datos de la compra



### Funcionalidad notas: Se ha añadido un botón en cada pantalla, donde el usuario puede acceder a un fichero donde puede escribir las notas que desee. Estas notas se guardan en un fichero de texto que no tiene relación con el resto de ficheros donde se guarda información (el fichero de acceso, el de errores, etc.).



# **2.-Diseño de programas**

Aquí se introduce la descripción y diagramas de los siguiente elementos usados: variables, arrays, bibliotecas, ficheros, funciones y las condiciones para la solución.

El contenido se describe según el orden de aparición en la aplicación, y de arriba a abajo si dos elementos coinciden. Es decir, hay que usar el ámbito y no el orden alfabético de los nombres de las variables para saber dónde encontrarlas.

## A)Tabla 1: Variables

### 2.1 Pantalla Inicio

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nombre | Tipo | Valor inicial | Comentarios | Ámbito |
| validarUsurio | clase Validacion(biblioteca) | Null. | Es una instancia de una clase en la biblioteca libValidacionDatos. Se usará para usar los métodos de la clase Validacion. | -Local en el procedimiento txtUsuario.TextChanged del formulario PantallaInicio.  -Local en el procedimiento txtContrasena.TextChanged del formulario PantallaInicio. |
| valido | boolean | False | Una variable local usada para guardar el resultado(true o false) que devuelve llamar al método para validar el usuario. | Local en el procedimiento txtUsuario.TextChanged del formulario PantallaInicio.  -Local en el procedimiento txtContrasena.TextChanged del formulario PantallaInicio. |
| validarAcceso | clase Validacion | Null. | Variable usada para tener acceso a los métodos de la clase Validacion en la biblioteca libValidacionDatos. | -Local en el procedimientobtnIniciarSesion.Click del formulario PantallaInicio. |
| resultado | Integer | Null | Variable usada para guardar el valor que devuelve el método comprobarDatosSecuncial | local a procedimiento btnIniciarSesion.Click en formulario PantallaInicio |
| NO SE USA BORRARcoincidencia | String | “” | --- | local a procedimiento btnIniciarSesion.Click en formulario PantallaInicio |

