# MANUAL DE USUARIO

A continuación se presenta el manual de usuario para la implementación de un ejemplo básico de la utilización de DDS - RTPS.

## REQUERIMIENTOS

Para poder realizar un ejemplo se debe anteriormente seguir el Manual de Usuario del Protocolo.

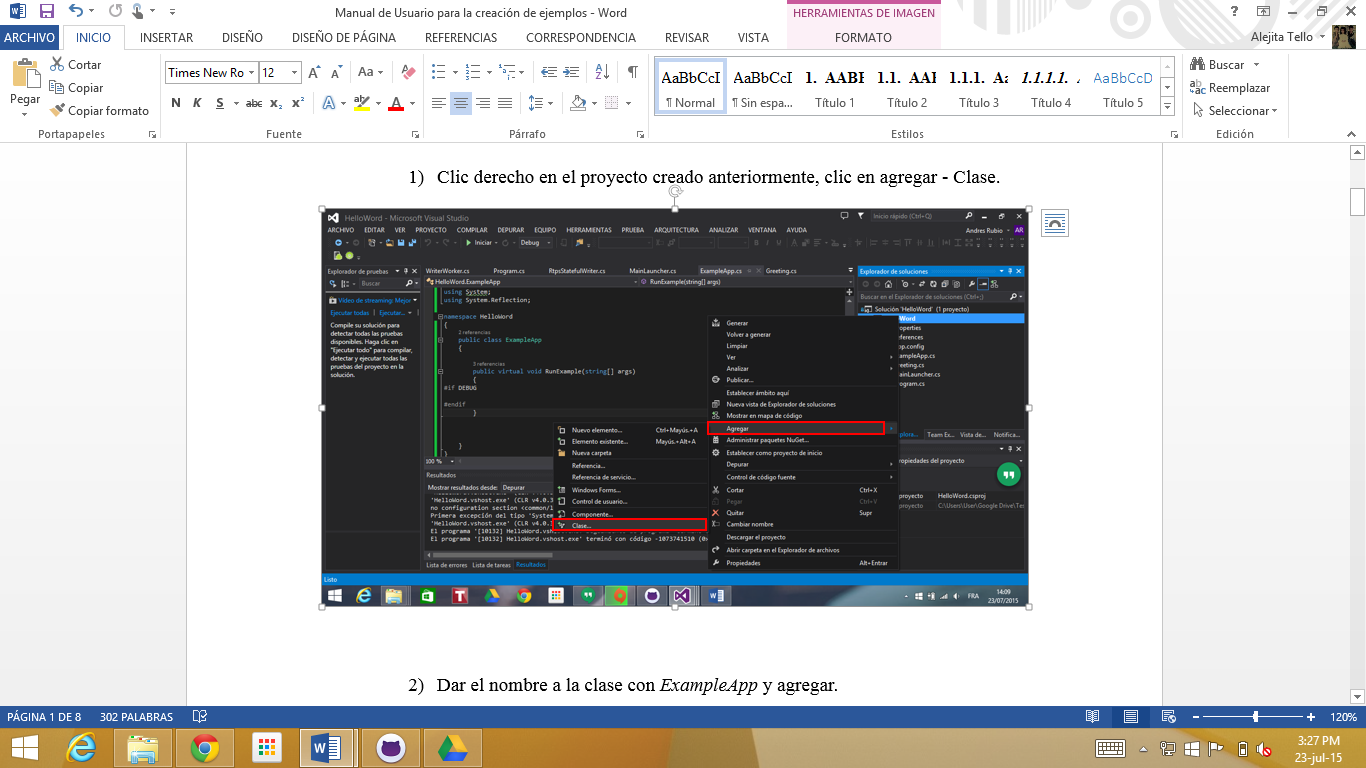
## PROCESO DE CREACIÓN DE UN ejemplo “hola mundo”

Dentro de este programa se genera el envío y recepción de la secuencia Hola Mundo.

