**POO TP FINAL**

Se desea crear un sistema de stock escalable lo más genérico posible para adaptarlo a distintos tipos de negocios, el mismo se orientara a una cadena de mini Markets.

**ADMINISTRACIÓN**

El Mismo contara con una sección de administración, la cual le permitirá ingresar productos nuevos, comparar ventas según fechas, ver que productos se venden mas y que productos tienen mas rentabilidad. (alta rotación o alto margen de ganancia), ver información o notificaciones de los locales (merma de mercadería, movimientos raros etc.) Existirá una sección para ajustar los precios de los productos por proveedor o por varios factores generales.

**USUARIO**

El mismo programa contara con una sección de usuario vendedor, el cual estará colocado en el negocio correspondiente (se podrá manejar a gusto del cliente, para controlar las ventas, los usuarios según local o según empleado).

La misma interfaz contara con la sección de cobro, la cual mediante un lector de código de barras (o a mano) se ingresara el producto, con una interfaz “Friendly” para poder con un solo botón (enter) agregar otro producto o con otro (tab) ingresar mas cantidad del mismo producto). La misma contara todos los productos y podrá quitarlos o modificarlos a gusto. Luego se finalizara la factura con otro formulario para poder elegir la forma de pago,

Además, habrá una sección de stock para el ingresar mercadería cuando llega el proveedor, ajustar cuando hay rotura o robo.

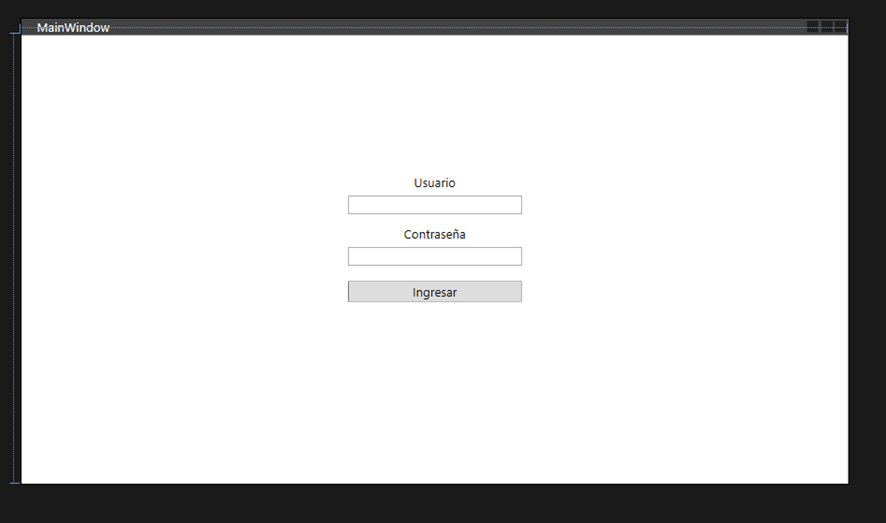
La misma tendrá una sección de cierre de caja que hace el recuento de cuánto dinero debería haber en la caja (y en obligaciones a cobrar). La diferencia se arreglara con el empleado de turno, responsable del faltante.

Un visor de ventas (solo vista) para controlar las ventas del día.

Los usuarios se regirán por permisos, que van desde el “0” en adelante, el “0” es el que tiene acceso a todo y el mas alto será el que menos acceso tenga al programa. Un usuario medio solo tendrá acceso a la interfaz de usuario, y los usuarios “0” y “1” tendrán acceso a la administración

Las tablas de datos se dictaran al final del documento

Pendiente de modificaciones.

**FORMULARIO DE INGRESO**

Para el ingreso del programa se requiere usuario y contraseña, los cual serán brindados por el administrador con permiso “0” a menos que sea requerido por el cliente, se agregar un label de registro, el cual se registrara con permiso “99” que será sin acceso, el Administrador le asignara el permiso correspondiente.

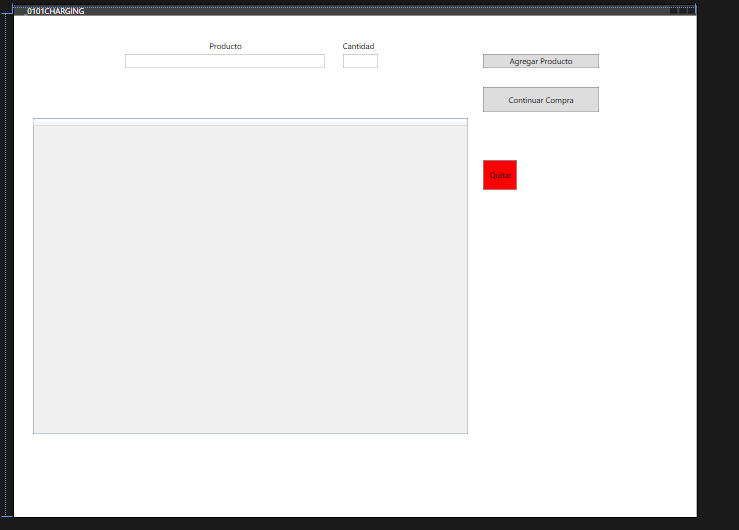
En cuanto a la validación, para los usuarios con permiso “0” y “1” será opcional según el cliente, agregar un método de validación para el ingreso el cual pedirá mediante un Textbox (IsEnabled = “False” => IsEnabled = “True”) para el ingreso seguro, el cual constara de una operación matemática entre la fecha y la hora de ingreso, la misma se enviará por mail.

