

## **Lautaro Nicolás Fernández 2°A**

### **Trabajo práctico N°3.**

#### **Descripción del programa:**

Mi programa está pensado como un software que manejan los empleados del sector de “Atención al Cliente” de una compañía de servicio de telefonía móvil, el cual permite la modificación directa de una tabla de base de datos SQL. En esta tabla se agregan, modifican, buscan y eliminan clientes, en cuanto así lo necesite el empleado.

Consiste en un menú principal en el cual al desplegarlo nos encontramos con varias herramientas útiles, como la lista de los clientes que se encuentran abonando un plan mensual.

#### **Lista de Clientes:**

	Id	Documento	Nombre	Apellido	Plan	PlanInt
▶	1	29900139	Lionel	Scaloni	Empresa	1004
	2	39872123	Julian	Alvarez	Basico	1000
	3	20180912	Marcelo	Gallardo	Empresa	1004
	4	25645823	Lionel	Messi	Familiar	1003
	5	35912018	Gonzalo	Montiel	Premiun	1002
	6	28909010	Emiliano	Martinez	Premiun	1002
	7	28091872	Walter	Erviti	Inicial	1001
	8	26762435	Javier	Zanetti	Premiun	1002
	9	19972134	Claudio	Caniggia	Familiar	1003
	10	21983472	Gabriela	Sabattini	Familiar	1003
	11	18765423	Marcela	Morelo	Premiun	1002
	12	38945602	Emilia	Mernes	Premiun	1002
	13	16919999	Gloria	Estefan	Premiun	1002
<	14	12345678	Carmen	Barbieri	Premiun	1002

Por otro lado, tenemos un listbox donde se muestran los planes que tiene la compañía, a modo de facilitarle al usuario la consulta que tenga el cliente.

## Servicios ofrecidos:

CODIGO:#1000
Plan basico:2gb + 1gb Whatsapp (\$1000)
CODIGO:#1001
Plan inicial:5gb + Whastapp FREE(\$1500)
CODIGO:#1002
Plan premium:GB ilimtadas + Whastapp + Llamadas FREE(\$3000)
CODIGO:#1003
Pack Familiar:4 LINEAS + Pack inicial(\$4800)
CODIGO:#1004
Pack Empresa:30 LINEAS + Pack basico + Llamadas FREE entre flota(\$26000)

También tiene un buscador de clientes, el cual ingresándole un numero de cliente valido nos mostrara la información de este y nos permitirá la opción de manipularlo o no.

### Busqueda de Cliente

Ingresa numero de Cliente:

ID:1  
Nombre:Lionel  
Apellido:Scaloni  
DNI:29900139  
Plan:Empresa

Acceder a Gestion de Clientes

Desea gestionar el cliente encontrado?

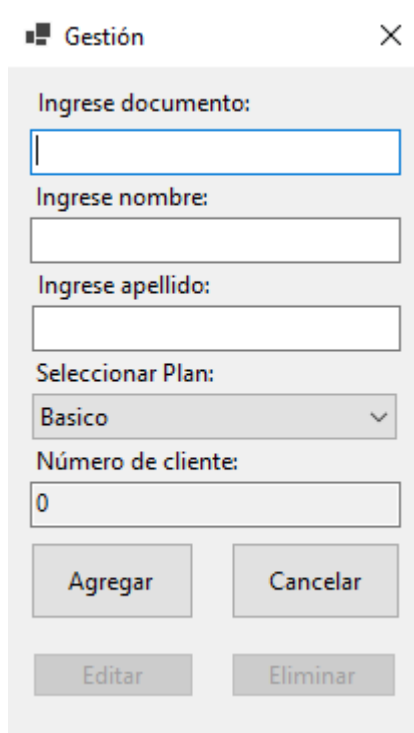
También contamos con el apartado de “Agregar Cliente”:

### Añadir Cliente

Podremos acceder a un menú de gestión de cliente al querer manipular un cliente encontrado o al clicar sobre el botón del apartado previo. Este menú se cargara de una forma u otra dependiendo lo que queramos hacer, por lo que si lo que buscamos es añadir un cliente, estarán deshabilitadas las opciones de modificar y eliminar. Pero si lo que queremos es manipular un cliente, lo que se encontrara bloqueado es la

opción de Añadir uno y nos cargará los campos con el cliente que hayamos encontrado.