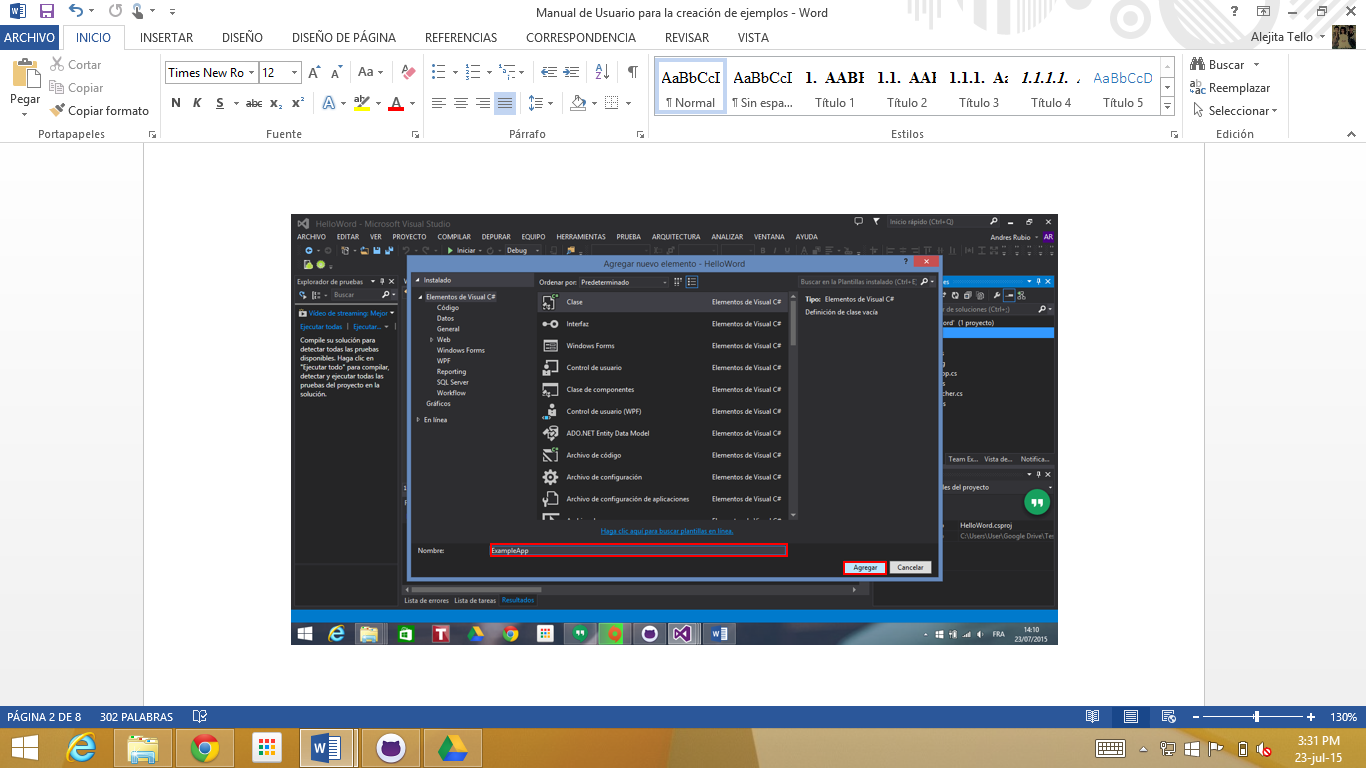
### Creación de clases adicionales

#### Creación de la clase ExampleApp.cs

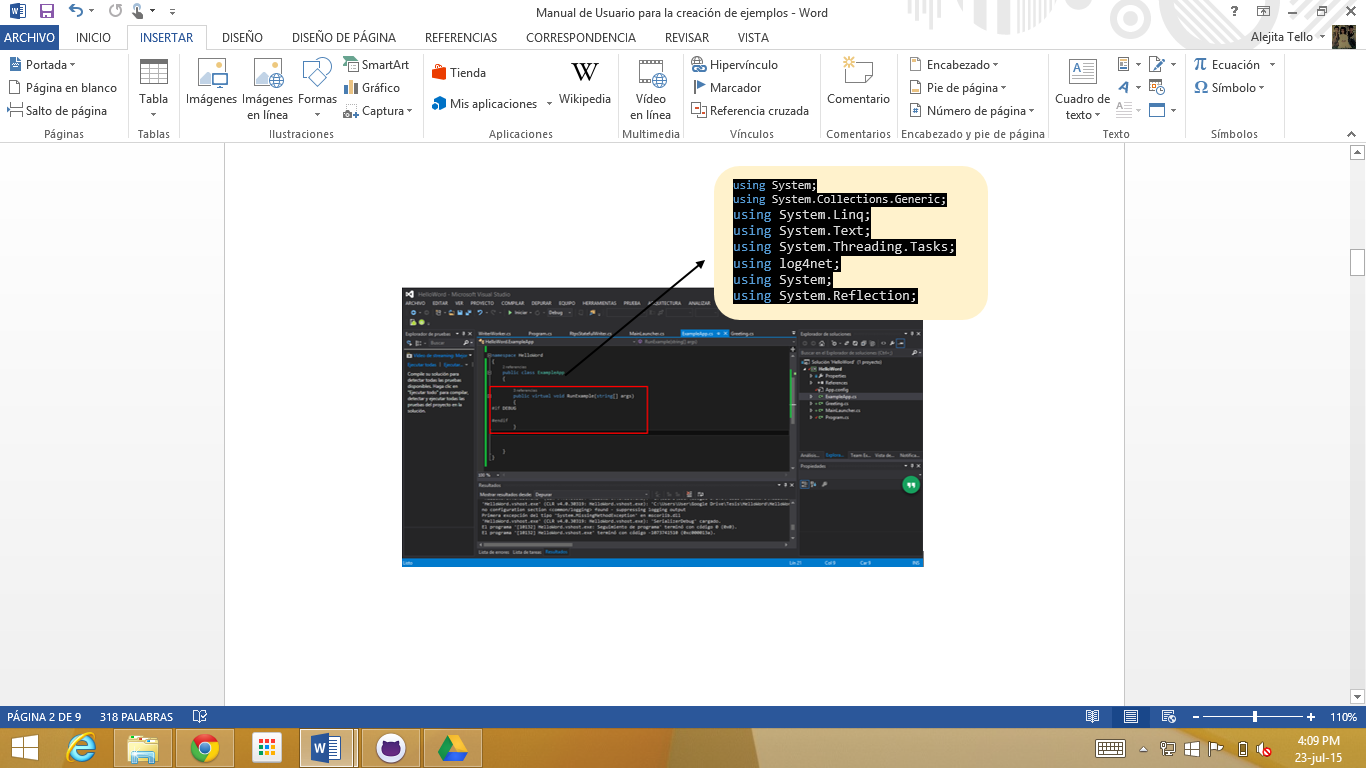
1. Clic derecho en el proyecto creado anteriormente, clic en agregar - Clase.