**MENU USUARIO**



Sin más que agregar, luego de la validación de usuario se desplegará un formulario en el que tenga el acceso al cobro, stock, cierre de caja, las ventas del día y el botón de salir. En la sección de administración se contará con un formulario previo

**COBRO**



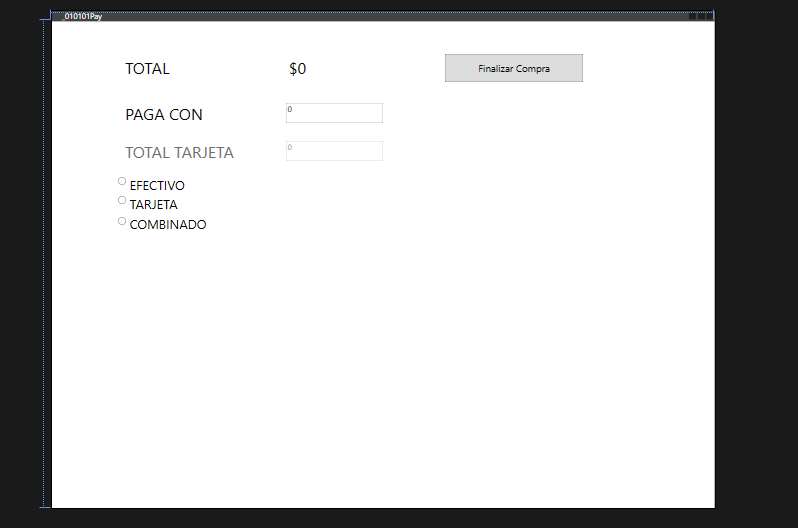
En la sección de cobro existirá un DataGridView, que será llenado con un Datasource y completado automáticamente con los campos traídos de la lista que se llenara con los productos agregados dgvNombre.DataSource = lista();

Para agregar los productos a la lista se escribira el textbox y por cada modificacion hara una consulta a la base de datos de productos que contiene todos los productos registrados, buscara por NombreProducto y por CodigoDeBarra, en un desplegable se mostraran todos los coincidentes. Con un enter se seleccionara el mas proximo y con otro enter se agregara UN producto a la lista, en caso de querer agregar mas productos a la lista se debera apretar la tecla TAB y agregar la cantidad a agregar. Cada producto podra ser quitado independientemente de la lista a partir de la propiedad codigo de barras con el boton quitar.

Para el acceso a todos los datos se contara con todas las tablas armadas al final del documento

Luego para finalizar la compra se debera presionar en el boton continuar compra, que desplegara el formulario de factura

**PAGO**



Para el cobro final se contará con este formulario, el cual contará con el total en un label, que será la suma de todos los ítems de la lista, un textbox que se llenara opcionalmente (no influye el no uso) con la cantidad de efectivo que se pagara y en un label con la cantidad de vuelto que le corresponderá al cliente. Trabajara de la misma forma para pagos combinados, solo que se descontara el pago en tarjeta.

Si se selecciona pago en tarjeta, se pondra en false, el isEnabled del “PAGA CON” y quedara habilitado el Total tarjeta.

Si se selecciona el pago combinado se pondran en True ambos IsEnabled

El Pago por defecto para asegurar una alta fluidez de la clientela sera por pago en efectivo, sin usar la opcion del cálculo del vuelto, asumiendo que un empleado eficiente sabe hacer las cuentas mentalmente. Solo con la tecla enter se deberia poder finalizar la compra

Todos los datos de la venta se guardaran en la base de datos de VENTAS la cual guardara todos los datos de la compra, y para guardar la cantidad de productos comprados se guardaran en un nvarchar(max) en un jSon serialized, el cual se deserializara para poder joinear tablas en otras partes del programa. Las ventas tendran un historico en una tabla, por producto para poder hacer devoluciones que solo mostrara el datetime de la venta incluyendo minutos y segundos y el codigo de barra del producto. Obviamente el datetime se repetira tantas veces como productos se hayan comprado en dicha factura

**Tablas:**

* **dbo.Productos:** toda la información relacionada a los productos, nombre, codigo de barra, precio, proveedor y demás

Datos

Id\_producto as string => nvarchar(10)

UF as string => nvarchar(10)

CodigoBarra as string => nvarchar(10)

Precio as decimal => decimal

* **dbo.Ventas**: toda la información sobre cada venta, (no sobre cada producto vendido). Para guardar todos los productos vendidos se serializara en un jSON en un campo llamado Productos\_vendidos guardado en un nvarchar(max)

Datos:

Productos as jSon => nvarchar(max)

* **dbo.ProductoVendido**: todos los productos vendidos con su datetime, los datetime se repetirán tantas veces como productos se hayan vendido en cada factura(estará relacionada con dbo.Ventas).

Datos:

Id\_venta as GUID,

Codigo\_barra as String => nvarchar(10)

Fecha\_y\_Hora\_Venta as DateTime

* **dbo.Usuarios:** Todos los datos de registro de usuarios, User, password, y permisos
* **dbo.UsuariosPerfil:** Todos los perfiles segun el permiso, ejemplo: 0 => AdministradorGod, 1 => Administrador, 2 => Supervisor 3 => Usuario, 4 => Solo vista, 99 => Usuario a aprobar

Clases

Crear una clase que se llame ValidInfoUser(Enum TipoValid, Enum Propertie, value string) la cual devolvera un bool true or false, en el que el primer enum sera dependiendo el valor, para un valor si existe el valor (sea para comprobar emails repetidos etc), y para otro valor si el string es valido (por ejemplo que un nombre tenga un entero). En el segundo parametro se enviara la propiedad que se desea validar, sea email, nombre, dni, etc, y por ultimo el valor.

Dentro de la misma habra un Switch Case para El tipo, Otro para la propiedad, y en caso de repetirse las validaciones, se haran en un metodo incluido en la misma clase

 La primer parte del switch sera una conexion a la base de datos y la segunda sera un analizador de string.