TP N.º 4 - Laboratorio de computación II.

El trabajo presentado funciona como boceto a un posible programa para controlar los datos de una colonia de vacaciones. Es posible agregar alumnos a la colonia en donde cada uno que ingrese es comparado con la lista de grupos existentes para verificar si existe un alumno con el mismo DNI. En caso de que no exista ningun alumno se agregará a la colonia según su edad. Alumnos de 3 a 6 años irán al grupo de los “pequeños”, de 7 a 10 años al grupo de los “medianos” y de 11 a 13 años al grupo de los “grandes”.
Además es posible modificar o eliminar alguno de los alumnos existentes en la colonia.
Sólo los “colonos” (alumno de la colonia) estarán cargados en la base de datos. En la colonia también es posible dar de alta productos para la venta, tales como “Gorritos” y “Antiparras”. Estos productos no se encuentra en base de datos, viven en memoria y algunos están hardcodeados para poder realizar pruebas.
El usuario puede agregar más cantidad de productos y venderle a los colonos algunos de estos. En caso de que se produzca una venta o se de de alta a algún producto, se modificará el stock. Al realizar una venta el sistema notificará la cantidad, de ese producto vendido, que queda disponible.

El programa cuenta con un menú principal al que se puede acceder a las distintas secciones del sistema:

- Alta alumno: Agrega el colono a la colonia.
- Buscar colono: Buscar por DNI y muestra toda la información.
- Mostrar grupos: Detalla cada grupo de alumnos existente en la colonia (pequeños, medianos y grandes).
- Alta productos: Agrega un nuevo producto a la colonia.
- Detalles ingresos: Muestra los ingresos en caja y la descripción de la lista de pagos realizados(los pagos pueden ser por cuota o por venta de productos).

Contenidos implementados:

EXCEPTION

Un ejemplo de manejo de excepciones se desarrolla en la clase Validar en la siguiente firma:

```
namespace Validaciones
{
    public static class Validar
    {

        public static int ValidarSoloNumeros(string cadenaNumerica)
        {
          ///Codigo...
        }
    }
}
```

Verifica que una cadena de caracteres ingresados sean solo números. En caso de que algún carácter no sea un número lanzará `ValidacionIncorrectaException`.

TEST UNITARIOS

Se implementan para:
- Testear que si se intenta agregar un alumno repetido, este sea rechazado ya que su dni es igual al alumno existente en la colonia.
- Testear que si se intenta agregar un colono a la colonia, esta actualice su lista de colonos conteniendo al alumno ingresado.
- Testear que un colono que un colono que intenta ser dado de alta en la colonia, se encuentre entre el rango correcto de edades(entre 3 y 13 años). Sino, no permite su ingreso.
- Testear que el nombre, apellido de un colono, sea apto. (Solo letras).

TIPOS GENÉRICOS

Ha sido utilizado como clase contenedora de dos tipos diferentes de productos a vender: “Gorritas” y “Antiparras”. Se implementa en el namespace STOCK en la clase ControlStock. Esta clase contiene una List<T> que puede ser cargada por cualquiera de los productos vendidos.

```
namespace Stock
{
    public class ControlStock<T>
    {
       ///Codigo...
    }
}
```

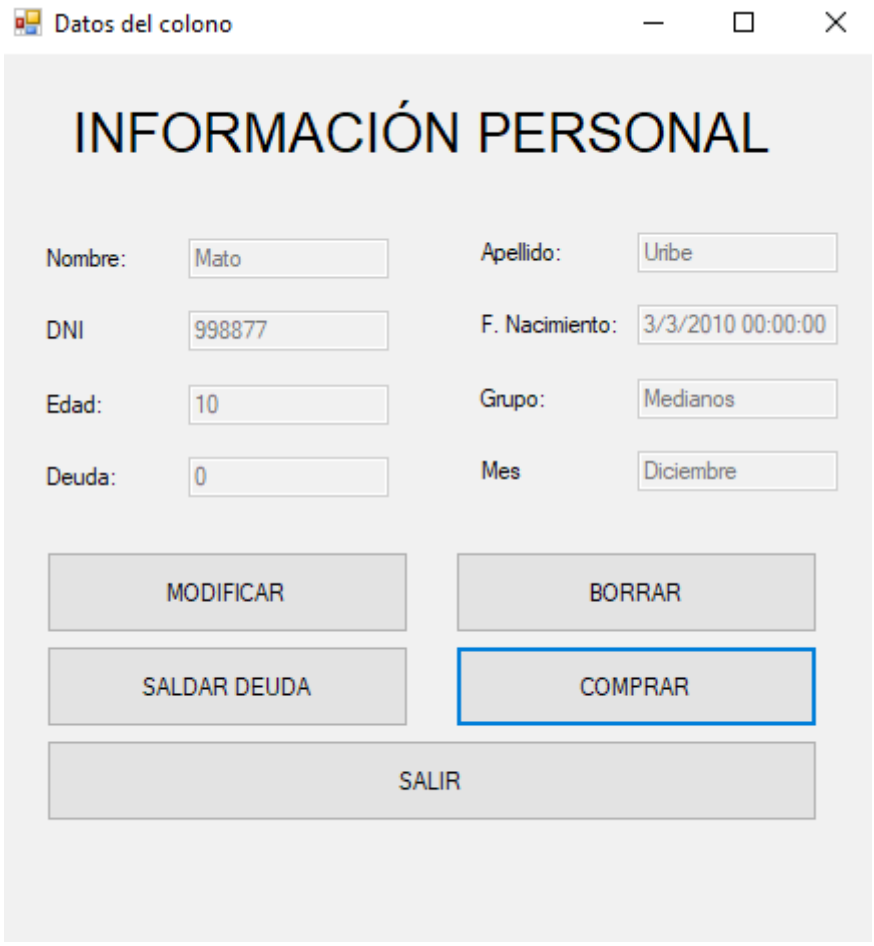
INTERFACES

Se encuentra desarrollada dentro del namespace Entidades dentro de la interfaz “Icomprar”. Es la encargada de desarrollar las ventas de los productos que la colonia tiene en stock.

```
namespace Entidades
{
    public interface IComprar
    {
        void RealizaVenta(Colonia colonia, Producto producto, Colono colono, int cantidad);


    }
}
```

