

# TERMINAL DE PUESTO DE VENTA



# ÍNDICE

1.Descripción del problema .....	3
2.Solución proporcionada .....	3
2.1 Introducción .....	3
2.2 Herramientas utilizadas .....	5
2.3 Base de datos .....	6
2.4 Estructura del proyecto .....	7
2.4.1 Organización de la aplicación .....	7
2.4.2 Explicación de las clases .....	9
2.4.2.1 BackEnd .....	9
2.4.2.2 FrontEnd .....	10
2.4.2.2.1 Permisos.....	11
2.4.2.2.2 MainWindow.....	11
2.4.2.2.3 Controles de Usuario.....	12
2.4.2.2.4 Diálogos.....	17
2.4.3 Interfaz gráfica .....	25
2.5 Utilización del programa .....	26
3.Conclusiones .....	33
3.1 Dificultades encontradas .....	33
3.2 Ampliaciones y futuros proyectos .....	33
3.3 Conclusiones personales .....	34

# 1. DESCRIPCIÓN DEL PROBLEMA

Un nuevo bar va a abrir en la ciudad y necesita un modo sencillo y eficiente de llevar un registro de las ventas y otras operaciones inherentes a un negocio de este tipo, puesto que en su anterior empresa no fueron capaces de lograr este cometido debido a que no contaban con la tecnología necesaria y los errores eran frecuentes debido a la limitación que ello suponía, llevando todo esto a una seria desorganización de los datos que debían manejar hasta que fue demasiado tarde para remediarlo.

Habiendo pasado un tiempo decidieron emprender un negocio nuevo (el actual), pero esta vez aprendiendo de la anterior experiencia y sabiendo que necesitarían una herramienta (un software en este caso) que manejase lo necesario para tener todo al día y en orden, además de minimizar las posibilidades de que se cometan errores humanos que podrían conllevar problemas en el futuro.

## 2. SOLUCIÓN PROPORCIONADA

### 2.1 Introducción

Para satisfacer esta necesidad crearemos una aplicación de Terminal de Puesto de Venta (**TPV** de ahora en adelante) que tendrá un uso sencillo e intuitivo para que el usuario sea capaz de manejarlo rápidamente.

El TPV es un dispositivo que se utiliza en el sector de la hostelería para llevar a cabo transacciones comerciales. Suelen estar conectados a una computadora o a un sistema de caja registradora y permiten a los clientes hacer sus pagos de forma fácil y rápida. Pero esta no será su única función, puesto que también nos permitirá llevar un control del stock actual y realizar operaciones de introducción, modificación, eliminación y consulta sobre las ventas, productos, clientes y usuarios.

Su implementación supone unas ventajas de las que, sin la aplicación, no podríamos disfrutar. Mencionamos las tres que considero más importantes:

La más evidente, mencionada antes, es llevar un mejor control del negocio. Al estar digitalizado, minimiza el riesgo de cometer errores humanos que luego haya que subsanar, costando ello tiempo y dinero.

Esto lleva a otra ventaja: la maximización de los beneficios, pues siendo conocedor de los datos de ventas, clientes, inventario... se conseguirá sacar una mayor rentabilidad al negocio (un ejemplo de esto sería comprobar qué productos son más vendidos y en qué franjas horarias para poder actuar en consecuencia conociendo esta información, por ejemplo, favoreciendo la compra de dicho producto en detrimento de otro menos vendido).

La tercera sería una mejora en la imagen del negocio, puesto que la implementación de tecnología con el fin de buscar la optimización siempre va a dar una imagen de eficiencia y profesionalidad de cara a los potenciales y actuales clientes.

La aplicación también implementará un sistema de permisos y roles que restringirá ciertas funcionalidades en función del rol asignado a cada usuario. Concretamente tendríamos los siguientes tres roles:

- Empleado -> El más restringido. Prácticamente solo podrá realizar ventas
- Encargado -> Rol intermedio sin apenas restricciones
- Gerente -> Rol que goza de todas las libertades y privilegios

Veremos más adelante cuáles son exactamente los privilegios de que goza cada rol y una breve justificación de estas decisiones.

Dicho esto, veamos cómo se va a llevar a cabo el diseño de la aplicación.

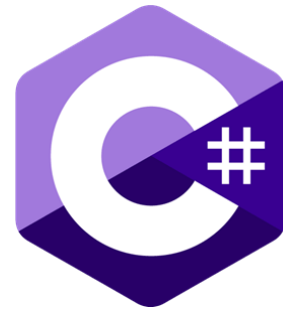
## 2.2 Herramientas utilizadas



Visual Studio 2019



XAML



C#



SQL



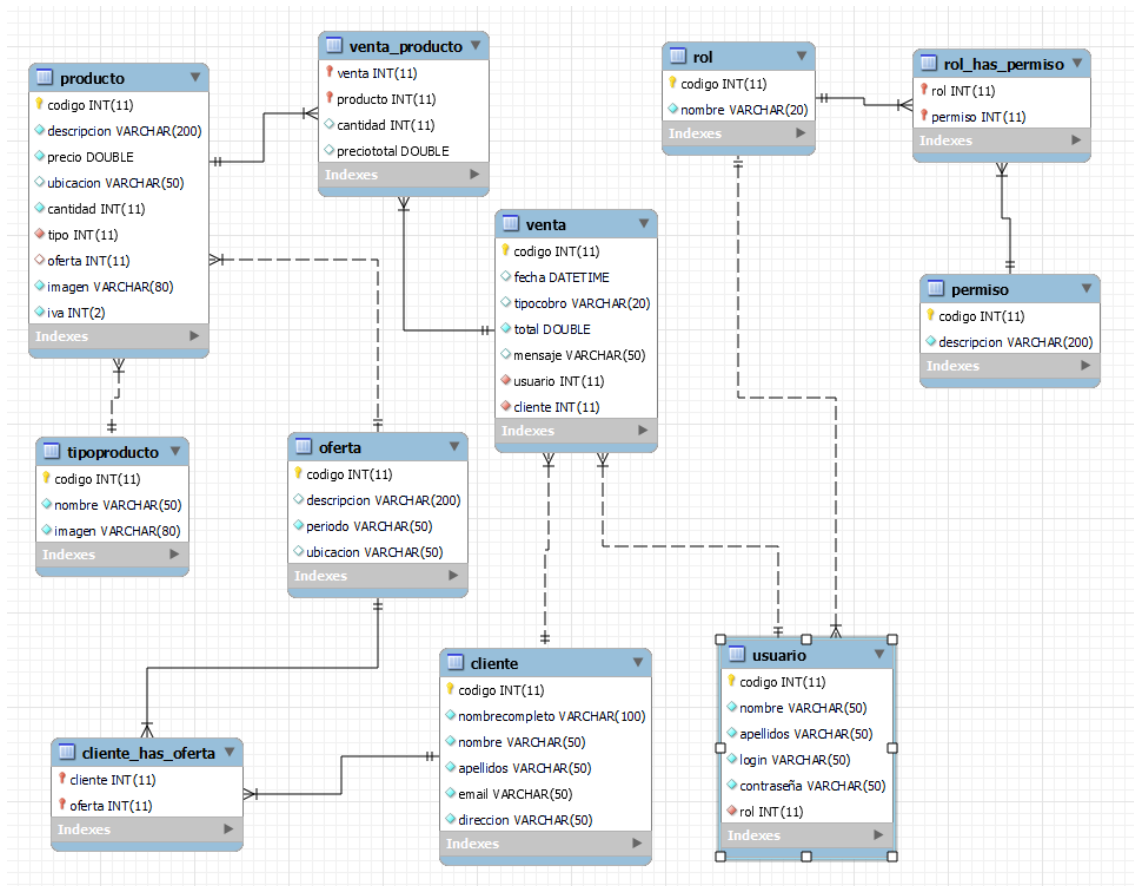
MySQL

Además de la aplicación en sí será necesaria una base de datos (**BBDD** de ahora en adelante) que la respalde y complemente con la que poder almacenar y consultar los datos. Dicha BBDD será creada en lenguaje SQL para posteriormente ser integrada en la aplicación.

