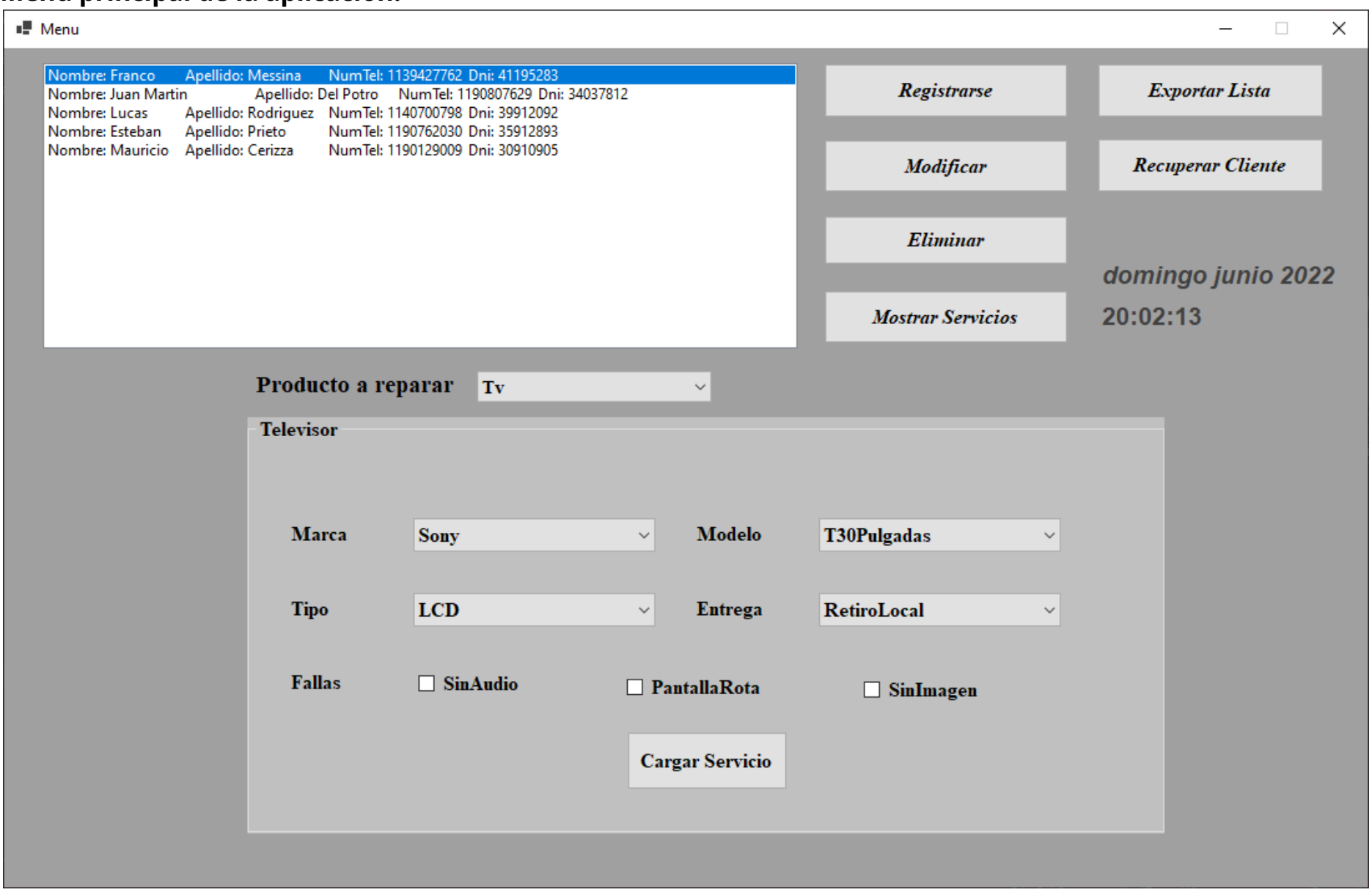
FRANCO DAVID MESSINA, DIV 2E
TP N3 LABORATORIO II

