

## Trabajo practico N°3 Laboratorio II



**Nombre y apellido:** Gabriel Alegre.

**Nota:** la explicación de cómo y dónde implemente los temas vistos entre la clase 10 y 15 es a partir de la página 7, igualmente, se recomienda leer lo anterior para entender el funcionamiento del programa.

NetCom es una empresa de telecomunicaciones. Su oferta de servicios incluye la provisión de servicios de televisión por cable, telefonía fija e Internet.

Los planes disponibles que se les ofrecerá a los clientes cuando se quieran dar de alta son los siguientes:

**Plan básico:**

Cantidad de megas de internet: 50

Incluye Cable: Si

Incluye Telefonía fija: No

Incluye Fibra óptica: No

Precio final del plan: 2000

**Beneficios por plan básico:**

- pack HBO gratis

**Plan intermedio:**

Cantidad de megas de internet: 100

Incluye Cable: Si

Incluye Telefonía fija: Si

Incluye Fibra óptica: No

Precio final del plan: 3500

**Beneficios por plan intermedio:**

- Disney+ gratis

- Paramount plus gratis

**Plan premium:**

Cantidad de megas de internet: 1000

Incluye Cable: Si

Incluye Telefonía fija: Si

Incluye Fibra óptica: Si

Precio final del plan: 6000

**Beneficios por contratar el plan premium:**

- Netflix gratis

- Amazon prime video gratis

- Pack futbol gratis

**Importante antes de arrancar:** La especificación de la ruta donde están los archivos Xml, Json y txt está al final de la página 13.

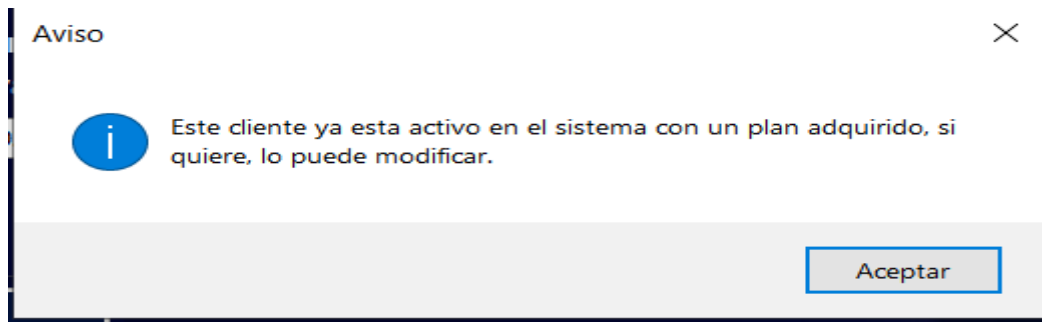
Explicación de la funcionalidad del programa y como interactúa con cada botón:

#### Vamos a empezar por el Alta:

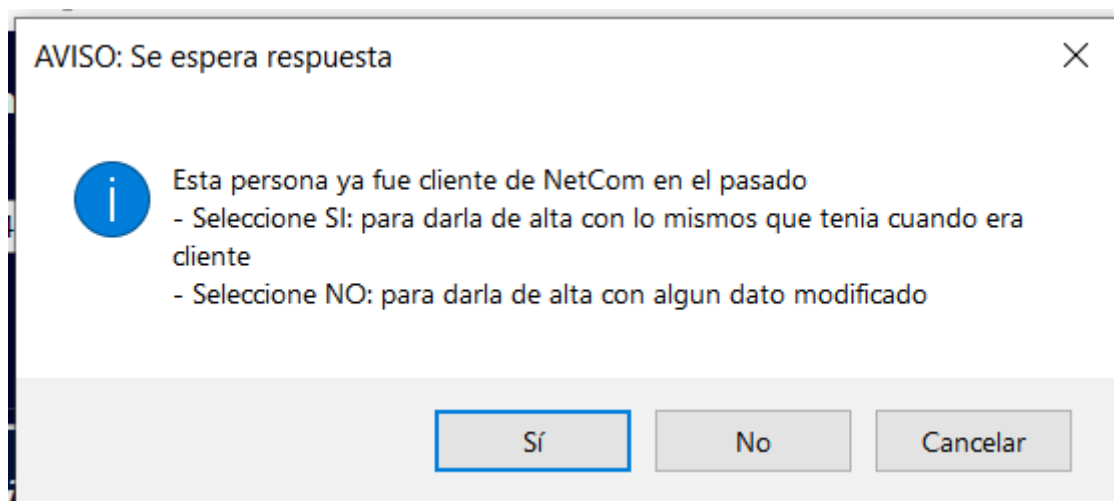
Al presionar el botón de alta nos pedirán el documento del cliente que se quiera dar de alta, al momento de ingresarlo (el documento) hay tres opciones:

1) Que el DNI corresponda a una persona que no esté en la lista de clientes, es decir, que sea un cliente totalmente nuevo y en tal caso se nos abrirá el siguiente form de alta para poder instanciar el cliente.