Al agregar:

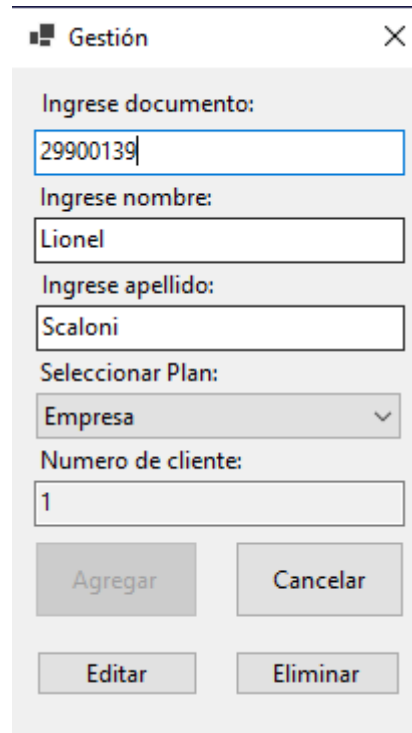


Formulario de gestión de clientes al agregar un nuevo cliente. El formulario tiene un título 'Gestión' y un botón de cerrar 'X'. Los campos son:

- Ingrese documento: (campo vacío)
- Ingrese nombre: (campo vacío)
- Ingrese apellido: (campo vacío)
- Seleccionar Plan: (menú desplegable con 'Basico' seleccionado)
- Número de cliente: (campo con '0')

Botones: 'Agregar', 'Cancelar', 'Editar', 'Eliminar'.

Al manipular:



Formulario de gestión de clientes al manipular un cliente existente. El formulario tiene un título 'Gestión' y un botón de cerrar 'X'. Los campos son:

- Ingrese documento: (campo con '29900139')
- Ingrese nombre: (campo con 'Lionel')
- Ingrese apellido: (campo con 'Scaloni')
- Seleccionar Plan: (menú desplegable con 'Empresa' seleccionado)
- Numero de cliente: (campo con '1')

Botones: 'Agregar', 'Cancelar', 'Editar', 'Eliminar'.

Al agregar en Numero De Cliente se mostrara 0, no por un error sino porque el programa genera automáticamente el Id, siendo este el siguiente al último cargado en la lista.

Por ultimo pero no menos importante, tendremos la opción de “Guardar Datos”, la cual sin interrumpir el hilo principal se encargara de guardar la lista de clientes en un archivo “.xml”, a modo de tener un Backup de los datos, si así lo desea el empleado.

Guardar Datos

Es una tarea “costosa”, por lo cual luego de unos segundos mostrara por pantalla un mensaje de que pudo guardar los datos como ‘datos.xml’.

Como último agregado, contiene un checkbox el cual alternara el modo oscuro del programa. Muy básico pero funcional.

☐ DarkMode

## Donde se utilizan los temas?

**Manejo de SQL:** dentro de la biblioteca de Entidades, se encuentra la clase ClienteDAO, la cual permite mediante distintas query el abm del programa.

**Delegados:** aplique varios en todo el programa ya que los encontré útil, por ejemplo, para asignar mensajes en bibliotecas de clase. En este caso, en ClienteDAO algunos métodos reciben un parámetro de tipo GeneradorMensaje, el cual recibe un string. Esto permite que luego en el formulario ese delegado encapsule al método “GeneradorDeMensaje”, el cual imprime en un MessageBox.Show la cadena que le pasemos por parámetro.

**Eventos:** tengo solamente dos eventos en todo el programa. Uno de ellos es para alternar entre el modo oscuro en el formulario principal. (En realidad son dos ya que uno cambia a modo oscuro y el otro vuelve al modo claro).

```
1 referencia
private void chb_darkMode_CheckedChanged(object sender, EventArgs e)
{
    if (chb_darkMode.Checked)
    {
        if (cambiarModoOscuro is not null)
        {
            cambiarModoOscuro.Invoke();
        }
    }
    else
    {
        if (cambiarModoClaro is not null)
        {
            cambiarModoClaro.Invoke();
        }
    }
}
```

Tanto cambiarModoOscuro como cambiarModoClaro modificaran los colores de los labels y del fondo del programa.