Introducción:

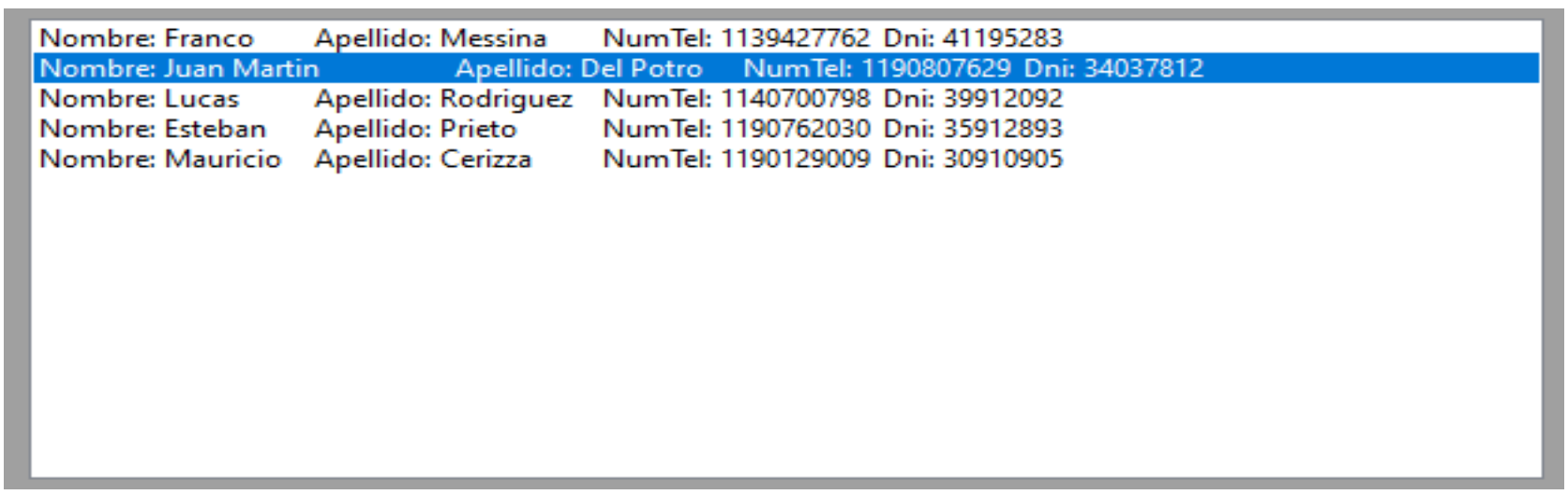
La aplicación se creó para un local de Ramos Mejias en el cual hacen servicios de Televisores, Controles(TV, Aire acondicionado) y Aire Acondicionado.

La aplicación administra una lista de clientes con sus datos respectivamente y los servicios pedidos.

