```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.IO;
using System.Xml;
using System.Net;


namespace WinAppli_client_WS
{
    /// <summary>
    /// Application cliente utilisant les WebServices
    /// </summary>
    public partial class Form1 : Form
    {
        /// <summary>
        /// WebService contenant les 4 opérations HelloWorldPerso, TTC, inverser_chaine, attendre_N_secondes
        /// </summary>
        private localhost.Service1 monWS = new localhost.Service1();

        /// <summary>
        /// WebService contenant l'operation de collecte de donnée sur une ville (météo...)
        /// </summary>
        private com.webservice.globalweather.GlobalWeather meteoWS = new WinAppli_client_WS.com.webservice.globalweather.GlobalWeather();

        /// <summary>
        /// WebService contenant l'operation de generation de codeBarre
        /// </summary>
        private com.barcodesoft.bcdgen.BarCodeWebService bareCodeWS = new WinAppli_client_WS.com.barcodesoft.bcdgen.BarCodeWebService();

        public Form1()
        {
            InitializeComponent();
            initializeAsyncMessage();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            ServicePointManager.Expect100Continue = false;
            textBoxCodeBarreValue.Text = "4006381333689";
            textBoxHeightValue.Text = "20";
            textBoxWidthValue.Text = "40";
            textBoxResolutionValue.Text = "400";
        }

        /// <summary>
        /// Definit les associations des appels de WS asynchrones avec leur callback
        /// </summary>
        public void initializeAsyncMessage() {
            //CallBack pour le WS attendre N Secondes
            monWS.attendre_N_secondesCompleted += new localhost.attendre_N_secondesCompletedEventHandler(callBackWaitNSecondes);
        }

        /// <summary>
        /// CallBack de retour à l'appel asynchrone
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="args"></param>
        public void callBackWaitNSecondes(object sender, localhost.attendre_N_secondesCompletedEventArgs args) {
            MessageBox.Show("Retour OK :" + args.Result);
        }
```

```
        /// <summary>
        /// Methode appelant le WS d'affichage du Hello World personnalisé classique.
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void buttonHelloPerso_Click(object sender, EventArgs e)
        {
            MessageBox.Show(monWS.HelloWorldPerso(textBoxSayHello.Text));
        }

        /// <summary>
        /// Méthode appelant le WS de calcul du prix TTC lors du clic du bouton Calculer TTC
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void buttonCalculateTTC_Click(object sender, EventArgs e)
        {
            double HT = 0.0;
            double taux = 20.0;
            bool HT_OK = Double.TryParse(textBoxPriceHTValue.Text,out HT);
            bool TVA_OK = Double.TryParse(textBoxTVAValue.Text, out taux);
            if (HT_OK && TVA_OK)
            {
                labelPriceTCCValue.Text = monWS.TTC(HT, taux).ToString();
            }
            else {
                labelPriceTCCValue.Text = "";
                MessageBox.Show("le prix HT ou la TVA n'est pas correctement renseignée.");
            }

        }

        /// <summary>
        /// Méthode appelant le WS d'inversion de chaine de caracteres
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void buttonInverserMot_Click(object sender, EventArgs e)
        {
            labelInverserMotValue.Text = monWS.inverser_chaine(textBoxInversionValue.Text);
        }

        /// <summary>
        /// Appel de l'operation WS d'attente de N sec en synchrone
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void buttonWaitSync_Click(object sender, EventArgs e)
        {
            if (textBoxWaitSecondValue.Text != null && textBoxWaitSecondValue.Text != "")
            {
                monWS.attendre_N_secondes(Convert.ToInt32(textBoxWaitSecondValue.Text));
                MessageBox.Show("Attente synchrone de " + textBoxWaitSecondValue.Text + " secondes demandée. Processus bloquant.");
            }
        }

        /// <summary>
        /// Appel de l'operation WS d'attente de N sec en asynchrone
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void buttonWaitAsync_Click(object sender, EventArgs e)
        {
            if (textBoxWaitSecondValue.Text != null && textBoxWaitSecondValue.Text != "")
            {
                monWS.attendre_N_secondesAsync(Convert.ToInt32(textBoxWaitSecondValue.Text));
                MessageBox.Show("Attente asynchrone de " + textBoxWaitSecondValue.Text + " secondes demandée. Processus non bloquant.");
```

```
            }
        }

        /// <summary>
        /// Alimente la liste des villes pour le pays recherché
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void buttonSearchAllCity_Click(object sender, EventArgs e)
        {

            String country = textBoxStateMeteoValue.Text;
            if (country != null && country != "") {
                String fluxXML = meteoWS.GetCitiesByCountry(country);
                XmlDocument docXML = new XmlDocument();
                docXML.LoadXml(fluxXML);
                XmlNodeList elementsXML = docXML.SelectNodes("//City");

                //StreamWriter writer = new StreamWriter("meteo.xml");
                //writer.WriteLine(fluxXML);
                //writer.Close();
                foreach (XmlNode element in elementsXML)
                {
                    comboBoxCitiesMeteo.Items.Add(element.InnerText);
                }
                comboBoxCitiesMeteo.Sorted = true;
                comboBoxCitiesMeteo.SelectedItem = 1;
                comboBoxCitiesMeteo.Focus();
            }
        }

        /// <summary>
        /// Récupère la météo pour la ville selectionnee
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void buttonGetMeteo_Click(object sender, EventArgs e)
        {
            String city = comboBoxCitiesMeteo.Text;
            if (city != null && city != "") {
                String xmlWeatherResult = meteoWS.GetWeather(city, textBoxStateMeteoValue.↙
    Text);

                XmlDocument xmlDocumentWeatherResult = new XmlDocument();
                xmlDocumentWeatherResult.LoadXml(xmlWeatherResult);

                XmlNodeList nodesWeather = xmlDocumentWeatherResult.SelectNodes("/          ↙
    CurrentWeather/*");

                foreach (XmlNode node in nodesWeather)
                {
                    listBoxMeteoDescription.Items.Add(node.Name + " : " + node.InnerText);
                }
            }
        }

        /// <summary>
        /// Récupère le code barre EAN13 et l'affiche
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void buttonGenerateBarCode_Click(object sender, EventArgs e)
        {
            if (   textBoxCodeBarreValue.Text == ""
                || textBoxResolutionValue.Text == ""
                || textBoxWidthValue.Text == ""
                || textBoxHeightValue.Text == "") {
                    MessageBox.Show("Veuillez renseigner correctement les champs pour      ↙
    obtenir le code barre.");
                return;
            }
            Byte[] codeBarreImgStream = bareCodeWS.EAN13(
                textBoxCodeBarreValue.Text,
                "strabbon",
```

```
                com.barcodesoft.bcdgen.BcsImageFormat.BMP,
                WinAppli_client_WS.com.barcodesoft.bcdgen.BcsOrientation.Original,
                int.Parse(textBoxResolutionValue.Text),
                int.Parse(textBoxWidthValue.Text),
                int.Parse(textBoxHeightValue.Text),
                "strtoken"
                );
            MemoryStream inStream = new MemoryStream(codeBarreImgStream);
            Image codeBarreImg = Image.FromStream(inStream);
            pictureBoxBareCode.SizeMode = PictureBoxSizeMode.AutoSize;
            pictureBoxBareCode.Image = codeBarreImg;
            pictureBoxBareCode.Refresh();
        }

        private void buttonExitApp_Click(object sender, EventArgs e)
        {
            Application.Exit();
        }
    }
}
```