Dicha aplicación se desarrollará en Visual Studio 2019 con la combinación de dos lenguajes de programación. Por un lado, XAML para diseños y FrontEnd, y, por el otro, C# para el BackEnd, es decir, toda la lógica interna de aplicación, además de una sección específica del FrontEnd. Dichos detalles serán descritos en secciones posteriores.

## 2.3 Base de Datos

Este sería el esquema de la BBDD:



Siendo las principales clases:

- **Producto**
  - Código (PK)
  - Descripción
  - Precio
  - Ubicación
  - Cantidad
  - Imagen
  - IVA
  - Tipo (FK)
  - Oferta (FK)

- **Venta**
  - Código (PK)
  - Fecha
  - Tipo de cobro
  - Total
  - Mensaje
  - Usuario (FK)
  - Cliente (FK)
- **Usuario**
  - Código (PK)
  - Nombre
  - Apellidos
  - Login (nombre de usuario)
  - Contraseña
  - Rol (FK)
- **Cliente**
  - Código (PK)
  - Nombre completo
  - Nombre
  - Apellidos
  - Email
  - Dirección

## 2.4 Estructura del proyecto

Veamos ahora la aplicación más al detalle. Explicaremos en qué secciones está dividida y qué clases la componen.

### 2.4.1 Organización de la aplicación

La aplicación está dividida en 3 carpetas principales: BackEnd, FrontEnd y Recursos.

Dentro del BackEnd encontramos dos grandes carpetas: Modelo y Servicios. Dentro de Modelo está la versión de Visual Studio de nuestra BBDD con sus objetos,

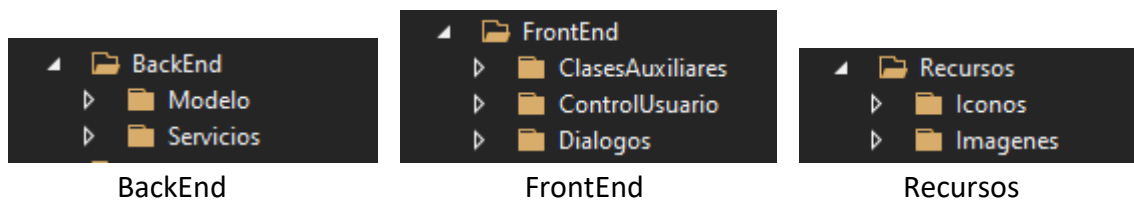
relaciones y registros que habremos incorporado previamente. En Servicios tendremos los objetos que representan la lógica de la aplicación. Pueden servir para sacar una lista de los productos guardados en la BBDD o comprobar si el username de algún usuario ya existe, por poner dos ejemplos.

En el FrontEnd tenemos el grueso de la aplicación. Aquí estarán los Diálogos y los Controles de Usuario (**UC** a partir de ahora). Los Diálogos están enfocados a añadir y editar información (a excepción de las ventas) y la pantalla de Login. Los UCs tienen su foco en los listados en los que poder consultar la información de la BBDD y poder filtrarla por diversos criterios, además del panel más importante: el panel donde seleccionamos los productos para hacer las ventas.

En la carpeta Recursos están almacenados todos los iconos e imágenes que utilizaremos en la aplicación, que servirán para dar apoyo visual a productos y botones principalmente.

La razón de elegir esta organización es simplemente porque me parece bastante intuitiva y puedes encontrar rápidamente el archivo que estás buscando usando una nomenclatura bastante clara y descriptiva para las carpetas y subcarpetas que componen la aplicación.

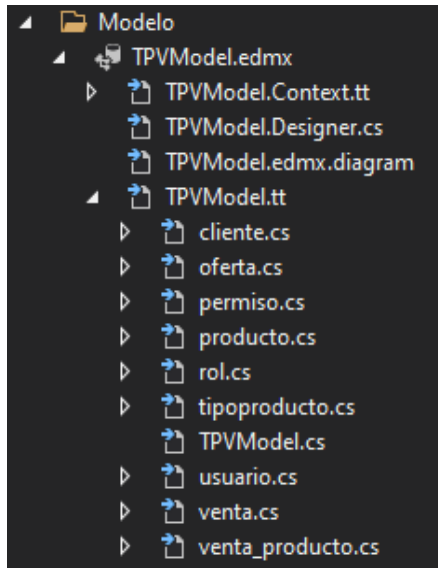
Veamos en profundidad el contenido de cada una en la siguiente sección.



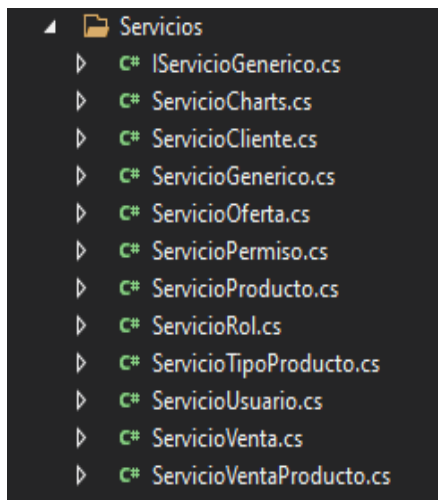


## 2.4.2 Explicación de las clases

### 2.4.2.1 BackEnd



Empezando por el BackEnd tenemos el modelo donde está guardado cada entidad que utilizaremos de la BBDD en formato CS. Estos son las tablas de la BBDD menos las formadas producto de una relación muchos a muchos con la excepción de venta\_producto por poseer atributos propios además de las claves primarias que forman la relación.



Dentro también del BackEnd se encuentran los Servicios, los cuales ya hemos definido antes como los objetos que representan la lógica de la aplicación.

En el Servicio Genérico encontraremos los métodos base compartidos por el resto de servicios que serán los de añadir, editar o eliminar una entidad de nuestra BBDD, además de un método para encontrar un objeto por su ID y el método `getAll()`, el cual utilizaremos bastante para sacar listas de los objetos que necesitamos en cada momento. Éste, además, hereda de `IServicioGenerico`, la interfaz de la cual hereda sus métodos.