### 2.2 Pantalla Ventas

## 

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nombre | Tipo | Valor inicial | Comentarios | Ámbito |
| precioTotal | Single | 0 | Variable usada para que el label del precio total que verá el usuario en pantalla tome el valor del precio total recaudado ese día. | Local a clase pantallaVentas. |
| arrayPreciosLista | ArrayList | 0 | Lista usada para guardar los precios de cada item guardado en el ListBox que representa el carrito de la compra.  -Usada en procedimiento btnAnadirProd.click | Local a clase pantallaVentas. |
| aux | String | “” | Variable donde se guardan datos tras una lectura, usada para comprobar que no hay duplicidades en los combobox. | 1) Local a procedimiento load de PantallaVentas  2)Local a procedimiento comboCategoria.SelectedIndexChanged en formulario PantallaVentas  3)Local aprocedimientocomboNombreProd.SelectedIndexChanged en formulario Pantalla Ventas |
| categoria | String | “” | Variable donde se guardan datos tras una lectura, usada para comprobar que no hay duplicidades en los combobox. | 1) Local a procedimiento load de PantallaVentas  2)Local a procedimiento comboCategoria.SelectedIndexChanged en formulario PantallaVentas  3)Local aprocedimientocomboNombreProd.SelectedIndexChanged en formulario Pantalla Ventas |
| nombre | String | “” | Variable donde se guardan datos tras una lectura, usada para comprobar que no hay duplicidades en los combobox. | 1) Local a procedimiento load de PantallaVentas  2)Local a procedimiento comboCategoria.SelectedIndexChanged en formulario PantallaVentas  3)Local aprocedimientocomboNombreProd.SelectedIndexChanged en formulario Pantalla Ventas |
| tamaño | String | “” | Variable donde se guardan datos tras una lectura, usada para comprobar que no hay duplicidades en los combobox. | 1) Local a procedimiento load de PantallaVentas  2)Local a procedimiento comboCategoria.SelectedIndexChanged en formulario PantallaVentas  3)Local aprocedimientocomboNombreProd.SelectedIndexChanged en formulario Pantalla Ventas |
| precio | Single | o | Variable donde se guardan datos tras una lectura, usada para comprobar que no hay duplicidades en los combobox. | 1) Local a procedimiento load de PantallaVentas  2)Local a procedimiento comboCategoria.SelectedIndexChanged en formulario PantallaVentas  3)Local aprocedimientocomboNombreProd.SelectedIndexChanged en formulario Pantalla Ventas |
| numeroRegistros | Integer | Depende. Lo que devuelva el método numeroRegistros(File) | Guarda el número de registros en un archivo de acceso directo. | 1) Local a procedimiento load de PantallaVentas  2)Local a procedimiento comboCategoria.SelectedIndexChanged en formulario PantallaVentas |
| listaCategoria | ArrayList | 0 | Guarda los valores ya introducidos para evitar duplicados. | Local a procedimiento load de PantallaVentas |
| doble | boolean | false | bandera para conocer si el valor es aceptado o no. | 1) Local a procedimiento load de PantallaVentas  2)Local a procedimiento comboCategoria.SelectedIndexChanged en formulario PantallaVentas |
| CREO QUE NO ESTA SIENDO USADOlectura | objeto de clase Lectura de la biblioteca LecuraEscrituraArchivos | Null |  | Local a procedimiento btnAnadirProd.Click en formularo PantallaVentas |
| posicion | integer | 0 |  | Local a procedimiento btnAnadirProd.Click en formularo PantallaVentas |
| categoria | String | El valor del item seleccionado en el combobox. |  | Local a procedimiento btnAnadirProd.Click en formularo PantallaVentas |
| linea | String | “” |  | Local a procedimiento btnAnadirProd.Click en formularo PantallaVentas |
| fichero | String | “Errorlog.txt” | Se guarda el string que luego se usará para indicar la ruta para abrir un archivo para I/O. | Bloque excepción del procedimiento btnAnadirProdClick en formulario PantallaVentas |
| fich | FileStream | (fichero, FileMode.Append, FileAccess.Write) | Flujo para escribir en el fichero de errores. | Bloque excepción del procedimiento btnAnadirProdClick en formulario PantallaVentas |
| rs | StreamWriter | (fich) | Flujo para escribir en el fichero de errores. | Bloque excepción del procedimiento btnAnadirProdClick en formulario PantallaVentas |
| listNombres | ArrayList | 0 | Usado para guardar valores y evitar duplicidades. | Local a procedimiento comboCategoria.SelectedIndexChanged en formulario PantallaVentas |
| lectura | Lectura de biblioteca LecturaEscrituraArchivos | Lectura | Usado para obtener acceso a los métodos de la clase Lectura. | Local a procedimiento comboCategoria.SelectedIndexChanged en formulario PantallaVentas |
| datosAcceso | FileStream | ("cajaDiaria.txt", FileMode.Append, FileAccess.Write) | Usado para escribir en el fichero donde se guarda la recaudación diaria. | 1)Local a procedimiento stripGuardarCaja.Click en formulario PantallaVentas. |
| sw | StreamWriter | datosAcceso | Usado Para guardar datos en el fichero de recaudación diaria. | 1)Local a procedimiento stripGuardarCaja.Click en formulario PantallaVentas. |