2) Que el DNI corresponda a un cliente que ya este dado de alta en el sistema y en tal caso el programa nos avisara con el siguiente mensaje:



3) Que el DNI corresponda a un cliente que no activo (esta en mi sistema, en mi lista de cliente pero con baja logica), es decir, un cliente que en algún momento se dio de baja, y ahora quiere volver a darse de alta, en tal caso aparecerá el siguiente mensaje:

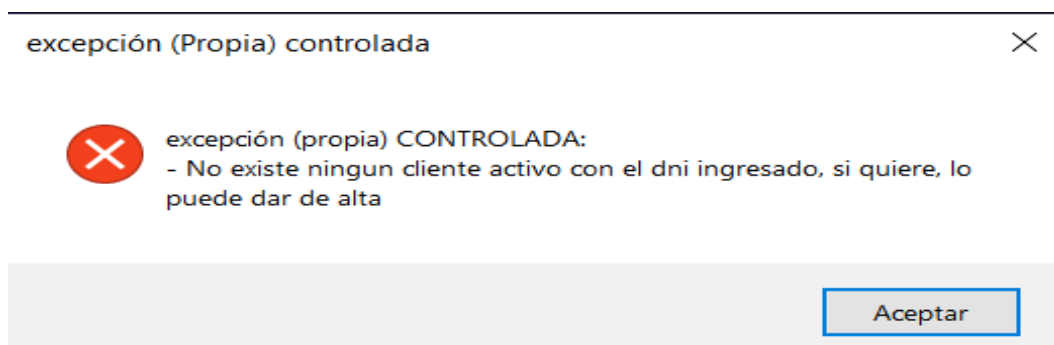


Si selecciona SI el cliente se dará de alta automáticamente con los mismos datos que tenía al momento de darse de baja. Al seleccionar NO, se abrirá un nuevo form para dar de alta al cliente permitiendo modificar alguno de sus atributos.

#### **Botón Modificar:**

Al presionar el botón modificar, se pedirá que se ingrese el DNI del cliente a modificar y verificara que dicho DNI ingresado realmente pertenezca a un cliente activo en el sistema (En la lista)

Si el DNI NO pertenece a un cliente activo, se avisará con el siguiente mensaje:



En caso contrario, es decir, que el DNI ingresado pertenezca a un cliente activo del sistema, se abrirá el siguiente formulario autocompletado con los datos del cliente, una vez que se hayan modificado los datos correspondientes, a la derecha, esta la opción de ver como quedaría el usuario con dichas modificaciones.

Modificación de datos

Estos son los datos actuales del cliente, modifique lo que corresponda

Nombre

Gabriel

Apellido

Alegre

Dirección

Av Mitre 750

Dni ingresado

43901903

El plan seleccionado actualmente es el que tenía el cliente, puede modificarlo

Planes disponibles

☐ Plan básico

☐ Plan intermedio

☐ Plan premium

Modificar

Cancelar

visibilidad

☐ Mostrar cliente con los datos originales

☐ Mostrar como quedaría el cliente con las modificaciones

Nombre: Gabriel

Apellido: Alegre

Domicilio: Av Mitre 750

DNI: 43901903

Numero de cliente: 3

Plan adquirido:

Detalles del plan premium

Cantidad de megas de internet: 1000

Incluye Cable: Si

Incluye Telefonía fija: Si

Incluye Fibra óptica: Si

Precio final del plan: 6000

Beneficios por plan premium:

- Netflix gratis

- Amazon prime video gratis

### Botón Baja:

Al presionar el botón de dar de Baja, se pedirá que se ingrese el DNI del cliente se quiere dar de baja y verificara que dicho DNI ingresado realmente pertenezca a un cliente activo en el sistema (En la lista)

Si el DNI NO pertenece a un cliente activo, se avisará con el siguiente mensaje:


excepción (Propia) controlada

excepción (propia) CONTROLADA:

- No existe ningún cliente activo con el dni ingresado, si quiere, lo puede dar de alta

Aceptar

En caso contrario, es decir, que el DNI ingresado pertenezca a un cliente activo del sistema, se abrirá el siguiente formulario, si se presiona el botón aceptar se dará de baja automáticamente al cliente.