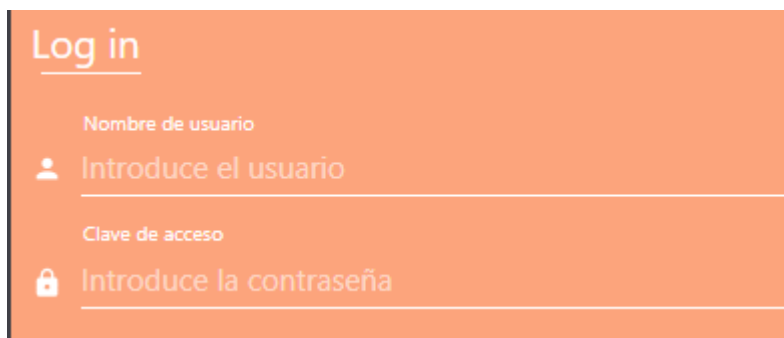
Los 4 servicios que utilizaremos principalmente son los de Producto, Cliente, Usuario y Venta, que son las entidades con las que trabajaremos principalmente en la aplicación. Estos servicios tienen un método en común, el cual es `getLastId()`, que, como su nombre indica, nos permitirá encontrar el último ID de una tabla para después sumarle 1 y así no repetir ID, cosa que la BBDD no nos permitirá y nos privará de añadir datos a la misma.

Dentro de los métodos únicos de estos servicios, el más destacado es el método `login(usuario, contraseña)` del [Servicio Usuario](#), el cual sirve, tras introducir un nombre de usuario y una contraseña, para comprobar las credenciales de los usuarios que acceden a la aplicación. Lo veremos con un poco más de detalle cuando revisemos la ventana de [Login](#).

#### 2.4.2.2 FrontEnd

Recordemos que el FrontEnd estaba compuesto de tres carpetas: Clases Auxiliares, Controles de Usuario y Diálogos.

Veamos primero la ventana de [Login](#) (ubicada en la carpeta de Diálogos), pues es la primera que veremos al arrancar la aplicación. Al hacerlo el método `conectar()` tratará de hacer conexión con la BBDD y comprobará si ésta se ha podido realizar.



Una interfaz sencilla que pedirá al usuario su username y su contraseña. Una vez introducidos y habiendo pulsado el botón de Login, un método comprobará que los

campos no estén vacíos y además accederá al método login mencionado anteriormente en el Servicio Usuario que comprobará que las credenciales coinciden con las presentes en la BBDD.

Si todo ha ido bien, la aplicación nos permitirá acceder a la ventana principal o [Main Window](#), ventana desde la cual accederemos a prácticamente todas las funciones del TPV.

#### 2.4.2.2.1 Permisos

Antes de pasar a la ventana principal, vamos a echar un vistazo a la tabla que resume a qué funcionalidades podrá acceder el usuario en función de su rol:

Permiso ▼	Empleado ▼	Encargado ▼	Gerente ▼
Introducir ventas	Sí	Sí	Sí
Devolución de ventas	No	Sí	Sí
Introducir productos/clientes	Sí	Sí	Sí
Modificar/eliminar productos/clientes	No	Sí	Sí
Introducir usuarios	No	Sí	Sí
Eliminar usuarios	No	No	Sí
Cambiar contraseñas	Solo la suya	Solo la suya	Sí

Como se puede observar, el gerente sería como un administrador, pues tiene plenos poderes sobre la aplicación pudiendo hacer y deshacer a su antojo.

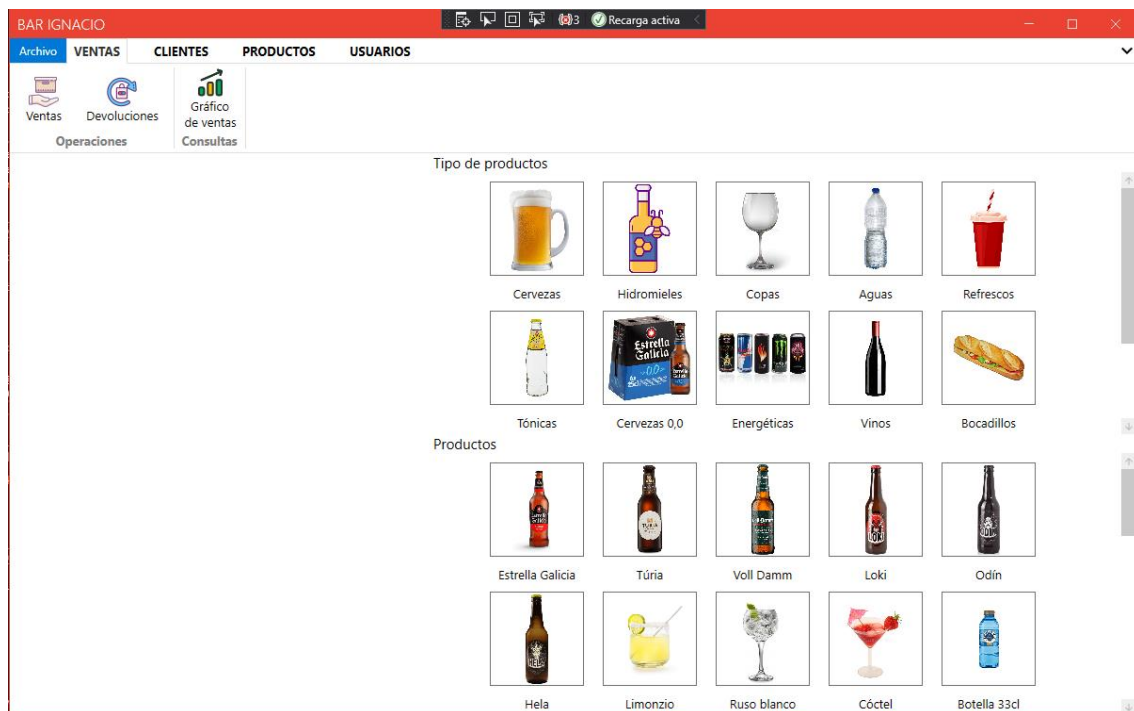
El encargado tiene casi los mismos privilegios, con la salvedad de no poder eliminar usuarios (privilegio reservado al gerente) y que la única contraseña que puede modificar es la suya propia; por lo demás puede hacer y deshacer a su antojo.

Por último, tenemos al empleado, el rol más limitado. Normalmente sólo se encargaría de las ventas, pero hemos permitido que añada clientes que deseen recibir ofertas y que puedan modificar su clave de acceso.

#### 2.4.2.2.2 MainWindow

Como su nombre indica, esta es la ventana principal de la aplicación y desde donde se accederá al resto de las funcionalidades como son el añadir, editar o eliminar datos, consulta de los mismos y, por supuesto, la realización de ventas. Esta última es con la que arrancará por defecto la ventana principal, pues es, con diferencia, la más utilizada en un negocio (muchas veces la única utilizada por los empleados).

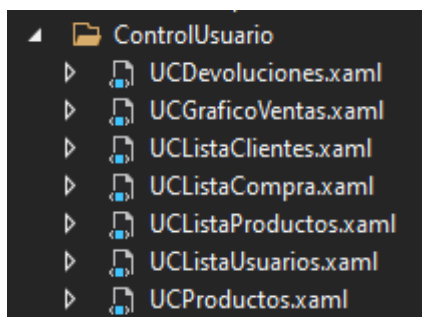
Su aspecto inicial es tal que este:



Se puede observar que en la parte superior hay 4 pestañas por las que podemos navegar. Hay una quinta pestaña a la izquierda del todo que nos da la opción de recargar la pantalla de ventas (la que se ve en la captura) o de cerrar la sesión y volver al [Login](#).

Ya que la lista de productos a la derecha pertenece a un Control de Usuario, empezaremos explicando éstos y posteriormente los Diálogos.