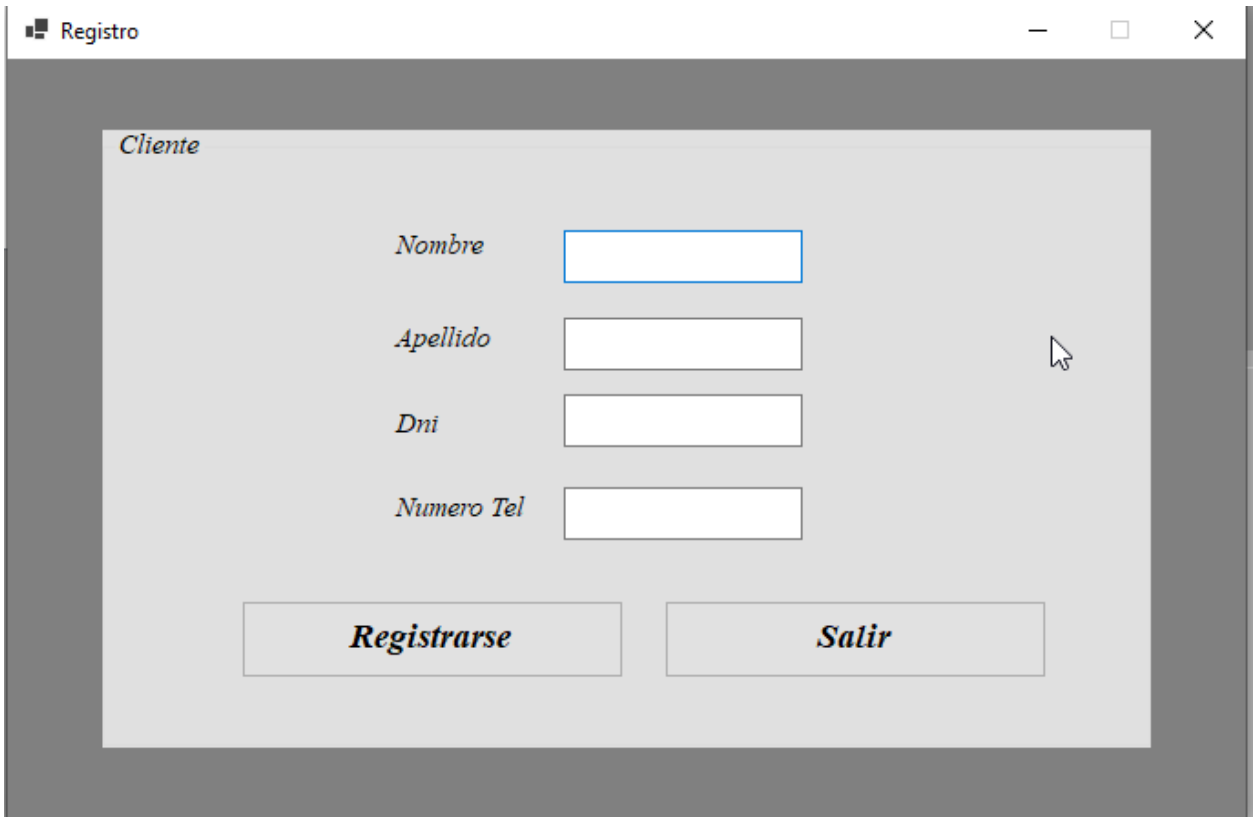
Menú principal de la aplicación.