 Dar baja ✕

**Se dara el siguiente cliente de baja, esta seguro?**

Nombre: Gabriel  
Apellido: Alegre  
Domicilio: Calle falsa 123  
DNI: 43901903  
Numero de cliente: 4  
Plan adquirido:  
Detalles del plan basico  
Cantidad de megas de internet: 50  
Incluye Cable: Si  
Incluye Telefonía fija: No  
Incluye Fibra optica: No  
Precio final del plan: 2000

Beneficios por plan basico:  
- pack HBO gratis

**Aceptar**

### **Botón Mostrar Clientes:**

Al presionar el botón de mostrar clientes se abrirá el siguiente formulario, abajo, está la opción de mostrar los clientes activos o lo clientes no activos (Los dados de baja).

 Listado de clientes ✕

**Listado de clientes**

Listado de clientes activos:

Nombre: Daenerys  
Apellido: Targaryen  
Domicilio: Italia 874  
DNI: 44902904  
Numero de cliente: 1  
Plan adquirido:  
Detalles del plan premium  
Cantidad de megas de internet: 1000  
Incluye Cable: Si  
Incluye Telefonía fija: Si  
Incluye Fibra optica: Si  
Precio final del plan: 6000

Beneficios por plan premium:  
- Netflix gratis  
- Amazon prime video gratis  
- Pack futbol gratis

Nombre: Jon  
Apellido: Snow  
Domicilio: Av mitre 750  
DNI: 45902905  
Numero de cliente: 2

☐ Mostrar clientes activos ☒ Mostrar clientes dados de baja

**Aceptar**

### **Botón Historial de operaciones:**

Al presionar el botón de historial de operaciones se abrirá el siguiente formulario que a partir de un archivo txt (la explicación de cómo y dónde lo implemente esta más adelante) que guarda todas las operaciones realizadas en forma de bitácora y cuando se presiona el botón y lo levanta, lo lee, y lo muestra:

■ Registro de operaciones

✕

**Historial/registro de operaciones**

Adquirio el PlanBasico

domingo, 5 de junio de 2022 23:55hs - Se realizo el alta de un cliente:  
Llamado: Mercedes  
Con Dni: 47904907  
Adquirio el PlanIntermedio

domingo, 5 de junio de 2022 23:58hs - Se realizo el alta de un cliente:  
Llamado: Obi-Wan  
Con Dni: 43334242  
Adquirio el PlanPremium

lunes, 6 de junio de 2022 00:10hs - Se realizo la baja del cliente:  
Llamado: Gabriel  
Con Dni: 43901903

lunes, 6 de junio de 2022 00:10hs - Se realizo el alta de un cliente que anteriormente se habia dado de baja  
Llamado: Gabriel  
Con DNI: 43901903

lunes, 6 de junio de 2022 00:56hs - Se realizo una modificacion de datos al cliente:  
Llamado: Daenerys  
Con Dni: 44902904

Aceptar

### **Botón de los informes estadísticos:**

Al presionar el botón se abre el siguiente formulario el informe estadístico del programa, cantidad de altas dadas, cantidad de modificaciones realizadas, cantidad de bajas, etc.

■ Informe estadístico

✕

**Informe estadístico sobre los servicios**

- Cantidad de Altas: 7

- Cantidad de clientes que modificaron alguno de sus datos: 3

- Cantidad de clientes que se dieron de baja: 1

- Cantidad de clientes no activos (se dieron de baja) y luego se volvieron a dar de alta: 1

- Cantidad total de operaciones (alta, baja, modificacion) que se realizaron: 11

- Cantidad de personas cargadas en el sistema (Tanto las activas como las no activas): 6

- Cantidad de Clientes activos: 6

- Cantidad de Clientes NO activos: 0

Aceptar

## Temas vistos entre la clase 10 y 15 utilizados

### Excepciones:

#### Primera excepción:

```
6 referencias
public class NoExisteClienteActivoConElDniIngresadoException : Exception
{
    /// <summary>
    /// </summary>
    /// <param name="message">Mensaje que describe el porque se produjo la excepcion</param>
    1 referencia
    public NoExisteClienteActivoConElDniIngresadoException(string message) : this(message, null)
    {
    }

    /// <summary>
    /// </summary>
    /// <param name="message">Mensaje que describe el porque se produjo la excepcion</param>
    /// <param name="innerException">Inner exception de la excepcion</param>
    1 referencia
    public NoExisteClienteActivoConElDniIngresadoException(string message, Exception innerException) : base(message, innerException)
    {
    }
}
```