#### 2.4.2.2.3 Controles de Usuario



Ya hemos mencionado que, en esta aplicación, los UCs van a ser utilizados principalmente para ver listados detallados de ciertos datos de la BBDD, así como la posibilidad de filtrarlos por diversos criterios referentes a sus atributos.

Todos los UCs tienen una estructura base común en el código:

- Creación primera de objetos privados, los cuales son principalmente Servicios, Lists y entities (el contexto de la BBDD).
- Un método **inicializa()** que inicializa dichos objetos además de los creados en el correspondiente archivo XAML

Además de estas dos características, los UCs que sacan listados de datos comparten estas otras:

- Un método de filtro combinado para aplicar más de un filtro simultáneamente
- Un método para inicializar los criterios
- Un método para incluir cada criterio aportado por el usuario
- Botones de editado y borrado

Vamos a ver primero los dos que solo cumplen las dos primeras características comunes, que son **UCProductos** y **UCListaCompra**, ambos involucrados en ese orden en las operaciones de venta.

Como se vio en MainWindow, **UCProductos** es donde seleccionaremos los productos que vamos a vender. Está compuesto de dos WrapPanels de botones. El primero contiene una lista de los tipos de productos con sus respectivos nombres e imágenes y el segundo una lista, al principio completa de todos los productos en la BBDD también con sus respectivos nombres e imágenes.

Hacer click en cualquier botón de tipo de producto hará que solo se muestren los productos del tipo seleccionado para hacer más sencilla la búsqueda de un producto determinado. El último botón de tipo de producto hará que vuelvan a aparecer todos los productos.

Hacer click en cualquier botón de producto por primera vez creará el siguiente UC, **UCListaCompra**. Veámoslo más en detalle:

DESCRIPCION	CANTIDAD	PRECIO/UD	TOTAL
Estrella Galicia	1	2.5	2.5
Hela	1	3.5	3.5
Limonzio	1	7	7
Ruso blanco	2	7	14
Cóctel	1	8.5	8.5
Botella 33cl	1	1	1



La primera vez que hagamos click en un botón de producto, aparecerá este UC al lado izquierdo de la ventana principal y aparecerá el primer producto de la lista. Hacer click en un producto que no esté en la lista lo añadirá a la misma y hacerlo sobre uno ya listado le añadirá una unidad al mismo.



**Total:** 36,5

Comprar

Borrar Todo

Quitar Producto

Quitar 1 unidad

1

2

3

10

4

5

6

100

7

8

9

1000

Podremos ver el nombre de los productos, así como su cantidad, precio unitario y precio total, además del precio de toda la compra, el cual es recalculado con método cada vez que la lista sufre algún cambio.

Los cuatro primeros botones son bastante autodescriptivos. El primero abre el diálogo **AddVenta**, el cual veremos más adelante,

para agregar los últimos detalles de la venta con la lista de productos elegida (debe haber mínimo un producto). El segundo borra toda la lista de la compra. Para que los dos botones inferiores funcionen se debe haber seleccionado previamente un producto **de la lista** (no del panel de productos) y, respectivamente, quitarán el producto de la lista o solo 1 unidad.

En cuanto a los botones numéricos también es necesario haber seleccionado un producto de la lista y pulsar en cualquiera de estos botones añadirá la cantidad mostrada siempre que haya stock suficiente de dicho producto.

Aunque el usuario nunca lo vaya a notar, el stock es restado mientras se está manejando este UC y no en el momento de realizar la venta, del mismo modo que eliminar unidades o un producto de la lista repondrá el correspondiente stock inmediatamente.

Ya que hemos empezado con las ventas, vamos a ver el [UCDevoluciones](#), que no deja de ser también un listado de éstas.

DEVOLUCIONES/LISTADO DE VENTAS			
Fecha inicial	Fecha final	Buscar por cliente	Seleccione un cliente
FECHA	TOTAL	TIPO DE COBRO	CLIENTE
2/22/2023 10:54:35 PM	10	Efectivo	Mónica Salazar
5/4/2023 9:31:45 AM	38.5	Efectivo	Luis Brazofuente
PRODUCTO	CANTIDAD	PRECIO/UD	TOTAL
Voll Damm	1	3	3
Loki	1	3.5	3.5
Odín	1	3	3
Ruso blanco	4	7	28
Botella 33cl	1	1	1
4/14/2023 11:32:10 PM	16.4	Bizum	Victoria De La Vega

Esta ventana tiene diversas funcionalidades. La primera que salta a la vista es que es un listado con las ventas realizadas hasta el momento incluyendo la fecha y hora en que se realizó, el precio total, el tipo de cobro y el cliente si éste está registrado en la BBDD. No sólo eso, si hacemos click en una venta podremos ver también su “lista de la compra”, o lo que es lo mismo, los productos que la componen y el valor y cantidad de cada uno, exactamente lo que se mostraba en [UCListaCompra](#). Dichos detalles pueden volver a esconderse con el botón a la derecha de la pantalla.

La segunda es la de filtrar por fecha inicial, fecha final y por cliente. Las fechas seleccionadas estarán incluidas en el filtro (por ejemplo, si se selecciona 1 de junio de 2023 como fecha, tanto inicial como final, aparecerán ventas de dicha fecha también).

La última es la de devolver una venta o, lo que es lo mismo para nosotros, eliminarla de la BBDD. No solo eliminará dicha venta, sino que además eliminará la “lista de la compra” que lleva asociada, es decir, los valores de la tabla venta\_producto y repondrá el stock correspondiente de cada producto. Este es el código:

```

MessageBoxResult resultado = MessageBox.Show(
    "¿Desea devolver esta venta?", "GESTIÓN VENTAS", MessageBoxButtons.YesNo);
if(resultado == DialogResult.Yes)
{
    try
    {
        foreach(venta_producto ventaProd in listaVentaProds)
        {
            if(ventaProd.venta == ventaSeleccionada.codigo)
            {
                producto prod = ventaProd.producto1;
                prod.cantidad += (int)ventaProd.cantidad;


                servVentaProd.delete(ventaProd);
            }
        }

        servVenta.delete(ventaSeleccionada);
        servVenta.save();
    }
}

```

UCListaClientes, UCListaProductos y UCListaUsuarios son muy similares así que los revisaremos brevemente. Todos los filtros de texto son `Contains()` y no `StartsWith()`

### UCListaClientes

LISTA DE CLIENTES			
Buscar por nombre:	<input type="text"/>	Buscar por apellido:	<input type="text"/>
			
NOMBRE	APELLIDOS	EMAIL	DIRECCIÓN
Evaristo	Rey de la Baraja	evaristorey@gmail.com	C/Falsa 123

- Listado de clientes con su nombre, apellidos, email y dirección
- Filtros por nombre y apellidos
- Botón de quitar filtros, edición y borrado

### UCListaProductos

LISTA DE PRODUCTOS				
Filtrar por nombre	<input type="text"/>	Filtrar por tipo producto	<input type="text" value="Seleccione un tipo"/>	<input type="checkbox"/> Stock escaso (<=10)
				
IMAGEN	NOMBRE	PRECIO	CANTIDAD	TIPO PRODUCTO
	Estrella Galicia	2.5	98	Cervezas

- Listado de productos con imagen, nombre, precio, cantidad y tipo
- Filtro por nombre, comboBox de tipo de producto y checkBox de stock escaso o productos con menos de 10 unidades en stock
- Botón de quitar filtro, edición, borrado y reponer. Este último simplemente añade la cantidad indicada por el usuario al producto. Se asegura de que solo reciba números y de que sean positivos.