El otro evento se encuentra en el formulario de gestión de clientes (Frm\_ABM), el cual cuando solicitamos la llamada al formulario para manipular un cliente encontrado, se encargara de asignar los botones, ya que el formulario por defecto solo trae habilitado el botón para agregar un cliente. Por ende este evento deshabilitara el mencionado y dejara disponibles los botones de Eliminar y Modificar. El evento se llama ‘asignarBotonesGestion’.

1 referencia

```
public Frm_ABM(Cliente cliente)
{
    InitializeComponent();
    this.txt_Documento.Text = cliente.Documento.ToString();
    this.txt_Nombre.Text = cliente.Nombre;
    this.txt_Apellido.Text = cliente.Apellido;
    CargarComboBox(cliente.Plan);
    this.txt_nroCliente.Text = cliente.Id.ToString();
    this.lb_nroCliente.Text = "Numero de cliente:";

    asignarBotonesGestion += BotonesGestion;

    asignarBotonesGestion.Invoke();
}
```

**Hilos:** utilice un metodo asincrónico que guardará, mediante una serializacion xml, los datos que contenga la lista. Pensé en realizarlo en otro hilo para poder seguir realizando tareas mientras tanto. Al finalizar la carga mostrara por pantalla un mensaje de éxito junto con el nombre del archivo creado.

1 referencia

```
public static async Task<string> SerializarDatos(GuardadorDeDatos guardadorDeDatos)
{
    string informacion = await Task.Run(() =>
    {
        Thread.Sleep(3000); // Con fines didacticos.
        guardadorDeDatos.Invoke();

        return "Se han guardado los datos como:\n\t'datos.xml'!";
    });

    if(informacion.Length < 0)
    {
        throw new Exception("No se han podido guardar los datos!");
    }

    return informacion;
}
```

Este método por el evento click del botón del formulario:

```
1 referencia
private async void btn_GuardarNuevosDatos_Click(object sender, EventArgs e)
{
    try
    {
        string retorno = await SerializarDatos(GuardarXML);
        MessageBox.Show(retorno);
    }
    catch(Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}
```

**Excepciones:** cree dos excepciones propias para manejarlas, aparte de manejar todas las posibles. Una de ellas es 'ClientFoundException', la cual se lanzara al momento de querer agregar un cliente, si este ya se encuentra agregado a la lista.

```
if (!existe)
{
    clientesDB.Agregar(clienteAgregar, GeneradorDeMensaje);
    ActualizarDatos();
}
else
{
    throw new ClientFoundException("El cliente ya se encuentra cargado!");
}
```

La otra es 'PathIsEmptyException', la cual se lanzara si al momento de cargar el txt de servicios, el path es un string vacio.

```
if (path != string.Empty)
{
    using (StreamReader sr = new StreamReader(path))
    {
        String servicio;

        while ((servicio = sr.ReadLine()) != null)
        {
            servicios.Text += string.Concat($"{servicio}\n");
        }
    }
}
else
{
    throw new PathIsEmptyException("La ruta esta vacia!");
}
```

**Interfaces y clases genericas:** como ya lo había desarrollado en el tp3, estos temas se encuentran tanto en la clase 'IManejadorDeArchivos' (interfaz) y 'SerializadorXML' (clase genérica) que sirven para poder cumplir con la serializacion.

**Métodos de extensión:** clase StringMessages de la biblioteca entidades. Genera dos mensajes que se utilizaran en el formulario principal.

**Unit Testing:** desarolle un proyecto de tipo MSTEST que contiene dos clases. Una es ClienteDAOTest, que testea la función BuscarCliente de la clase ClienteDAO.

La otra es ValidacionesTest, que se encarga de realizar multiples validaciones de las funciones que contiene la clase Validaciones, que justamente validan la carga de datos al momento de editar, modificar un cliente.

**Archivos:** tanto en el formulario principal al momento de cargar el '.txt' de los servicios, como en la clase previamente comentada que serializa en '.xml'.

**Expresiones lambda:** evento click del botón buscar y agregar.

```
id = int.Parse(txt_nroCliente.Text);  
existe = listaClientes.Exists(cliente => cliente.Id == id);
```

**Si llega a faltar algún tema:**



*Fernández, Lautaro Nicolás.*

*2°A.*

*PD: el script se encuentra suelto en la carpeta; y el archivo de serialización no está creado, se crea al momento de solicitarlo en el programa para que sea más realista.*