Mediante el método implementado en la Clase Colonia denominado RealizaVenta() se efectúa una venta. En la misma se corrobora que el producto seleccionado se encuentre dentro del stock. El saldo de los productos comprados se sumará al saldo deudor de colono y se agregará el producto a su lista de comprados.(atributo vivo en memoria). Luego llamará al método del namespace Stock, clase ControlStock denominado BajarCantidad() que realizará la baja del producto según la cantidad comprada. En caso de que la cantidad del producto sea igual a cero, lo eliminará del stock.



Ejemplo->Boton para realizar compras.

ARCHIVOS O SERIALIZACION XML.

Se ha utilizado la escritura de archivos para guardar el saldo a favor de la colonia y el detalle de los pagos realizados.
Estos son tomados desde el archivo al abrir el formulario principal. Estos datos son utilizados para mostrar el detalle de ingresos de la colonia.

```
namespace Entidades
{
    public class Colonia : IComprar
    {
        public static bool GuardarImporte(Colonia catalinas)
        {
              ///Codigo...
        }
    }
}
```

Bases de datos / SQL

A fines didácticos solamente se ha el alta, la modificacion y el borrado de los colonos en en base de daatos. Al crear un formulario principal toma los datos de los colonos que obtiene de base de datos, dándo vida al atributo colonia con sus grupos y sus colonos.
Los productos son hardcodeados para poder realizar pruebas. “Se simula” el guardado del saldo y los pagos hechos por cada uno de los colonos a través de los archivos, ya que sino se guardasen en los mismos, esta información solamente viviría en memoria.

THREADS

Ha de utilizarse para modificar el aspecto visual del formulario apenas arranca el formulario principal.

EVENTOS

```
 public partial class frmVenta : Form
    {

        public event DelegadoProductos<Producto> EventoStock;


         private void btnAceptar_Click(object sender, EventArgs e)
        {

           //Lanzar el evento que controla el stock.
           string existencia = this.EventoStock(this.producto);

         }

     }

namespace Entidades
{
    public class Colonia : IComprar
    {

       public string AnalizaStock(Producto producto)
       {
              //Codigo
       }
    }
}
```

El delegado es utilizado para cargar el método manejador del evento que se declara en la clase Colonia. Este método analizará el stock de productos y retornará una cadena con la descripción del mismo cada vez que sea realice una compra. Es decir que si en stock existen 2 Gorritos rojos y se compra uno, mostrará un mensaje diciendo que solo queda un gorrito rojo.

MÉTODO DE EXTENSIÓN

```
namespace Entidades
{
    public static class ExtensionColonoPagaCuota
    {
         public static void PagarDeudas(this Colono claseColono, Colono colono, Colonia catalina)
        {
              ///Codigo...
        }
    }
}
```

El mismo realiza una extensión de la Clase Colono. Ha sido implementado para abonar la cuota (y elementos que compre el colono) bajando su deudo a cero y aumentando el saldo de la colonia. Además escribirá en un archivo quién ha realizado el pago y por qué valor.

FUNCIONALIDADES PARA UN SISTEMA DE VENTA

Durante el trabajo práctico se ha realizado el alta de Colonos, Gorritos y Antiparras.
Los colonos formarán parte de los grupos de la colonia:

-Entre 3 y 6 años : “Peques”.
-Entre 7 y 10 años: “Medianos”
-Entre 11 y 13 años “Grandes”.

Si no existe ningún colono en la colonia, el primer grupo que se creará será el de el colono que primero ingrese. Si el segundo colono que ingresa, pertenece al grupo del primer colono, este se agregará a su grupo, sino se creará uno nuevo, etc. También cada alumno pagará una cuota según el período en el que se anote a la colonia.

-Periodo mes: Valor $10.000.-
-Periodo quincena: Valor $6.000.-
-Periodo semana: Valor $3.500.-

Este valor actualizará al crear un nuevo colono mediante el método que se encuentra en la clase Colono, llamado CalcularDeuda(periodo).

Además se ha creado otro sistema de venta en el que efectivamente se pueden vender “Gorritas” y “Antiparras”. En el mismo el usuario podrá dar de alta nuevas “Gorritas” o nuevas “Antiparras”. Si el producto ingresante coincide con alguno de los que ya se encuentran en stock, estos se sumarán en cantidades y ocuparán “un bloque” de productos en stock, En caso de que no coincidan, se agregará como nuevo producto, ocupando un “nuevo bloque”. La cantidad de “bloques” de stock tiene un total de 5 disponibles, es decir que si se llega al máximo de bloques ocupados, no dejará continuar cargando nuevos productos y habrá que vender algún bloque entero (todos los gorritos amarillos o todas las antiparras verdes, etc).