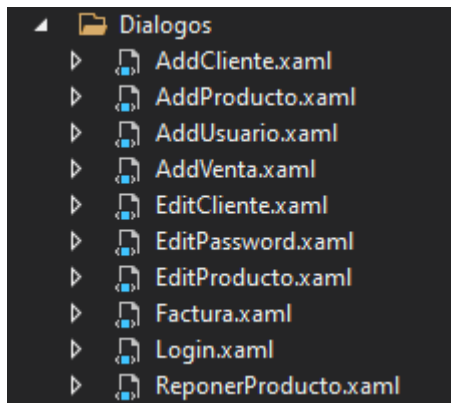
### UCListaUsuarios

LISTA DE USUARIOS			
Buscar por username:	<input type="text"/>	Buscar por nombre y/o apellidos:	<input type="text"/>
			
USERNAME	ROL	NOMBRE	APELLIDOS
gerente	Gerente	Gerente	Gerencia

- Listado de usuarios por username, rol, nombre y apellidos
- Filtro por username, nombre, apellidos y comboBox de rol
- Botones de quitar filtro, edición y borrado. La edición solo permitirá cambiar la contraseña de dicho usuario con las restricciones expuestas en la tabla de privilegios.



#### 2.4.2.2.4 Diálogos



En los diálogos encontramos principalmente las ventanas de inserción y modificación de datos, además de la emisión de la factura de una venta y el revisado anteriormente Login. Igual que en los UCs vamos a verlos por grupos

A excepción de **Login**, **Factura** y **ReponerProducto**, todos los diálogos tienen esto en común:

- Creación primera de objetos privados, los cuales son principalmente Servicios, Lists, entities (el contexto de la BBDD) y el objeto al que se añadirán o editarán los atributos.
- Un método **inicializa()** que inicializa dichos objetos además de los creados en el correspondiente archivo XAML.
- Un botón para cancelar y otro para guardar. En este último es donde se llama al método **save()** del **Servicio Genérico** para que guarde los cambios hechos a la BBDD.
- Un método **recogeDatos()** que asignará valores al objeto.
- Un método **valida()** para asegurarse de que los campos obligatorios han sido rellenados.
- Los métodos **muestraError()** y **quitaError()** que, respectivamente, resaltan los campos en los que hay conflictos o quitar dicho resaltado al suprimir ante la eliminación del conflicto.

El proceso sería el siguiente:

Se rellenan los campos -> **valida()** comprueba que todo está correcto -> **recogeDatos()** asigna los valores de los campos a atributos del objeto nuevo -> **add()** agrega el objeto a la BBDD / **edit()** edita los datos del objeto -> **save()** guarda los cambios

### AddCliente

## NUEVO CLIENTE

Todos los campos son obligatorios

<b>NOMBRE</b>	<b>APELLIDOS</b>
<input type="text"/>	<input type="text"/>
<b>EMAIL</b>	<b>DIRECCIÓN</b>
<input type="text"/>	<input type="text"/>

Hay que rellenar los 4 atributos del cliente para poder guardarlo, de lo contrario, la aplicación no nos lo permitirá.

Tanto aquí como en [AddVenta](#) y [AddProducto](#) haremos uso del método [getLastId\(\)](#) que se mencionó en el apartado Servicios de la siguiente forma:

```
if (listaClientes.Count() == 0) clienteNuevo.codigo = 1;  
else { clienteNuevo.codigo = servCli.getLastId() + 1; }
```

Si todavía no existe ningún cliente/venta/producto se le asignará id 1. Si ya hay datos en la tabla, el método buscará el id de mayor valor y le sumará 1. De esta forma no habrá ningún id repetido y no saltará ninguna excepción ni fallo de la BBDD.

### AddProducto

## NUEVO PRODUCTO

Los campos marcados con \* son obligatorios

<b>DESCRIPCIÓN*</b>	<b>PRECIO*</b>
<input type="text"/>	<input type="text" value="0"/> + -
<b>UBICACIÓN</b>	<b>CANTIDAD*</b>
<input type="text" value="Selecciona una ubicación"/>	<input type="text" value="0"/> + -
<b>IVA*</b>	<b>TIPO*</b>
<input type="text" value="Selecciona un % de IVA"/>	<input type="text" value="Selecciona un tipo de productc"/>
<b>IMAGEN*</b>	
<input type="text"/> ...	<input type="button" value="CANCELAR"/> <input type="button" value="GUARDAR"/>

Similar al anterior, será necesario rellenar los campos obligatorios, aunque en este caso hay uno opcional. Los valores de precio y cantidad se pueden teclear manualmente o se puede ir haciendo click en los símbolos +- para añadir o restar valor. Para la imagen se abrirá un OpenFileDialog para buscar y seleccionar una imagen.

La forma de asignar un id al producto es idéntica a la del cliente.

### AddUsuario

## NUEVO USUARIO

**Todos los campos son obligatorios**

<b>NOMBRE*</b>	<b>APELLIDOS*</b>
<input type="text"/>	<input type="text"/>
<b>USERNAME*</b>	<b>CONTRASEÑA*</b>
<input type="text"/>	<input type="text"/>
<b>ROL*</b>	
<input type="text" value="Selecciona un rol"/>	

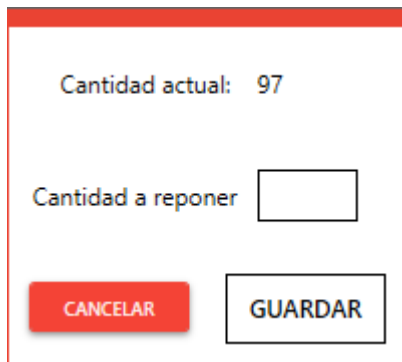
De nuevo, similar a los anteriores. Esta vez también son obligatorios todos los campos y también son alfanuméricos con la excepción de la selección del rol.

En este caso, no hay ninguna comprobación de si existe algún usuario o no para darle valor al usuario nuevo, puesto que debe de existir un usuario en primera instancia para utilizar la aplicación. Se usa también el método `getLastId()` y se le suma 1 después.

### EditCliente y EditProducto

Son exactamente iguales a `AddCliente` y `AddProducto` con la única salvedad de que ya hay datos rellenos al abrir el diálogo y que a voluntad del usuario el modificarlos o no. Evidentemente, el único atributo que no puede modificarse es el id, de modo que en estas clases no encontraremos el método `getLastId()`.

### ReponerProducto



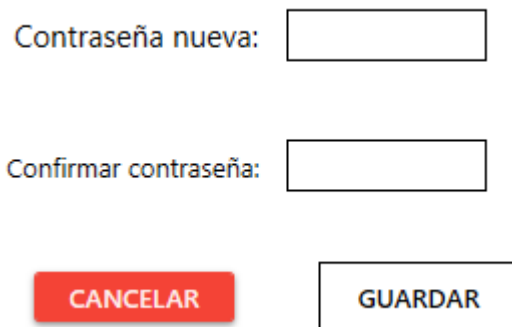
Cantidad actual: 97

Cantidad a reponer

**CANCELAR** **GUARDAR**

Sencilla interfaz en la que indicamos la cantidad en la que queremos incrementar el stock de un producto determinado. La aplicación se encarga de que sólo se acepten valores numéricos por encima de 0 mediante una condición y una `FormatException`.