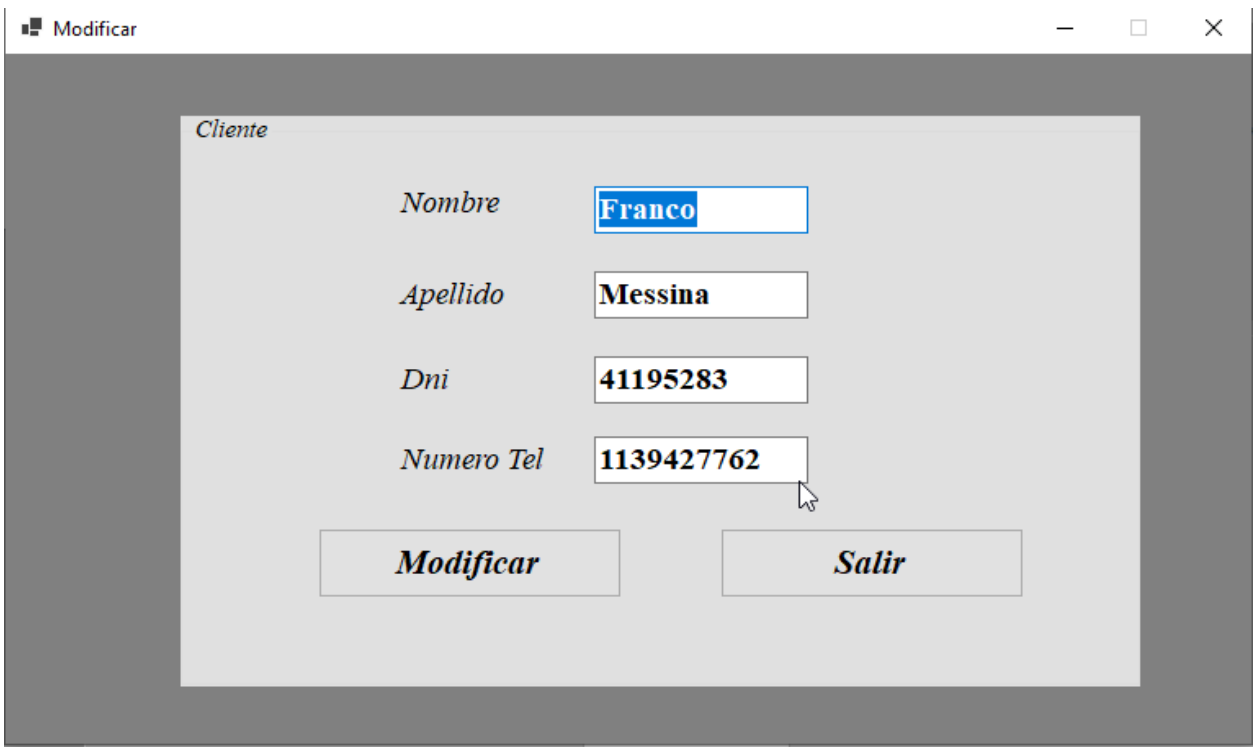
Consta de una listbox donde se puede visualizar los clientes registrados y que están dados de alta.



Registrar Cliente: El cliente deberá ingresar su nombre, apellido, DNI, y número de teléfono.