### 2.3 Gestión de Perfiles

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nombre | Tipo | Valor inicial | Comentarios | Ámbito |
| validacion | Objeto de la clase Validacion de la librería libValidacionDatos | Validacion | Variable usada para acceder a los métodos de la clase Validación. | Local a procedimientostripGuardar.Click del formulario Gestión Perfiles |
| coincidencia | String | “” | Variable usada para comparar el dato del item seleccionado en el ListBox de usuarios con el valor guardado en esta variable tras la lectura del archivo “usuarios.txt” | 1)Local a procedimientostripGuardar.Click del formulario Gestión Perfiles  2)Local a procedimiento stripEliminar.Click en el formulario Gestión Perfiles |
| libre | boolean | true | Usada para saber si el dato de usuario ya existe. | 1)Local a procedimientostripGuardar.Click del formulario Gestión Perfiles  2)Local a procedimiento stripEliminar.Click en el formulario Gestión Perfiles |
| validarUsuario | Objeto de la claseValidación de la librería libValidacionDatos | Validacion | Usada para acceder a los métodos de la clase Validacion. | 1)Local a procedimiento txtUsuario.TextChanged en formulario Gestión de Perfiles.  2)Local a procedimiento txtNombre.TextChanged del formulario Gestión Perfiles.  3)Local a procedimiento txtApellidos.TextChanged del formulario Gestión Perfiles.  4)Local a procedimiento txtDireccion.TextChanged del formulario Gestión Perfiles. |
| valido | boolean | No se le da. | Recibe el resultado de llamar al método validarUsuario de la librería libValidacionDatos. En base al dato recogido, se sabe si el dato intoducido es válido o no. | 1)Local a procedimiento txtUsuario.TextChanged en formulario Gestión de Perfiles.  2)Local a procedimiento txtContrasena.TextChanged del formulario Gestión de Perfiles.  3)Local a procedimiento txtNombre.TextChanged del formulario Gestión Perfiles.  4)Local a procedimiento txtApellidos.TextChanged del formulario Gestión Perfiles.  5)Local a procedimiento txtDireccion.TextChanged del formulario Gestión Perfiles.  6)Local a procedimiento txtTelefono.TextChanged del formulario Gestionar Perfiles. |
| validarCodigo | Objeto de la clase Validacion de la librería libValidacionDatos | Validacion |  | 1)Local a procedimiento txtContrasena.TextChanged del formulario Gestión de Perfiles.  2)Local a procedimiento txtTelefono.TextChanged del formulario Gestionar Perfiles. |

### 2.4 Gestión Precios/Productos

Alex

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nombre | Tipo | Valor inicial | Comentarios | Ámbito |
| aux | String | "" |  | Utilizado siempre que se escriben/leen datos de un fichero. |
| categoria | String | "" |  | Utilizado siempre que se escriben/leen datos de un fichero. |
| nombre | String | "" |  | Utilizado siempre que se escriben/leen datos de un fichero. |
| tamaño | String | "" |  | Utilizado siempre que se escriben/leen datos de un fichero. |
| precio | Integer | 0 |  | Utilizado siempre que se escriben/leen datos de un fichero. |
| lectura | LecturaEscrituraArchivos |  |  | Método GestionPrecios\_Load |
| numeroRegistros | Integer | lectura.numeroRegistros |  | Método GestionPrecios\_Load |
| listaCategoria | ArrayList |  |  | Método GestionPrecios\_Load |
| doble | Boolean |  |  | Método GestionPrecios\_Load |
| listaNombre | ArrayList |  |  | Método comboCategoria\_SelectedIndexChanged |
| doble | Boolean | True |  | Método comboCategoria\_SelectedIndexChanged |
| lectura | LecturaEscrituraArchivos |  |  | Método comboCategoria\_SelectedIndexChanged |
| numeroRegistros | Integer | lectura.numeroRegistro |  | Método comboCategoria\_SelectedIndexChanged |
| aux | String | comboProducto.SelectedItem |  | Método comboProducto\_SelectedIndexChanged |
| validacion | libValidacionDatos.Validacion |  |  | Método TextBox1\_TextChanged |
| valido | Boolean |  |  | Método TextBox1\_TextChanged |
| validacion | libValidacionDatos.Validacion |  |  | Método TextBox2\_TextChanged |
| valido | Boolean |  |  | Método TextBox2\_TextChanged |
| validacion | libValidacionDatos.Validacion |  |  | Método TextBox3\_TextChanged |
| valido | Boolean |  |  | Método TextBox3\_TextChanged |
| validacion | libValidacionDatos.Validacion |  |  | Método TextBox4\_TextChanged |
| valido | Boolean |  |  | Método TextBox4\_TextChanged |
| categoria | String | comboCategoria.SelectedItem |  | Método comboTamano\_SelectedIndexChanged |
| nombre | String | comboProducto.SelectedItem |  | Método comboTamano\_SelectedIndexChanged |
| tamanno | String | comboTamano.SelectedItem |  | Método comboTamano\_SelectedIndexChanged |
| lectura | LecturaEscrituraArchivos |  |  | Método comboTamano\_SelectedIndexChanged |
| numeroRegistros | Integer | lectura.numeroRegistros |  | Método comboTamano\_SelectedIndexChanged |
| codigoRespuesta | Single | MsgBox |  | Método btnModificar\_Click |
| lectura | LecturaEscrituraArchivos |  |  | Método btnModificar\_Click |
| numeroRegistros | Integer | lectura.numeroRegistros |  | Método btnModificar\_Click |
| escritura | LecturaEscrituraArchivos |  |  | Método btnModificar\_Click |
| encontradoID | Boolean | False |  | Método btnModificar\_Click |

