```
using System;
using System.Data;
using System.IO;
using System.Net.NetworkInformation;
using System.Security.Cryptography;
using System.Text;



namespace sigal
{
    class modulo1
    {
        /* zona de declaracion de variables
         *
         *
         */

        public static bool IsNumeric(object Expression)

        {

            bool isNum;

            double retNum;

            isNum = Double.TryParse(Convert.ToString(Expression),
System.Globalization.NumberStyles.Any,
System.Globalization.NumberFormatInfo.InvariantInfo, out retNum);

            return isNum;

        }
        public static void Exportar_Excel(DataTable dt, string archivo)
        {

            //open file
            StreamWriter wr = new StreamWriter(@"c:\\SIGAL\EXPORTACIONES\" +
archivo + ".xlsx");

            try
            {

                for (int i = 0; i < dt.Columns.Count; i++)
                {
                    wr.Write(dt.Columns[i].ToString().ToUpper() + "\t");
                }

                wr.WriteLine();

                //write rows to excel file
                for (int i = 0; i < (dt.Rows.Count); i++)
                {
                    for (int j = 0; j < dt.Columns.Count; j++)
                    {
                        if (dt.Rows[i][j] != null)
                        {
                            wr.Write(Convert.ToString(dt.Rows[i][j]) + "\t");
```

```
                        }
                        else
                        {
                            wr.Write("\t");
                        }
                    }
                    //go to next line
                    wr.WriteLine();
                }
                //close file
                wr.Close();
            }
            catch (Exception ex)
            {
                throw ex;
            }
        }

        public static string Desencriptar(string textoEncriptado)
        {
            try
            {
                string key = "qualityinfosolutions";
                byte[] keyArray;
                byte[] Array_a_Descifrar =
Convert.FromBase64String(textoEncriptado);

                //algoritmo MD5
                MD5CryptoServiceProvider hashmd5 = new
MD5CryptoServiceProvider();

                //keyArray =
hashmd5.ComputeHash(UTF8Encoding.UTF8.GetBytes(key));
                keyArray =
hashmd5.ComputeHash(UTF8Encoding.BigEndianUnicode.GetBytes(key));

                hashmd5.Clear();

                TripleDESCryptoServiceProvider tdes = new
TripleDESCryptoServiceProvider();

                tdes.Key = keyArray;
                tdes.Mode = CipherMode.ECB;
                tdes.Padding = PaddingMode.PKCS7;

                ICryptoTransform cTransform = tdes.CreateDecryptor();
                // byte[] resultArray =
cTransform.TransformFinalBlock(Array_a_Descifrar, 0, Array_a_Descifrar.Length);
                byte[] resultArray =
cTransform.TransformFinalBlock(Array_a_Descifrar, 0, Array_a_Descifrar.Length);

                tdes.Clear();
                // textoEncriptado = UTF8Encoding.UTF8.GetString(resultArray);
                textoEncriptado =
UTF8Encoding.BigEndianUnicode.GetString(resultArray);

            }
            catch (Exception)
```

```
            {

            }
            return textoEncriptado;
        }

        public static string Encriptar(string texto)
        {
            try
            {
                string key = "qualityinfosolutions"; //llave para encriptar
datos

                byte[] keyArray;
                // byte[] Arreglo_a_Cifrar = UTF8Encoding.UTF8.GetBytes(texto);
                byte[] Arreglo_a_Cifrar =
UTF8Encoding.BigEndianUnicode.GetBytes(texto);

                //Se utilizan las clases de encriptación MD5

                MD5CryptoServiceProvider hashmd5 = new
MD5CryptoServiceProvider();
                // keyArray =
hashmd5.ComputeHash(UTF8Encoding.UTF8.GetBytes(key));
                keyArray =
hashmd5.ComputeHash(UTF8Encoding.BigEndianUnicode.GetBytes(key));

                hashmd5.Clear();

                //Algoritmo TripleDES
                TripleDESCryptoServiceProvider tdes = new
TripleDESCryptoServiceProvider();

                tdes.Key = keyArray;
                tdes.Mode = CipherMode.ECB;
                tdes.Padding = PaddingMode.PKCS7;

                ICryptoTransform cTransform = tdes.CreateEncryptor();

                // byte[] ArrayResultado =
cTransform.TransformFinalBlock(Arreglo_a_Cifrar, 0, Arreglo_a_Cifrar.Length);
                byte[] ArrayResultado =
cTransform.TransformFinalBlock(Arreglo_a_Cifrar, 0, Arreglo_a_Cifrar.Length);

                tdes.Clear();

                //se regresa el resultado en forma de una cadena
                texto = Convert.ToBase64String(ArrayResultado, 0,
ArrayResultado.Length);

            }
            catch (Exception)
            {

            }
            return texto;
        }
```

```
        public static string pingdered(string ipstring)
        {
            // https://www.todavianose.com/hacer-ping-con-visual-c/
            Ping HacerPing = new Ping();
            int iTiempoEspera = 5000;
            PingReply RespuestaPing;
            string sDireccion, txtLog;
            string txtIP = ipstring;
            sDireccion = txtIP;
            RespuestaPing = HacerPing.Send(sDireccion, iTiempoEspera);
            if (RespuestaPing.Status == IPStatus.Success)
            {
                /* txtLog =("Ping a " +
                     sDireccion.ToString() +
                     "[" +
                     RespuestaPing.Address.ToString() +
                     "]" +
                     " Correcto" +
                     " Tiempo de respuesta = " +
                     RespuestaPing.RoundtripTime.ToString() +
                     " ms" +
                     "\n");*/
                txtLog = "si";
            }
            else
            {
                /*txtLog = ("Error: Ping a " +
                    sDireccion.ToString() +
                    "\n");*/
                txtLog = "no";


            }
            return txtLog;
        } //




    }
}
```