### EditPassword



Contraseña nueva:

Confirmar contraseña:

**CANCELAR** **GUARDAR**

De los usuarios solo seremos capaces de editar sus contraseñas. Si queremos modificar otros atributos tendríamos que ir directamente a la BBDD. Sólo los gerentes pueden modificar cualquier contraseña, el resto de usuarios solo podrán modificar la suya propia.

En esta operación hay unas cuantas cosas que pueden salir mal, de modo que es necesario hacer varias comprobaciones para cambiar la contraseña con éxito. Es necesario asegurarse de que:

- Los campos no están vacíos (igual que en el resto de diálogos)
- Los dos campos coinciden para confirmar el cambio de contraseña
- La nueva contraseña no es igual a la antigua

En código lo veríamos así:

```
//La contraseña nueva no está vacía
if (string.IsNullOrEmpty(txtNewPass.Password))
{
    txtNewPass.BorderBrush = Brushes.Red;
    correcto = false;
    txtError.Text = "Hay campos sin rellenar";
}
else
{
    txtNewPass.BorderBrush = Brushes.Gray;
}

//La confirmación tampoco está vacía
if (string.IsNullOrEmpty(txtConfirmarPass.Password))
{
    txtConfirmarPass.BorderBrush = Brushes.Red;
    correcto = false;
    txtError.Text = "Hay campos sin rellenar";
}
else
{
    txtConfirmarPass.BorderBrush = Brushes.Gray;
}

//Ambos campos coinciden
if(txtNewPass.Password != txtConfirmarPass.Password)
{
    txtNewPass.BorderBrush = Brushes.Red;
    txtConfirmarPass.BorderBrush = Brushes.Red;
    correcto = false;
    txtError.Text = "Las contraseñas nuevas no coinciden";
}
else
{
    txtNewPass.BorderBrush = Brushes.Gray;
    txtConfirmarPass.BorderBrush = Brushes.Gray;
}

//La contraseña nueva es distinta a la antigua
if (txtNewPass.Password == oldPass)
{
    txtNewPass.BorderBrush = Brushes.Red;
    txtConfirmarPass.BorderBrush = Brushes.Red;
    correcto = false;
    txtError.Text = "La nueva contraseña no puede ser la antigua";
}
else
{
    txtNewPass.BorderBrush = Brushes.Gray;
}
}
```

### AddVenta

## NUEVA VENTA

Los campos marcados con \* son obligatorios

CLIENTE	FECHA*
<input type="text" value="Selecciona un cliente"/>	<input type="text" value="6/5/2023 1:04:01 AM"/>
TIPO DE COBRO	MENSAJE
<input type="text" value="Selecciona una forma de pago"/>	<input type="text"/>

**Total:** 15,5

El paso final de la funcionalidad principal del TPV, ultimar los detalles de la venta. El único campo obligatorio es la fecha, pero al tener ya un valor por defecto, que es la fecha y hora actual, nunca podrá tener un valor nulo. Sí es susceptible de modificación.

El cliente y el tipo de cobro no son obligatorios, aunque sí recomendables en caso de ser conocedores de ellos, pues aportan valiosa información.

Al guardar la venta, también guardaremos la “lista de la compra” en `venta_producto` (lo que luego son los detalles de la venta) de la siguiente forma:

```
private void guardarVentaProductos()
{
    for (int i = 0; i < listaCompra.Count(); i++)
    {
        ventaProdNueva = new venta_producto();

        ventaProdNueva.venta1 = ventaNueva;
        ventaProdNueva.producto1 = listaCompra[i].producto1;
        ventaProdNueva.cantidad = listaCompra[i].cantidad;
        ventaProdNueva.preciototal = listaCompra[i].preciototal;

        servVentaProd.add(ventaProdNueva);
    }
}
```

Una vez confirmado todo aparecerá un MessageBox preguntado al usuario si desea emitir una factura. En caso afirmativo tendríamos lo siguiente:

### Factura

\*\*\* BAR IGNACIO \*\*\*

CIF: A322322322

Tlf atención al cliente: 660775796



DESCRIPCION	CANTIDAD	PRECIO/UD	TOTAL
Botella 33cl	3	1	3
Limonzio	3	7	21
Ruso blanco	2	7	14
Red Bull	3	2.5	7.5

=====

**TOTAL COMPRA** **45,5**

=====

FECHA 05/06/2023 1:05:50

Le atendió: Gerente

OK

ENVIAR

En la factura aparecerán los productos desglosados de la venta tal y como aparecían en el [UCListaCompra](#), el total final, la fecha en que se realizó y el usuario que gestionó la venta, el cual será el aparezca guardado en la BBDD como gestor de la venta.

Bajo estará el botón OK que simplemente cerrará la venta y el botón ENVIAR, que enviará la factura al email del cliente (en caso de haber seleccionado uno) y cerrará igualmente la ventana.



### 2.4.3 Interfaz gráfica

Para la ventana principal de la aplicación he elegido Fluent Ribbon para tener un acceso rápido a cada menú, así como una clara división entre secciones para facilitar al usuario encontrar lo que necesita en cada momento. En total son 4 pestañas (Ventas, Clientes, Productos y Usuarios) cada una dividida en 2 secciones: operaciones y consultas. De esta forma quedan bien definidas y facilita al usuario la visualización de la aplicación.

Como se ha mencionado antes, para las consultas se han utilizado UCs puesto que ofrecen una mayor comodidad que los diálogos a la hora de mostrar listados, además de tener una carga casi instantánea. Tampoco desplazan el foco de la aplicación, pues no se ha abierto ninguna ventana nueva.

Los diálogos son ventanas modales respecto a la aplicación. Esto quiere decir que, una vez abiertos, desplazan el foco hacia ellos y, aunque sigamos viendo la **MainWindow** en segundo plano, no podremos operar con ella hasta que cerremos el diálogo que hemos abierto, bien por cancelar la operación o por realizarla. De esta forma nos aseguramos de tratar primero con el diálogo recién abierto antes de seguir con otras tareas.

En lo que respecta a los controles utilizados en la parte del XAML se ha hecho lo siguiente:

- Para los campos de texto se ha utilizado TextBox, pues es, con diferencia el control que más se ajusta a esta necesidad. Este será el criterio general para la elección de controles.

- Para campos de introducción de contraseñas, PasswordBox, pues necesitamos que lo escrito aquí no pueda ser visto por terceros y vulnere nuestra seguridad y privacidad.

- Para campos de clave ajena en los diálogos de inserción y modificación de datos, ComboBox. Ofrece un listado de opciones entre los datos que necesitamos de la BBDD, evitando la introducción de cualquier dato erróneo.


- Para campos numéricos, NumericUpDown. De este modo sólo acepta valores numéricos. Con intervalos de 0.1 para números con decimales e intervalos de 1 para números enteros. La única excepción a esto está en **ReponerProducto**, donde se ha usado un TextBox asegurándose, eso sí, de que solo acepte números enteros.

- En **AddVenta** tenemos un campo DateTime que requiere no sólo una fecha, sino también una hora. De modo que la mejor solución a ello es un DateTimePicker que, como su nombre indica, permite seleccionar una fecha y una hora.