Cuando se quiera modificar o dar de baja a un cliente, se pedirá el ingreso del DNI para confirmar la existencia de dicho cliente en el servidor, en caso de que el DNI ingresado NO pertenezca a ningún cliente activo, se lanzara dicha excepción

Método: donde se lanza la excepción: VerificarSiElClienteExisteYEstaActivoEnElSistema de la clase: Central administradora. Línea: 142

```
2 referencias
public static bool VerificarSiElClienteExisteYEstaActivoEnElSistema(int documento)
{
    bool estaEnElSistemaActivo;

    if (BuscarClienteActivoPorDni(documento) is not null)
    {
        estaEnElSistemaActivo = true;
    }
    else
    {
        throw new NoExisteClienteActivoConElDniIngresadoException("excepción (propia) CONTROLADA:\n- No existe ningun cliente activo con el dni");
    }

    return estaEnElSistemaActivo;
}
```

Esta excepción será capturada (Si es que corresponde, es decir, si ingresan un DNI que NO pertenezca a ningún cliente activo) En el formulario: FormIngresoDeDni. En el Método: VerificarQueElClienteEsteEnElSistema. Línea: 106

```
private void VerificarQueElClienteEsteActivoEnElSistema()
{
    try
    {
        if (CentralAdministradora.VerificarSiElClienteExisteYEstaActivoEnElSistema(dni))
        {
            this.DialogResult = DialogResult.OK;
        }
    }
    catch (NoExisteClienteActivoConElDniIngresadoException e)
    {
        this.DialogResult = DialogResult.Cancel;
        MessageBox.Show(e.Message, "excepción (Propia) controlada", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

## Segunda excepción:

```
13 referencias
public class FallaDeArchivoException : Exception
{
    /// <summary>
    /// </summary>
    /// <param name="message">Mensaje que describe el porque se produjo la excepcion</param>
    7 referencias
    public FallaDeArchivoException(string message) : this(message, null)
    {
    }

    /// <summary>
    /// </summary>
    /// <param name="message">Mensaje que describe el porque se produjo la excepcion</param>
    /// <param name="innerException">Inner exception de la excepcion</param>
    1 referencia
    public FallaDeArchivoException(string message, Exception innerException) : base(message, innerException)
    {
    }
}
```

Esta excepción será lanzada al momento de querer guardar/serializar o leer/deserializar archivos y ocurra un fallo. Por ejemplo, una de las implementaciones es:

Clase: Central administradora. Métodos Guardar, Leer. Línea: 163 y 183 respectivamente.

```
5 referencias
public void Guardar(string ruta, string contenido)
{
    try
    {
        using (StreamWriter sw = new StreamWriter(ruta, true))
        {
            sw.WriteLine(contenido);
        }
    }
    catch (Exception)
    {
        throw new FallaDeArchivoException("Excepcion propia controlada:\n falla querer guardar el archivo txt del historial");
    }
}

/// <summary>
/// Metodo encargado de leer un archivo txt
/// </summary>
/// <param name="ruta">Ruta que señalan la ubicación del archivo</param>
/// <returns>El contenido del archivo en formato string, en caso contrario (que surga un error) lanza una excepción propia</returns>
2 referencias
public string Leer(string ruta)
{
    string contenidoDelArchivo = string.Empty;
    try
    {
        if (File.Exists(ruta) && new FileInfo(ruta).Length > 0)
        {
            using (StreamReader sw = new StreamReader(ruta))
            {
                contenidoDelArchivo = sw.ReadToEnd();
            }
        }
    }
    catch (Exception)
    {
        throw new FallaDeArchivoException("Excepcion propia controlada:\n falla al querer leer el archivo txt del historial");
    }
    return contenidoDelArchivo;
}
```

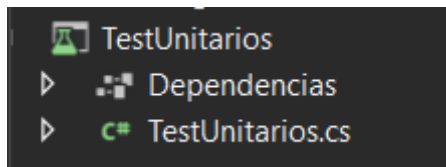


Esta excepción será capturada en el formClosing del form principal línea 220.