1. Dar el nombre a la clase con *ExampleApp* y agregar.

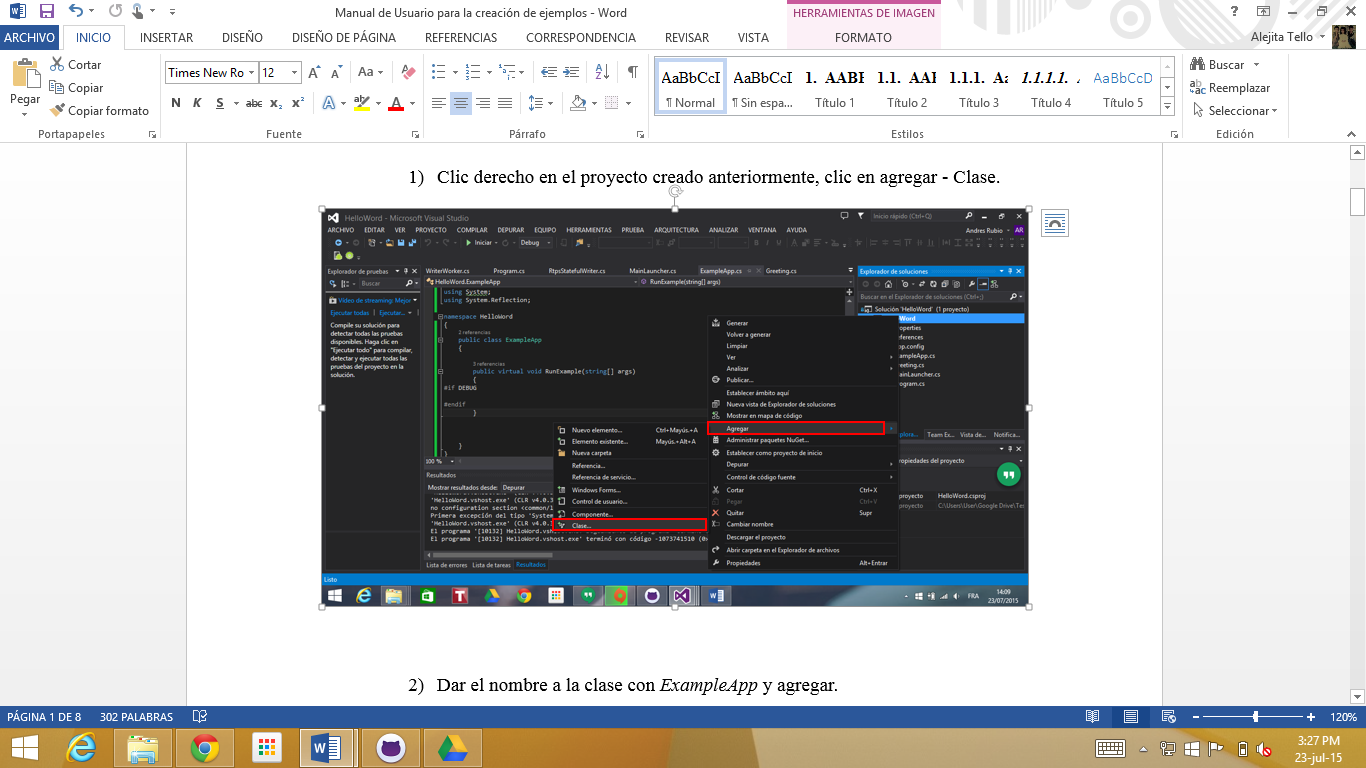


1. Dentro de la clase *ExampleApp* crear el método *Public Virtual RunExample*, como se muestra en la figura. Verificar que las siguientes referencias se encuentren añadidas.