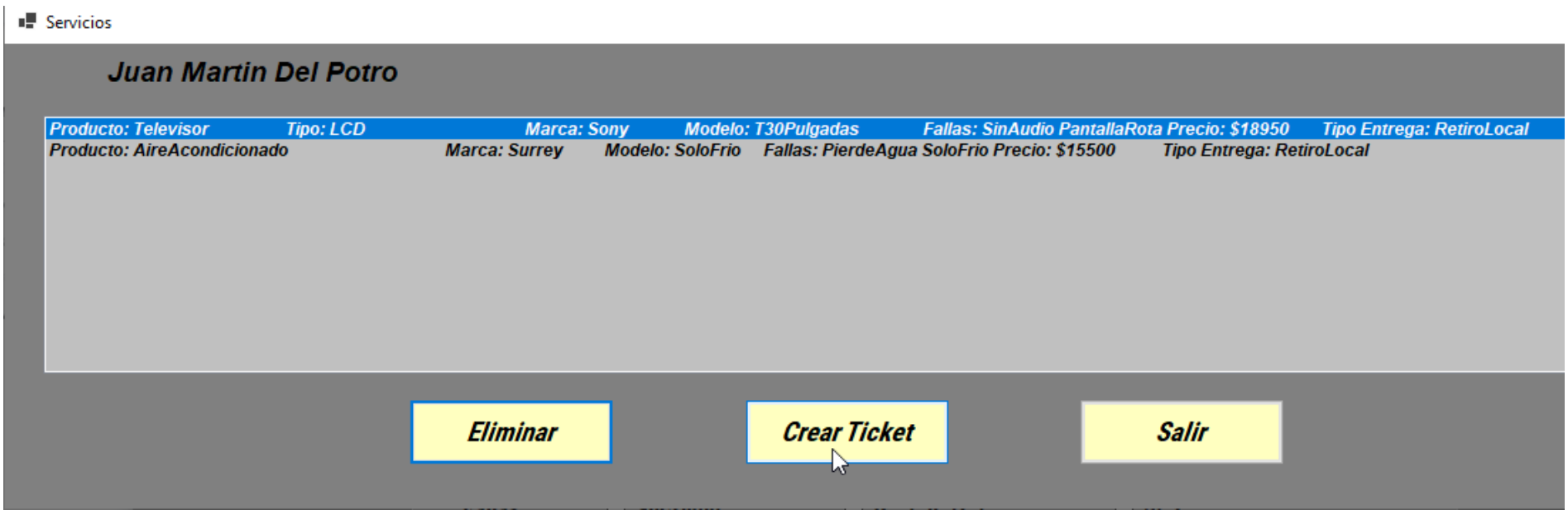


Modificar Cliente: Se selecciona un cliente de la lista y cuando se aprieta el botón modificar se abre un nuevo formulario con todos los datos cargados del cliente seleccionado.



Eliminar Cliente: Se selecciona un cliente de la lista y al apretar el botón eliminar se hace una baja lógica. Esto quiere decir que sigue existiendo en el sistema pero no se visualiza en la lista.

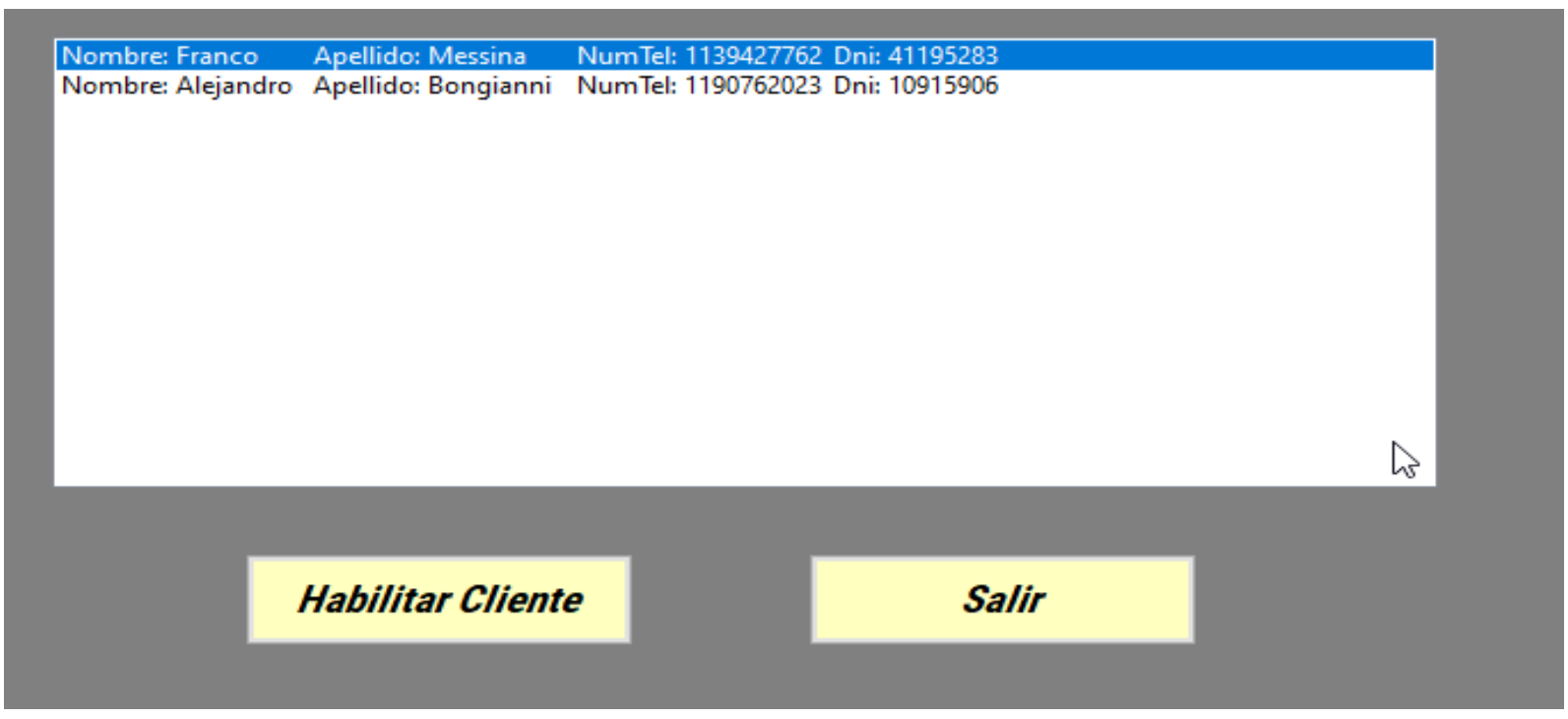
Mostrar Servicios: Se selecciona un cliente de la lista y al apretar el botón Mostrar se visualizará todos los servicios de ese cliente.