```
private void FormPrincipalMenu_FormClosing(object sender, FormClosingEventArgs e)
{
    try
    {
        serializadoraXml.Guardar(serializadoraXml.RutaArchivoXml, CentralAdministradora.ListaDeClientes);
        estadisticasDeLosServicios.Guardar(estadisticasDeLosServicios.RutaSerializarJson, estadisticasDeLosServicios);
        estadisticasDeLosServicios.calcularCantidadDeOperacionesQueSeRealizaron();
        estadisticasDeLosServicios.actualizarCantidadDeGenteActivaYnoActiva();
        MessageBox.Show("Datos guardados exitosamente", "Aviso", MessageBoxButtons.OK, MessageBoxIcon.Information);
    }
    catch (FallaDeArchivoException ex)
    {
        MessageBox.Show($"{ex.Message}", "Excepcion controlada", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    catch (Exception ex)
    {
        MessageBox.Show($"{ex.Message}", "Excepcion controlada", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

## Test Unitarios:

En el proyecto “TestUnitarios” hay varios test, la mayoría para testear los métodos relacionados a la búsqueda de clientes.



Prueba	Duration	Passes	Message
TestUnitarios (3)	3 ms		
TestUnitarios (3)	3 ms		
TestUnitarios (3)	3 ms		
BuscarClienteNoActivoEnLaList...	2 ms		
VerificarSiClienteEstaEnElSistem...	< 1 ms		
VerificarSiElClienteExisteYEstaA...	1 ms		

```
[TestMethod]
[ExpectedException(typeof(NoExisteClienteActivoConElDniIngresadoException))]
public void VerificarSiElClienteExisteYEstaActivoEnElSistema_CuandoElClienteNoExista_DeberiaLanzarNoExisteClienteActivoConElDniIngresadoException()
{
    // Arrange - instancio un cliente que no esta en mi sistema
    Cliente unCliente = new Cliente("Juan", "Perez", "Av. Mitre 1346", 895485765, new PlanPremium());

    // Act - Lo busco en el sistema, deberia lanzarme la excepcion xq dicho cliente no existe en mi sistema
    bool resultado = CentralAdministradora.VerificarSiElClienteExisteYEstaActivoEnElSistema(895485765);
}

[TestMethod]
public void VerificarSiClienteEstaEnElSistema_CuandoElClienteEsteEnElSistemaYActivo_DeberiaRetornarTrue()
{
    // Arrange - instancio un cliente que no esta en mi sistema y luego lo agrego
    Cliente unCliente = new Cliente("Juan", "Perez", "Av. Mitre 1346", 44965932, new PlanIntermedio());
    CentralAdministradora.ListaDeClientes.Add(unCliente);

    // Act - Verifico si el cliete esta en el sistema, deberia retornar true xq el si esta
    bool resultado = CentralAdministradora.VerificarSiClienteEstaEnElSistema(44965932);

    Assert.IsTrue(resultado);
}

[TestMethod]
public void BuscarClienteNoActivoEnLaLista_CuandoElClienteEsteActivo_DeberiaRetornarNull()
{
    // Arrange - instancio un cliente que no esta en mi sistema y luego lo agrego
    Cliente unCliente = new Cliente("Luis", "Miguel Hernandez", "Av. Belgrano 656", 43901903, new PlanIntermedio());
    unCliente.EstaActivo = true;
    CentralAdministradora.ListaDeClientes.Add(unCliente);

    // Act - Deberia retornar null xq el cliente esta activo
    Cliente resultado = CentralAdministradora.BuscarClienteNoActivoEnLaLista(43901903);

    Assert.IsNull(resultado);
}
```

## Interfaz y tipos genéricos

Interfaz genérica por un tema de practicidad ya que necesito guardar/serializar y leer/deserializar distintos objetos en distintas clases

```
3 referencias
public interface IArchivos<T>
{
    /// <summary>
    /// Guarda un objeto T en un archivo en una ruta específica, este metodo sera implemenado en las clases que impleten esta interfaz.
    /// </summary>
    /// <param name="ruta">Ruta que señalan la ubicación del archivo</param>
    /// <param name="contenido">T Objeto a guardar dentro del archivo.</param>
    9 referencias
    void Guardar(string ruta, T contenido);

    /// <summary>
    /// Lee el archivo contenido en la ruta y devuelve un tipo de objeto T, este metodo sera implemenado en las clases que impleten esta interfaz..
    /// </summary>
    /// <param name="ruta">Ruta que señalan la ubicación del archivo</param>
    /// <returns>Un objeto de tipo T</returns>
    8 referencias
    T Leer(string ruta);
}
```