-En **AddProducto** necesitamos añadir una imagen. Para ello hay un Button que abre un OpenFileDialog que nos permitirá seleccionar un archivo de imagen.

## 2.5 Utilización del programa

En el apartado explicación de clases hemos explicado la aplicación a nivel de programador, ahora vamos a explicarla a nivel usuario. Este apartado servirá de manual sencillo para que cualquier usuario pueda utilizarla.



The screenshot shows a 'Log in' window with an orange background. It contains two text input fields and a 'Log in' button. The first field is labeled 'Nombre de usuario' and contains the placeholder text 'Introduce el usuario', with a green box and the number '1' next to it. The second field is labeled 'Clave de acceso' and contains the placeholder text 'Introduce la contraseña', with a green box and the number '2' next to it. Below these fields is a white 'Log in' button with a green border and the number '3' next to it.

Empezaremos en la pantalla de **Login**, en la que simplemente debemos introducir el nombre de usuario (1), introducir la contraseña (2) y pulsar el botón Log in (3).

Si se introdujeron las credenciales correctamente, accederemos a la aplicación. En caso contrario, deberemos volver a introducir los datos



La aplicación nos llevará la **MainWindow**, donde hay 4 pestañas en función del área

en la que queramos trabajar (Ventas, Clientes, Productos y Usuarios), además una quinta opciones extra. Por defecto estará seleccionada la de ventas al abrir la aplicación.

Dentro de ella encontramos la funcionalidad principal de Ventas (1), la cual estará también abierta al iniciar, Devoluciones (2) y Gráfico de Ventas (3).

### Tipo de productos



### Productos



Este es el aspecto del panel de Ventas. Donde tenemos abajo una lista de los productos y arriba una lista de los tipos de producto para filtrar. Si se hace click en cualquiera de los de arriba, se mostrarán solo los productos de dicha categoría. Por ejemplo, esto se mostrará si pulsamos en Hidromieles:

### Productos



Sólo los productos de dicha categoría son mostrados

**TODOS**

Navegando entre los tipos de productos encontraremos este botón al final. Servirá para volver a cargar la lista entera de los productos en caso de que así lo deseemos

TODOS LOS PRODUCTOS

DESCRIPCION	CANTIDAD	PRECIO/UD	TOTAL
Loki	3	3.5	10.5
Odín	2	3	6
Hela	3	3.5	10.5

Total: 27

Comprar

Borrar Todo

Quitar Producto

Quitar 1 unidad

1

2

3

10

4

5

6

100

7

8

9

1000

Pulsar en cualquiera de los productos por primera vez hará aparecer la lista de la compra y los correspondientes botones a la izquierda de los paneles de los productos.

Cada vez que se pulse un producto nuevo, éste será añadido a la lista.

Si ya hemos acabado de añadir productos haremos click en Comprar (1) para ultimar detalles de la venta. Si queremos modificar algo de la lista podemos borrarla toda (2), quitar un producto previamente seleccionado de la lista (3) o quitar solo una unidad de dicho producto (4)

También hay en la parte inferior un panel con números. Habiendo seleccionado antes un producto, pulsar en cualquiera de ellos, añadirá la cantidad mostrada a dicho producto de la lista, siempre y cuando haya stock suficiente. En caso de no haberlo, simplemente no se añadirá la cantidad.

Una vez hayamos añadido todos los productos haremos click en Comprar (1) y nos llevará al menú de añadir la venta.

**Los campos marcados con \* son obligatorios**

**CLIENTE**

Selecciona un cliente ▼

**FECHA\***

6/8/2023 2:38:30 PM

**TIPO DE COBRO**

Selecciona una forma de pago ▼

**MENSAJE**

**Total: 18,5**

CANCELAR

GUARDAR

El único campo obligatorio es la fecha, pero, al tener puesta por defecto la fecha y hora y actual, no debemos preocuparnos por ella. Cliente y Tipo de cobro desplegarán distintas posibilidades a elegir, pero no son obligatorios. Lo mismo para Mensaje, podremos escribir un mensaje opcional de hasta 50 caracteres.

Una vez hayamos introducido la información necesaria, haremos click en Guardar y la venta será realizada. En caso contrario, podemos pulsar Cancelar y volveremos a la lista de la compra.

Pasamos a las Devoluciones.

DEVOLUCIONES/LISTADO DE VENTAS

Fecha inicial 1
Fecha final 2
Buscar por cliente

Selecciona un cliente ▼

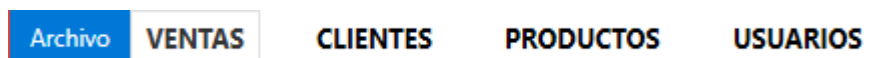
3
4
5

FECHA	TOTAL	TIPO DE COBRO	CLIENTE
2/22/2023 10:54:35 PM	10	Efectivo	Mónica Salazar

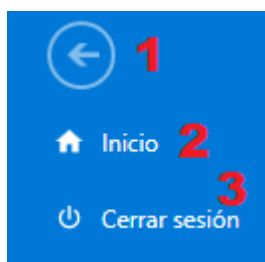
Tendremos un listado de las ventas y la posibilidad de filtrarlas. Podemos hacerlo por fecha inicial (1), fecha final (2) y cliente (3). Si seleccionamos el filtro por fecha aparecerá un calendario y seleccionaremos la fecha que deseemos. Seleccionar el filtro por cliente mostrará un desplegable con los nombres de los clientes para elegir uno de ellos. Estos filtros pueden utilizarse conjuntamente, no son excluyentes.

Habrà un botón para eliminar todos los filtros aplicados (4) y otro para devolver la venta que tengamos seleccionada (5).

Hasta aquí hemos visto lo que será utilizado de la aplicación, especialmente por los empleados que rara vez verán algo que no sea estas pantallas (ni siquiera podrán realizar devoluciones). Veamos el resto de las pestañas.



Al hacer click en Archivo, la pestaña azul, se abrirá este pequeño menú



Podremos volver a la ventana donde estuviéramos antes de pulsar (1), volver a la pantalla de Ventas (2) o cerrar la sesión y volver al Login (3).

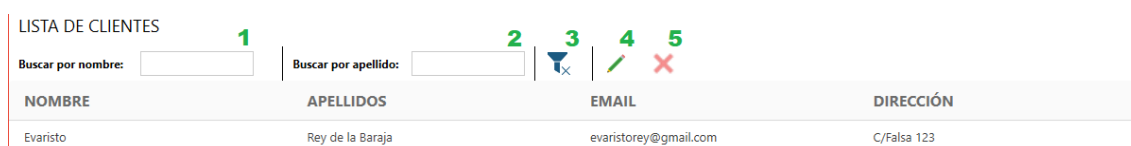
### Pestaña Clientes



Podemos añadir un cliente nuevo (1), gestionar las campañas de marketing (2) o ver el listado de clientes actuales (3).

En añadir cliente simplemente rellenamos los campos (todos son obligatorios) de nombre, apellidos, dirección e email. Todos son campos de texto, así que no habrá ninguna complicación. Si está todo correcto, pulsaremos Guardar. En caso contrario, haremos click en Cancelar.