Eliminar Servicio: Se selecciona el servicio y al apretar el botón se eliminará.

Crear Ticket: Se selecciona un servicio y al apretar el botón va a generar un TXT en el escritorio->Archivos->TXT-> Nombre,Apellido del cliente y horario actual con todos los datos del servicio.

Recuperar Cliente: Se van a visualizar todos los clientes eliminados, es decir, que existieron alguna vez pero al día de hoy están dados de baja. Para recuperarlo se selecciona el cliente y se lo vuelve a habilitar.



Exportar Lista: Exporta la lista de clientes con todos sus datos personales y servicios en formato XML (en la ruta de escritorio->Archivos->XML->ListaDeClientes.XML).

Cargar Servicio: Se selecciona un cliente se elige el producto a reparar con todas sus características y al apretar el botón se carga el servicio en la lista de servicios del cliente seleccionado.

Lista de temas.

Excepciones: Fue utilizado para el manejo de validaciones en todos los datos que se utilizan para registrar un cliente, modificar, y eliminar. También para validar si existe un servicio, cliente, y como último para excepciones de archivos.

Por ejemplo:

Se lanza la excepción

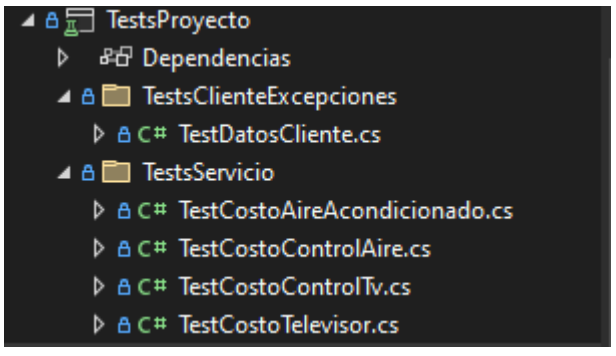
```
/// <summary> Valida todos los campos del cliente verificando si son null o vaci ...
6 referencias
public static bool ValidarCamposCliente(string nombre, string apellido, string dni, string numeroTel)
{
    if (string.IsNullOrWhiteSpace(nombre) || string.IsNullOrWhiteSpace(apellido) || string.IsNullOrWhiteSpace(dni)
        || string.IsNullOrWhiteSpace(numeroTel))
    {
        throw new CamposVaciosONullException("Cargar todos los datos del cliente!");
    }
    return true;

}
/// <summary> Valida nombre.
5 referencias
public static bool ValidarNombre(string nombre)
{
    if (nombre.Length > 20 || nombre.Length < 3)
    {
        throw new NombreInvalidoException("El nombre tiene que tener más de 3 letras y menos de 20");
    }
    else if (!VerificarContieneSoloLetras(nombre))
    {
        throw new NombreInvalidoException("El nombre tiene que contener solo letras.");
    }
    return true;
}
```