Implementación de la interfaz en la clase Central Administradora

```
28 referencias
public class CentralAdministradora : IArchivos<string>
{
```

Con los respectivos métodos: en este caso quería leer y guardar archivos txt

Línea del método guardar: 163

Línea del método leer: 183

```
163 public void Guardar(string ruta, string contenido)
164 {
165     try
166     {
167         using (StreamWriter sw = new StreamWriter(ruta, true))
168         {
169             sw.WriteLine(contenido);
170         }
171     }
172     catch (Exception)
173     {
174         throw new FallaDeArchivoException("Excepcion propia controlada:\n falla querer guardar el archivo txt del historial");
175     }
176 }
177
178 /// <summary>
179 /// Metodo encargado de leer un archivo txt
180 /// </summary>
181 /// <param name="ruta">Ruta que señalan la ubicación del archivo</param>
182 /// <returns>El contenido del archivo en formato string, en caso contrario (que surga un error) lanza una excepción propia</returns>
183 public string Leer(string ruta)
184 {
185     string contenidoDelArchivo = string.Empty;
186     try
187     {
188         if (File.Exists(ruta) && new FileInfo(ruta).Length > 0)
189         {
190             using (StreamReader sw = new StreamReader(ruta))
191             {
192                 contenidoDelArchivo = sw.ReadToEnd();
193             }
194         }
195     }
196     catch (Exception)
197     {
198         throw new FallaDeArchivoException("Excepcion propia controlada:\n falla al querer leer el archivo txt del historial");
199     }
200     return contenidoDelArchivo;
201 }
```

## Implementación de la interfaz en la ClaseSerializadoraXml

```
3 referencias
public class ClaseSerializadoraXml : IArchivos<List<Cliente>>
{
```

Con los respectivos métodos: en este caso quería serializar y deserializar la lista de clientes en un archivo XML

Línea del método guardar: 33

Línea del método leer: 54

```
2 referencias
public virtual void Guardar(string ruta, List<Cliente> contenido)
{
    try
    {
        using (StreamWriter sw = new StreamWriter(ruta))
        {
            XmlSerializer xml = new XmlSerializer(typeof(List<Cliente>));
            xml.Serialize(sw, contenido);
        }
    }
    catch (Exception)
    {
        throw new FallaDeArchivoException("Excepcion. No se pudo serializar el archivo correctamente");
    }
}

/// <summary>
/// Metodo encargado de deserializar un archivo XML
/// </summary>
/// <param name="ruta">Ruta que señalan la ubicación del archivo XML</param>
/// <returns>La lista de cliente contenida en el archivo o null en caso que el archivo no exista o este vacío, en caso contrario (que surga un
3 referencias
public virtual List<Cliente> Leer(string ruta)
{
    try
    {
        //Agregue el File.Exists(ruta) si empieza a fallar es por eso
        if (File.Exists(ruta) && new FileInfo(ruta).Length > 0)
        {
            using (StreamReader sr = new StreamReader(ruta))
            {
                XmlSerializer xml = new XmlSerializer(typeof(List<Cliente>));
                List<Cliente> objeto = xml.Deserialize(sr) as List<Cliente>;
                return objeto;
            }
        }
        else
        {
            return null;
        }
    }
    catch (Exception)
    {
        throw new FallaDeArchivoException("Excepcion. No se pudo deserializar el archivo correctamente");
    }
}
```

## Implementación de la interfaz en la clase EstadisticaServicios

```
11 referencias
public class EstadisticaServicios : IArchivos<EstadisticaServicios>
{
```

Con los respectivos métodos: en este caso quería serializar y deserializar en json

Línea del método guardar: 93

Línea del método leer: 114

```

2 referencias
public void Guardar(string ruta, EstadisticaServicios contenido)
{
    try
    {
        JsonSerializerOptions opciones = new JsonSerializerOptions();
        opciones.WriteIndented = true;

        string objJson = JsonSerializer.Serialize(contenido, opciones);
        File.WriteAllText(ruta, objJson);
    }
    catch (Exception)
    {
        throw new FallaDeArchivoException("Excepcion: Fallo la serializacion del archivo Json");
    }
}

/// <summary>
/// Metodo encargado de deserializar Json
/// </summary>
/// <param name="ruta">Ruta que señalan la ubicación del archivo Json</param>
/// <returns>Objeto de tipo EstadisticaServicios deserializado o null en caso que el archivo no exista o es
3 referencias
public EstadisticaServicios Leer(string ruta)
{
    try
    {
        if (File.Exists(ruta) && new FileInfo(ruta).Length > 0)
        {
            JsonSerializerOptions opciones = new JsonSerializerOptions();
            opciones.WriteIndented = true;
            string ContenidoArchivoJson = File.ReadAllText(ruta);
            return JsonSerializer.Deserialize<EstadisticaServicios>(ContenidoArchivoJson);
        }
        else
        {
            return null;
        }
    }
    catch (Exception)
    {
        throw new FallaDeArchivoException("Excepcion: Fallo la deserializacion del archivo Json");
    }
}