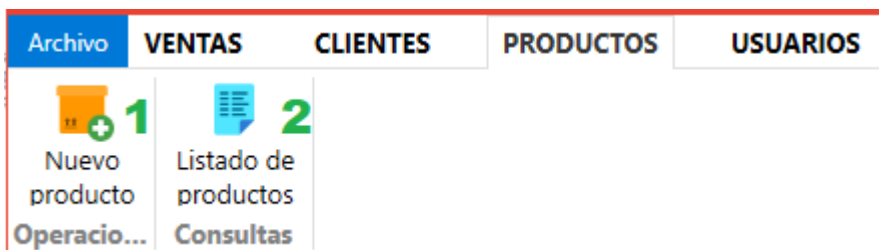
### Listado de clientes



Se nos mostrará un listado de los clientes y la posibilidad de filtrarlos por nombre (1) y apellidos (2). El filtro cogerá cualquier coincidencia con lo escrito, no solo si empieza por ello. Por ejemplo, si el usuario escribe “gna” aparecerá Ignacio en el listado.

Habr  bot n para quitar todos los filtros (3), bot n de edici n (4) que llevar  a una ventana similar a la de a adir cliente, pero con los datos ya escritos del cliente seleccionado y bot n de borrado (5) para borrar al cliente seleccionado.

### Pesta a Productos



Podemos a adir un producto (1) o ver el listado de los productos actuales (2).

### A adir producto

<b>DESCRIPCI�N*</b>	<b>PRECIO*</b>
<input type="text"/>	<input type="text" value="0"/> + -
<b>UBICACI�N</b>	<b>CANTIDAD*</b>
<input type="text" value="Selecciona una ubicaci�n"/>	<input type="text" value="0"/> + -
<b>IVA*</b>	<b>TIPO*</b>
<input type="text" value="Selecciona un % de IVA"/>	<input type="text" value="Selecciona un tipo de product"/>
<b>IMAGEN*</b>	
<input type="text"/> ...	<input type="button" value="CANCELAR"/> <input type="button" value="GUARDAR"/>

En descripci n escribiremos el nombre del producto. En precio y cantidad podemos introducir manualmente la cifra o ir pulsando los botones de + y -. Ubicaci n, IVA y tipo mostrar n un desplegable con las opciones. Para imagen tendremos que pulsar el bot n de los tres puntos (...) para abrir una ventana para buscar una imagen y seleccionarla.

## Lista productos

LISTA DE PRODUCTOS

1 2 3 4 5 6 7

Filtrar por nombre  Filtrar por tipo producto  ☐ Stock escaso (<=10)

IMAGEN	NOMBRE	PRECIO	CANTIDAD	TIPO PRODUCTO
	Estrella Galicia	2.5	98	Cervezas

Se nos mostrará un listado de los productos y la posibilidad de filtrarlos por nombre (1), tipo de producto (2) y productos con menos de 10 unidades en stock (3).

Habrà botón para quitar todos los filtros (4). Teniendo un producto de la lista seleccionado tendremos botón de edición (5) que llevará a una ventana similar a la de añadir productos, pero con los datos ya escritos de dicho producto, botón de borrado (6) para borrarlo y un botón para reponer stock del mismo (7).

## Pestaña Usuarios

Archivo VENTAS CLIENTES PRODUCTOS **USUARIOS**

1 2

Nuevo usuario Listado de usuarios

Operaci... Consultas

Similar al anterior con la opción de añadir usuario (1) o mostrar el listado de usuarios (2).

En la ventana de añadir usuario todos los campos son obligatorios, además de ser todos de texto, por lo que no habrá problema al rellenarlos. La única excepción es el rol, que es un desplegable que nos dará a elegir entre Gerente, Encargado y Empleado. Igual que los anteriores, Guardar para guardar y Cancelar para cancelar.

## Lista usuarios

LISTA DE USUARIOS

1 2 3 4 5

Buscar por username:  Buscar por nombre y/o apellidos:  Filtrar por rol

USERNAME	ROL	NOMBRE	APELLIDOS
gerente	Gerente	Gerente	Gerencia

También similar a los anteriores. Se mostrará un listado de los usuarios y la posibilidad de filtrarlos por nombre de usuario (1), nombre y/o apellidos (2) y rol (3).



Habría botón para quitar todos los filtros (3). Teniendo un usuario de la lista seleccionado tendríamos botón de edición (4) para cambiar la contraseña del usuario conociendo la antigua y botón de borrado (5) para eliminarlo.

## 3. CONCLUSIONES

### 3.1 Dificultades encontradas

Honestamente, he encontrado unas cuantas dificultades durante todo el desarrollo de la aplicación, puesto que es algo a lo que había hecho hasta entonces. Incluso antes de comenzar con el diseño, la BBDD en sí dio algún quebradero de cabeza al principio incluso con las pautas. Otorgarla atributos propios a una tabla fruto de una relación muchos a muchos fue la solución a esa dificultad inicial y resultó clave para el desempeño de la aplicación.

Posteriormente estuve indeciso con el diseño principal de la aplicación hasta que concluí que unas pestañas de FluentRibbon harían la visualización bastante agradecida e intuitiva.

Qué decir de la sección de las ventas, la principal en la que había que trabajar. Desde luego invertí más tiempo ahí que en ningún otro punto del proyecto, puesto que era de vital importancia que funcionase. Afortunadamente, logré que funcionase de la manera que me propuse y el resto de dificultades no parecieron tan grandes al lado de esta, que tenía muchos frentes que cubrir.

Por lo demás no ha habido ninguna dificultad excesivamente grande que pueda destacarse, sólo pequeñas piedras por el camino.

### 3.2 Ampliaciones y futuros proyectos

La primera ampliación interesante que podría tener es la versión para móvil, preferiblemente Android, para hacer la aplicación mucho más portátil y multiplataforma.

Agregar un datafono para permitir el pago con tarjeta. No lo calificaría de imprescindible (casi) pero sí altamente recomendable. Permitir al cliente pagar con tarjeta es mucho mejor recibido que sugerirle un Bizum, que se asegure de tener efectivo o incluso a un banco cercano a sacar dinero. Por ello, esta ampliación sería de las primeras en realizarse.

A nivel interno de la aplicación implementar el gráfico de ventas para obtener información de las mismas y poder mejorar el negocio, como ya se habló en los primeros puntos del trabajo. También poder mandar ofertas a los clientes con tentadores descuentos para fomentar la fidelización de éstos, incluso mandando campañas por las redes sociales o emails.

Un detalle que sería agradecido añadir sería que la aplicación avise, sin tener que comprobarlo previamente, que un producto anda escaso de stock. Podría hacerlo con MessageBox en el momento en que una venta nueva deja al producto corto de existencias.

### **3.3 Conclusiones personales**

A pesar de que es la primera vez que programo un TPV, no es la primera vez que estoy en contacto con uno, pues en ocasiones lo utilizo en un trabajo a tiempo parcial y no ha sido sino hasta ahora que he visto la importancia y la imprescindibilidad que supone que tener uno en un negocio, además avalada por numerosos estudios. No sólo es beneficio para el usuario y el negocio, sino que también para los clientes, pues mejora su experiencia al hacer la compra más fluida. Por no mencionar que favorece la seguridad de las transacciones.

Lo único que supera a lo que he aprendido haciendo este proyecto es lo que me he dado cuenta que todavía me queda por aprender y espero hacerlo en estos años venideros.