Se captura

```
try
{
    if (ValidarCliente.ValidarCamposCliente(txtNombre.Text, txtApellido.Text, txtDni.Text, txtTelefono.Text))
    {
        if (ValidarCliente.ValidarNombre(txtNombre.Text) && ValidarCliente.ValidarApellido(txtApellido.Text)
            && ValidarCliente.ValidarDni(txtDni.Text) && ValidarCliente.ValidarNumeroTel(txtTelefono.Text))
        {
            string mensaje = string.Empty;
            Cliente nuevoCliente = new Cliente(txtDni.Text, txtNombre.Text, txtApellido.Text, txtTelefono.Text);
            mensaje = Administracion.ModificarCliente(gestionServicios, cliente, nuevoCliente);
            MessageBox.Show(mensaje,"Modificacion",MessageBoxButtons.OK,MessageBoxIcon.Information);
            this.DialogResult = DialogResult.OK;
        }
    }
}
catch (CamposVaciosONullException ex)
{
    MessageBox.Show(ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Warning);
}
catch (ClienteNoExistenteException ex)
{
    MessageBox.Show(ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Warning);
}
catch (NombreInvalidoException ex)
{
    MessageBox.Show(ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Warning);
}
catch (DniInvalidoException ex)
{
    MessageBox.Show(ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Warning);
}
```

Pruebas Unitarias: Se testean los costos de los servicios, cuando un cliente tiene un servicio o varios.

También se testean todas las excepciones de los datos del cliente. En el proyecto hay un total de 36 pruebas.



Ejemplos:

```
[TestMethod]
0 referencias
public void CalcularCosto_DeberiaRetornar6000_CuandoAireTieneFallaSoloFrio()
{
    //Arrange
    List<string> listaFallas = new List<string>();
    listaFallas.Add(EFallaAire.SoloFrio.ToString());
    AireAcondicionado aireAcondicionado = new AireAcondicionado(EMarcaAire.Surrey.ToString(),
        EMarcaAire.Samsung.ToString(),listaFallas);
    float valorEsperado = 6000;

    //Act
    float valorActual = aireAcondicionado.CalcularCosto();

    //Assert
    Assert.AreEqual(valorEsperado, valorActual);
}
```

```
[TestMethod]
[ExpectedException(typeof(NombreInvalidoException))]
0 referencias
public void ValidarNombre_CuandoRecibeMenosDe3Letras_DeberiaLanzarNombreInvalidoException()
{
    //Arrange
    string nombre = "fr";

    //Act
    bool actual = ValidarCliente.ValidarNombre(nombre);

    //Assert

}
[TestMethod]
[ExpectedException(typeof(NombreInvalidoException))]
0 referencias
public void ValidarNombre_CuandoRecibeMasDe20Letras_DeberiaLanzarNombreInvalidoException()
{
    //Arrange
    string nombre = "carodasjskadjkaskjsakjadjkjkasjkddjkaskjsa";

    //Act
    bool actual = ValidarCliente.ValidarNombre(nombre);

    //Assert
}
```

Genéricos: Fue utilizado en la interfaz Archivos para poder emplear esta interfaz en otras clases aunque se manejan distintos tipos de datos, es decir, en la clase Archivo TXT se utiliza string y en Serializador otro genérico.

De esta forma utilizando el Método Escribir que contiene un genérico como parámetro se puede escribir un ticket en formato txt (Tipo de dato: String), y exportar una lista de clientes (Tipo de dato: List<Cliente>).

También se retorna un genérico en el método Leer (clase Serializador) para poder leer cualquier tipo de dato.

Clase ArchivoTXT

```
public string Escribir(string nombreArchivo, string contenido)
{
    try
    {
        string ruta = $"{rutaBase}\\{nombreArchivo}{DateTime.Now.ToString("D")}.txt";
        using (StreamWriter streamWriter = new StreamWriter(ruta))
        {
            streamWriter.WriteLine(contenido);
        }
        return "Ticket generado correctamente";
    }
    catch (Exception)
    {
        throw new ArchivoTxtException("Error al guardar los datos en formato TXT");
    }

}
```