```

## Archivos txt

El archivo txt lo utilice para escribir el historial de las operaciones realizadas. Cada vez que se realiza una operación como, por ejemplo, dar de alta a un cliente, modificar algún dato de los clientes o dar de baja se llamara al método guardar de la clase CentralAdministradora que se encarga de escribir el txt con la fecha, hora, y la operación realizada:

Clase y líneas en las cuales escribo el historial:

```

▲ Entidades\CentralAdministradora.cs (1)
  212 : Guardar(ruta, $"{DateTime.Now:f}hs - Se realizo el alta de un cliente:\nLlamado: {unCliente.Nombre}\nCon Dni: {unCliente.Dni}\nAdquirio el {unCliente.PlanEligido.GetType().Name}\n");
▲ Entidades\Archivos.cs (1)
  16 : void Guardar(string ruta, T contenido);
▲ Forms\FormIngresoDeDni.cs (1)
  196 : centralAdministradora.Guardar(centralAdministradora.Ruta, mensajeDelHistorial);
▲ Forms\FormPrincipalMenu.cs (2)
  77 : centralServicio.Guardar(centralServicio.Ruta, $"{DateTime.Now:f}hs - Se realizo una modificacion de datos al cliente:\nLlamado: {clienteParaModificar.Nombre}\nCon Dni: {clienteParaModificar.Dni}\n");
  106 : centralServicio.Guardar(centralServicio.Ruta, $"{DateTime.Now:f}hs - Se realizo la baja del cliente:\nLlamado: {clienteQueSeDaraDeBaja.Nombre}\nCon Dni: {clienteQueSeDaraDeBaja.Dni}\n");

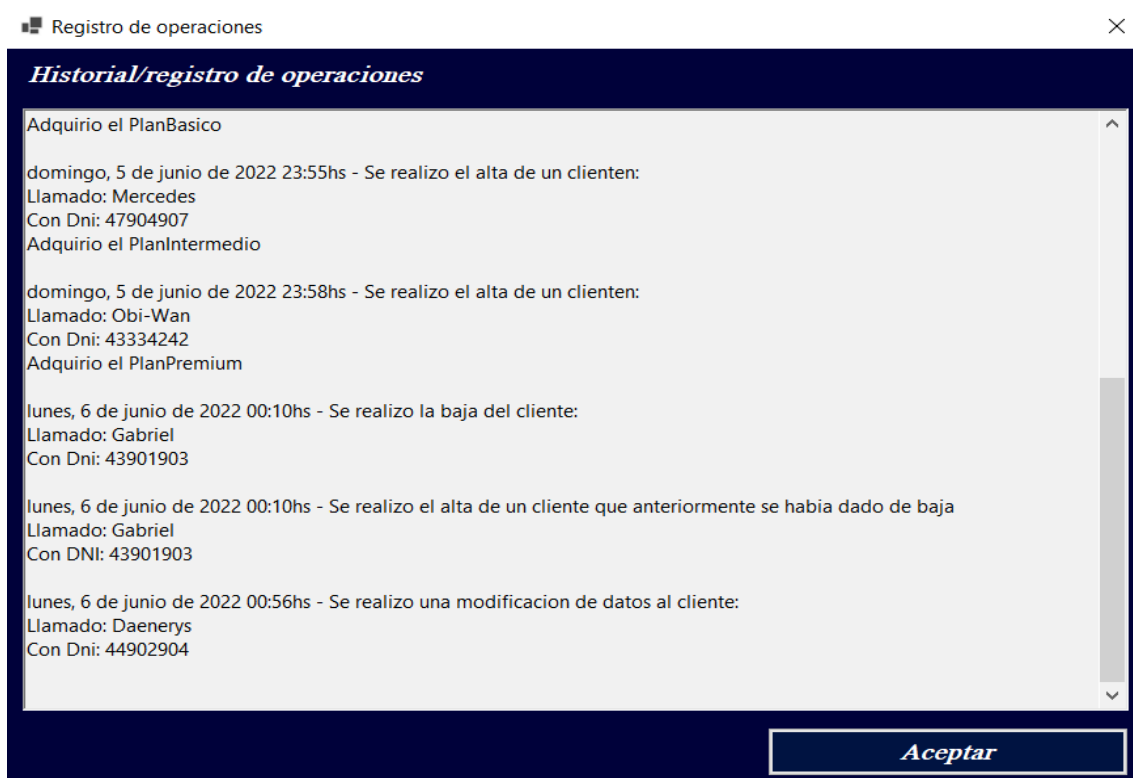
```