### 2.4 Cobros

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nombre | Tipo | Valor inicial | Comentarios | Ámbito |
| banderaPago | Boolean | False | Utilizado para indicar si se usa o no tarjeta de crédito | local a la clase Cobro |
| stringTicket | String | “” | Utilizado para albergar los datos utilizados para el ticket. | publico, a la clase PantallaVentas |
| numeroRegistros | Single | 0 | Utilizado para albergar la cantidad de registros con los que cuenta un archivo de acceso aleatorio | publico, a la clase PantallaVentas |
| lectura | LecturaEscrituraArchivos | null | Utilizado para acceder a los métodos de la clase LecturaEscrituraArchivos | local al método Cobro\_Load |
| numeroRegistros | Integer | El que le de el método LecturaEscrituraArchivos.numeroRegistros | Utilizado para conocer el número de registros con los que cuenta un fichero de acceso aleatorio | local al método Cobro\_Load |
| columna | String | "" | Utilizado para albergar los datos extraidos al leer ficheros | local al método Cobro\_Load |
| aux | String | variable columna |  | local al método Cobro\_Load |
| lectura | LecturaEscrituraArchivos | null | Utilizado para acceder a los métodos de la clase LecturaEscrituraArchivos | local al método ticket |
| espacios | Single | 120 |  | local al método ticket |
| incremento | Single | 25 | El valor que se incrementa cada vez que se añade una línea en el ticket para crear espacios. | local al método ticket |
| estilo | String | "Arial" |  | local al método ticket |
| auxIVA2 | Single | (Val(Label3.Text) \* 0.21) [Calculo del iva a reducir] |  | local al método ticket |
| auxIVA | Single | Val(Label3.Text) - auxIVA2 [reduccion del al precio total] |  | local al método ticket |
| auxStrinIVA | String | auxIVA |  | local al método ticket |

## 

## C)Bibliotecas

### LecturaEscrituraArchivos

Provee de métodos que permiten la lectura y escritura de los archivos especificados como argumento en los métodos. Utiliza la estructura correspondiente a los métodos.

#### 1.1) Clase escritura:

1. Structure producto:

|  |  |  |
| --- | --- | --- |
| **Nombre** | **Tipo** | **Tamaño** |
| categoria | String | 15 |
| nombreproducto | String | 15 |
| tamaño | String | 8 |
| precio | Single | 4 bytes |

1. Public Sub escribirProducto(ByVal numeroRegistro As Integer, ByVal nombreFichero As String, ByVal categoria As String, ByVal nombre As String,

ByVal tamaño As String, ByVal precio As Single)

1. Public Sub escribirListaListBox(ByVal numeroRegistro As Integer, ByVal nombreFichero As String, ByVal columnaListBox As String)

#### 1.2) Clase Lectura:

1. Structure producto:

|  |  |  |
| --- | --- | --- |
| **Nombre** | **Tipo** | **Tamaño** |
| categoria | String | 15 |
| nombreproducto | String | 15 |
| tamaño | String | 8 |
| precio | Single | 4 bytes |

