

Contexto:

Se desarrolló una aplicación para un centro de estética, donde se darán de alta clientes, con un turno reservado, se permitirá manejar estos clientes y modificarlos según sea necesario. También se podrán agregar productos y registrar entradas y salidas de los mismos.

Temas incluidos:

Tema 10 – Excepciones:

 Se implementó una excepción llamada
 ArchivoNoEncontradoException, la cual se lanzará si el archivo que se intenta deserializar existe o no coincide la ruta.

```
/// <summary>
/// Descrializa un archivo XML, retornando un objeto que contiene la información retribuida
// <summary>
// // cparam name: "nombreArchivo"> Nombre del archivo a leer 
// creturns>
// creturns
// cretur
```

Tema 11 – Pruebas unitarias:

• Se implementaron 6 tests unitarios, probando 2 funcionalidades.

```
[TestMethod]
Orderences
public void Test_AgregarElemento_CuandoNoHayaElementosEnLaLista_DeberiaRetornarTrue()

{
    // Arrange
    Controlador<Cliente> controlador = new Controlador<Cliente>();
    Cliente cliente = new Cliente(DateTime.Today, "Gonzalo", "Munioz", "44361856", "Mendoza 892");
    bool expected = true;

    // Act:
    bool actual = controlador + cliente;

    // Assert
    Assert.AreEqual(expected, actual);
}
```



Tema 12 – Tipos genéricos:

 Se implementó generics utilizando una clase controladora, que será la que se hará cargo de administrar nuestro listado (puede ser de cualquier objeto), y demás funciones. Además, se implementó generics para desarrollar una clase serializadora capaz de serializar cualquier objeto.

```
46 references
public class Controlador<T> where T : class // TEMA 12 - TIPOS GENÉRICOS
{
    List<T> listaDeElementos;

    11 references
    public List<T> ListaDeElementos
    {
        get { return listaDeElementos = value; }
}

8 references
public int CantidadDeElementos
{
        get { return listaDeElementos
}

8 references
public int CantidadDeElementos.Count; }
}

5 references
public Controlador()
{
        this.listaDeElementos = new List<T>();
}
```

Tema 13 – Interfaces:

 Se implementó interfaces, utilizando una clase genérica serializadora, en la que se define la firma para el método Serializar y Deserializar, y en las clases SerializadoraXML y SerializadoraJSON se le da la implementación.

```
2 references
public interface ISerializadora<T> // TEMA 13 - INTERFACES
{
    /// <summary>
    /// Serializa y guarda en un archivo cualquier objeto
    /// </summary>
    /// <param name="elemento"> Elemento a serializar </param>
    /// <param name="nombreArchivo"> Nombre del archivo a escribir </param>
    6 references
    public void Serializar(T elemento, string nombreArchivo);

    /// <summary>
    /// Deserializa un archivo y almacena la información retribuida en un objeto del mismo tipo
    /// </summary>
    /// <param name="nombreArchivo"> Nombre del archivo a leer </param>
    /// <returns>
    6 references
    public T Deserializar(string nombreArchivo);
}
```



Tema 14 – Archivos:

 Se implementó archivos para manipular los archivos de datos XML y JSON

Tema 15 – Serialización:

• Se implementó serialización para serializar y deserializar archivos XML y JSON de nuestro objeto, para así poder cargarlos o sobreescribirlos a conveniencia.