El archivo txt se levanta/lee cada en el evento click del btnHistorial del FormPrincipalMenu

Línea: 126

```
1 referencia
private void btnHistorial_Click(object sender, EventArgs e)
{
    try
    {
        string historialParaMostrar = centralServicio.Leer(centralServicio.Ruta);
        this.llamarFormMostrar("Registro de operaciones", "Historial/registro de operaciones", historialParaMostrar);
    }
    catch (FallaDeArchivoException ex)
    {
        MessageBox.Show(ex.Message, "Exception controlada", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

Para acceder a dicho historial hay un botón en el form principal llamado: "Historial de operaciones" que mostrara el contenido del archivo txt:



## Serializacion XML y Json

**Importante:** Los tantos archivos XML, Json y Txt serán encontrados en la siguiente ruta:

```
($"{AppDomain.CurrentDomain.BaseDirectory}"
s\TP_[3]\Alegre.Gabriel.2A.TP3\Forms\bin\Debug
```

**Xml:** Utilizo serialización xml para serializar la lista de clientes, ya que cada cliente tiene un plan asociado y dicho plan tiene una relación de herencia.

**Json:** Utilizo la serialización Json para poder hacer un informe estadístico del programa, sobre la atención al público, ya que si no lo hacía con archivo (el informe estadístico) al momento de cerrar el programa iba a perder la cuenta de la cantidad de altas que se realizaron, cantidad de bajas, cuantas veces modificaron a un cliente etc.

Dichos archivos Xml y Json son levantados/Deserializados al momento de abrir la aplicación, en el evento load del FormPrincipalMenu. Línea: 36

```
1 referencia
private void FormPrincipalMenu_Load(object sender, EventArgs e)
{
    Deserializar();
}
```

El método encargado de Deserializar se encuentra en el FormPrincipalMenu. Línea: 181

```
1 referencia
private void Deserializar()
{
    try
    {
        if (serializadoraXml.Leer(serializadoraXml.RutaArchivoXml) is not null)
        {
            CentralAdministradora.ListaDeClientes = serializadoraXml.Leer(serializadoraXml.RutaArchivoXml);
        }

        if (estadisticasDeLosServicios.Leer(estadisticasDeLosServicios.RutaSerializarJson) is not null)
        {
            estadisticasDeLosServicios = estadisticasDeLosServicios.Leer(estadisticasDeLosServicios.RutaSerializarJson);
        }
    }
    catch (FallaDeArchivoException ex)
    {
        MessageBox.Show($"{ex.Message}", "Excepcion controlada", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    catch (Exception ex)
    {
        MessageBox.Show($"{ex.Message}", "Excepcion controlada", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

Y estos archivos xml y json serán serializados al momento de cerrar la aplicación, en el evento formClosing del FormPrincipalMenu, Línea: 210

```
1 referencia
private void FormPrincipalMenu_FormClosing(object sender, FormClosingEventArgs e)
{
    try
    {
        serializadoraXml.Guardar(serializadoraXml.RutaArchivoXml, CentralAdministradora.ListaDeClientes);
        estadisticasDeLosServicios.Guardar(estadisticasDeLosServicios.RutaSerializarJson, estadisticasDeLosServicios);
        estadisticasDeLosServicios.calcularCantidadDeOperacionesQueSeRealizaron();
        estadisticasDeLosServicios.actualizarCantidadDeGenteActivaYnoActiva();
        MessageBox.Show("Datos guardados exitosamente", "Aviso", MessageBoxButtons.OK, MessageBoxIcon.Information);
    }
    catch (FallaDeArchivoException ex)
    {
        MessageBox.Show($"{ex.Message}", "Excepcion controlada", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    catch (Exception ex)
    {
        MessageBox.Show($"{ex.Message}", "Excepcion controlada", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```