Clase Serializador

```c#
public string Escribir(string nombreArchivo, T elemento)
{
    try
    {
        if(this.tipoArchivo == ETipoArchivo.XML)
        {
            if (Path.GetExtension(nombreArchivo) == ".xml")
            {

                string rutaFinal = $"{rutaBase}\\{nombreArchivo}";

                using (XmlTextWriter xmlTextWriter = new XmlTextWriter(rutaFinal,Encoding.UTF8))
                {
                    xmlTextWriter.Formatting = Formatting.Indented;
                    XmlSerializer serializer = new XmlSerializer(typeof(T));
                    serializer.Serialize(xmlTextWriter, elemento);
                }
                return "Archivo guardado con exito";
            }
            else
                throw new ArchivoSerializacionException("Extension invalida se esperaba XML.");
        }
        return "Formato invalido";
    }
    catch (Exception)
    {

        throw new ArchivoSerializacionException("Error al serializar.");
    }
}
```

Interfaz: Fue utilizado para crear la interfaz archivos donde voy a tener un método llamado Escribir que voy a utilizar en Archivo TXT y en Serializador. No agregue el método Leer en la interfaz porque no leo ningún archivo en formato TXT.

Se utiliza una interfaz porque la clase Serializador y ArchivoTXT no tienen ninguna relación.

```c#
2 referencias
public interface IArchivos<T> where T : class
{
    /// <summary>
    /// Metodo que va a escribir un dato generico.
    /// </summary>
    /// <param name="nombreArchivo">Nombre del archivo que vamos a Escribi</pa
    /// <param name="elemento">nombre del Elemento a escribir</param>
    /// <returns>Retornará un mensaje dependiendo el caso</returns>
    4 referencias
    string Escribir(string nombreArchivo, T elemento);
}
```

Archivos: Se utiliza en el formulario de Mostrar Servicios para poder crear un ticket en formato TXT por cada servicio que el cliente quiera.

```
private void btnCrearTicket_Click(object sender, EventArgs e)
{
    try
    {
        if (ValidarExisteServicio())
        {
            Servicio servicio = (Servicio)lstServicios.SelectedItem;
            ArchivoTXT archivoTexto = new ArchivoTXT();
            string mensaje = archivoTexto.Escribir($"{cliente.Nombre}{cliente.Apellido} ", Servicio.GenerarTicket(cliente, servicio));
            MessageBox.Show(mensaje, "Ticket generado", MessageBoxButtons.OK, MessageBoxIcon.None);
        }
    }
    catch (ServicioNoExistenteException ex)
    {
        MessageBox.Show(ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Warning);
    }
    catch (Exception)
    {
        MessageBox.Show("Error!");
    }
}
```

Serialización: Se utiliza para poder importar una lista de clientes en formato XML y para poder exportar una lista de clientes en formato XML.

Se utiliza en el load para que el empleado no tenga que importar sino que se importe automáticamente de la ruta preestablecida y que pueda exportar apretando el botón “Exportar Lista”.

```
1 referencia
private void btnGuardarClientes_Click(object sender, EventArgs e)
{
    try
    {
        Serializador<List<Cliente>> serializador = new Serializador<List<Cliente>>(GestorDeArchivo.ETipoArchivo.XML);
        string mensaje = serializador.Escribir("ListaClientes.xml", gestionServicios.ListaClientes);
        MessageBox.Show(mensaje,"Archivo",MessageBoxButtons.OK,MessageBoxIcon.Information);
    }
    catch(ArchivoSerializacionException ex)
    {
        MessageBox.Show(ex.Message);
    }
    catch (Exception)
    {

        MessageBox.Show("Error al serializar");
    }

}
```

Como recomendación:

Se dejará una carpeta llamada Archivos donde contiene una carpeta con el nombre XML que contiene una lista de clientes para que sea más fácil evaluar. Donde para poder utilizarla se tendrá que guardar en el escritorio.

La carpeta Archivos tiene que estar en el escritorio. En caso de no hacerlo cuando se exporte una lista o se genere un ticket se creará sola.