-Public Sub defaultProducto()

1. Structure Producto\_ListBox:

|  |  |  |
| --- | --- | --- |
| **Nombre** | **Tipo** | **Tamaño** |
| clumna | String | 46 |

- Public Sub defaultProducto\_ListBox()

1. Public Function numeroRegistros(ByVal nombreFichero As String)
2. Public Sub leerProducto(ByVal numeroRegistro As Integer, ByVal nombreFichero As String,

ByRef categoria As String, ByRef nombre As String,

ByRef tamaño As String, ByRef precio As Single)

1. Public Sub leerProducto\_ListBox(ByVal numeroRegistro As Integer, ByVal nombreFichero As String,

ByRef columna As String)

### libValidacionDatos

Usada para validar datos, e incluye también algunos métodos para sacar mensajes de errores y escribir en el fichero de errores relacionados con la validación de datos introducidos por el usuario.

2.1) Public Class Validacion:

1. Public Function validarUsuario(stringRecibido As String, tipo As Integer) As Boolean
2. Public Function validarCodigo(codigo As String, tipo As Integer) As Boolean
3. Public Function comprobarDatosSecuencial(file As String, usuario As String, contrasena As Integer) As Integer
4. Public Function errorLogWrite()
5. Public Function mensajeErrorDatos()
6. Public Function validarCampoProducto(stringRecibido As String, tipo As Integer) As Boolean

# **3.-Juego de pruebas**

## Inicio de sesión: Textbox usuario, evento de cambio de texto:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nº | Id | Descr. | Input | Resul.esperados |
| 1 | cp\_1 | Dejamos vacío el campo. | Vacío | Sin efecto.(El botón de inicio de sesión no está habilitado si quiera) |
| 2 | cp\_2 | Comprobamos cómo reacciona el textbox al cambio de texto con un carácter especial. | & | Error |
| 3 | cp\_3 | Comprobamos cómo reacciona el textbox al cambio de texto con demasiadas letras. | aaaaaaaaa | Error. |
| 4 | cp\_4 | Comprobamos cómo reacciona el textbox al cambio de texto con una combinación de dígito y carácter | 1A | Salta mensaje de error. |
| 5 | cp\_5 | Comprobamos cómo reacciona el textbox al cambio de texto con un número muy grande. | 99999999 | Salta mensaje de error. |

## Inicio de sesión: Textbox contraseña, evento de cambio de texto:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nº | Id | Descr. | Input | Resul.esperados |
| 1 | cp\_1 | Dejamos vacío el campo. | Vacío | Sin efecto.(El botón de inicio de sesión no está habilitado si quiera) |
| 2 | cp\_2 | Comprobamos cómo reacciona el textbox al cambio de texto. | & | Sin efecto, porque no se permite. |
| 3 | cp\_3 | Comprobamos qué pasa si ponemos caracteres. | a | Sin efecto, porque no se permite. |
| 4 | cp\_4 | Comprobamos si el campo de texto reacciona a un número muy largo de dígitos. | 77777777 | Error. No se permiten más de 4 dígitos. |

1. Creación de usuarios:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nº | Id | Descr. | Input | Resul.esperados |
| 1 | cp\_1 | Probamos números en el campo usuario. | 1241 | Aviso al usuario y limpiar textbox. |
| 2 | cp\_2 | Comprobamos cómo reacciona el textbox usuario al cambio de texto con carácter especial. | & | Aviso al usuario y limpiar textbox. |
| 3 | cp\_3 | Comprobamos qué pasa si ponemos caracteres. | a | Sin efecto, porque se permite. |
| 4 | cp\_4 | Comprobamos si el campo de texto reacciona a un número muy largo de caracteres | abcdefghijkeiuzczw | Aviso al usuario sobre longitud prohibida y limpiar textbox. |

# 

# 

# 

# 

# 

# 

# **4.-Diagrama de clases**