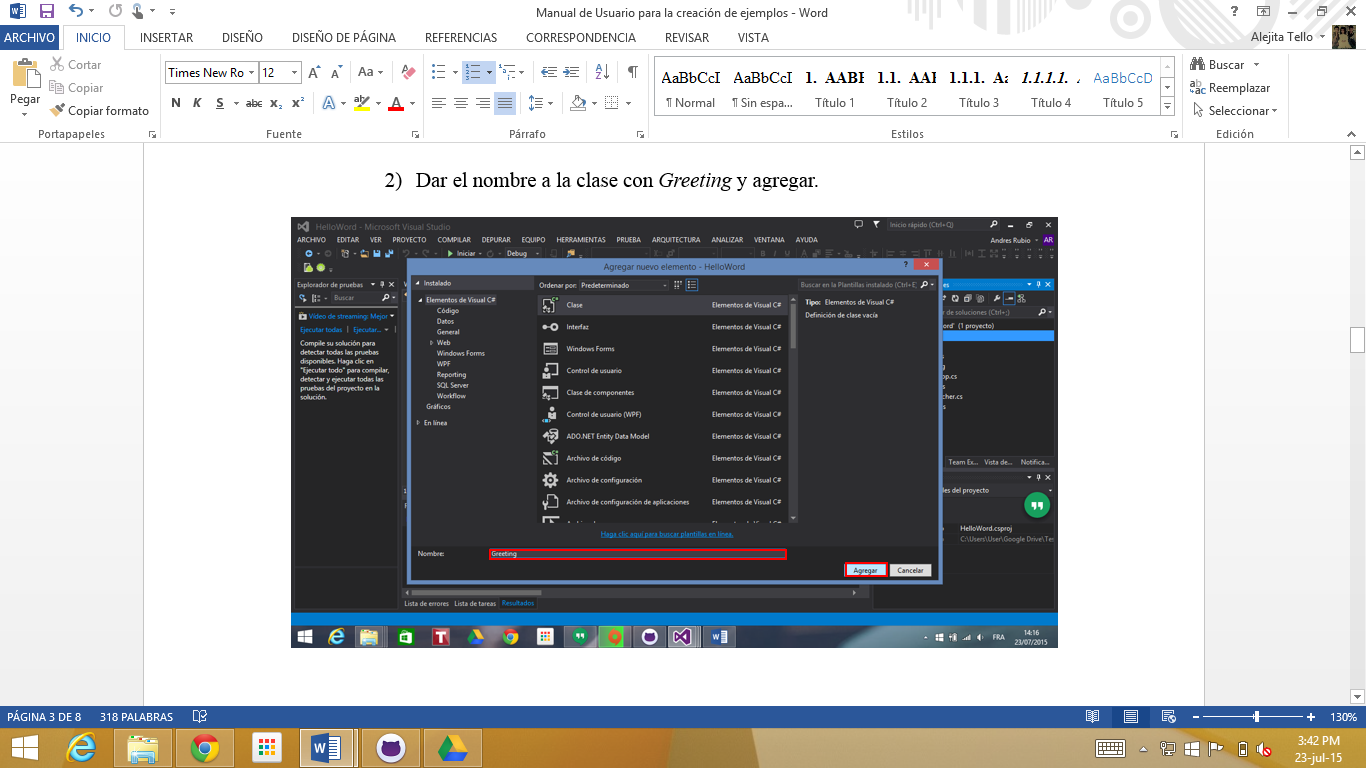


#### Creación de la clase Greeting.cs

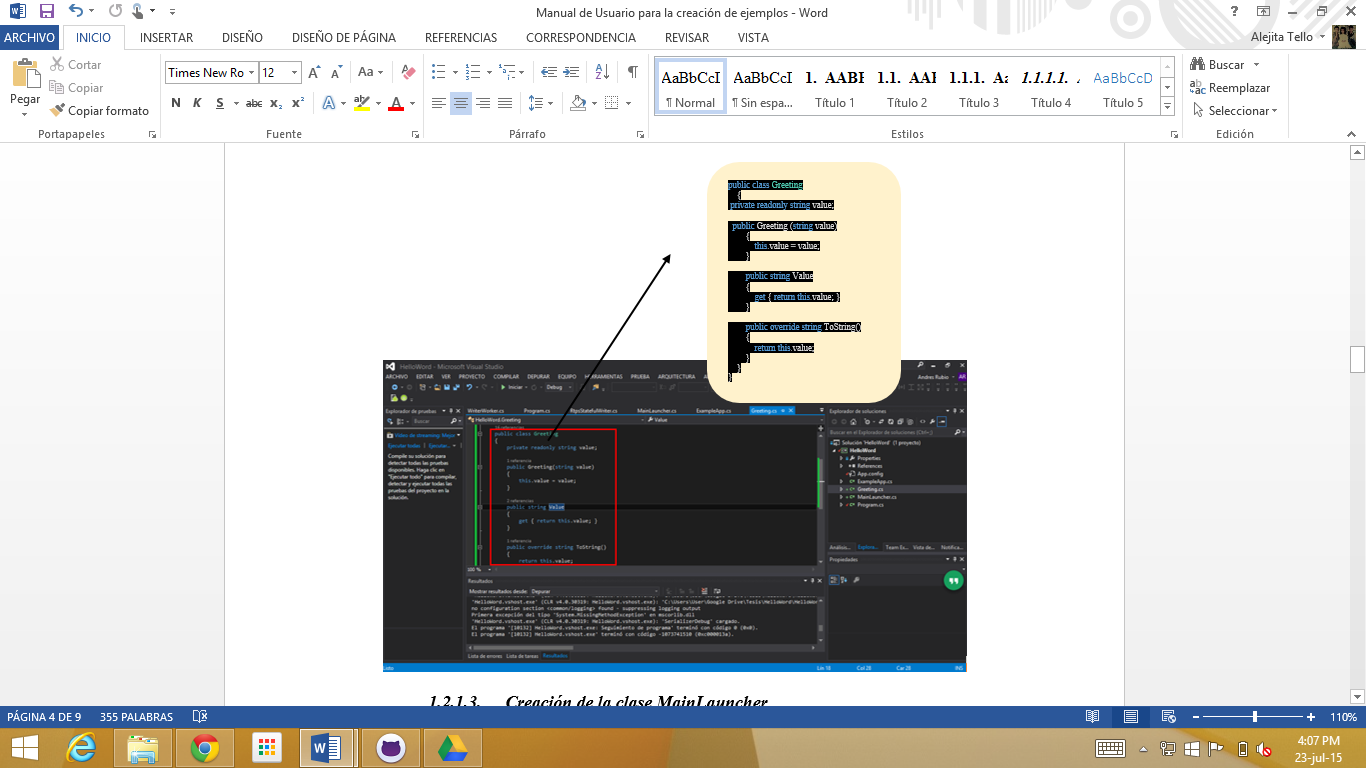
1. Clic derecho en el proyecto creado anteriormente, clic en agregar - Clase.



1. Dar el nombre a la clase con *Greeting* y agregar.

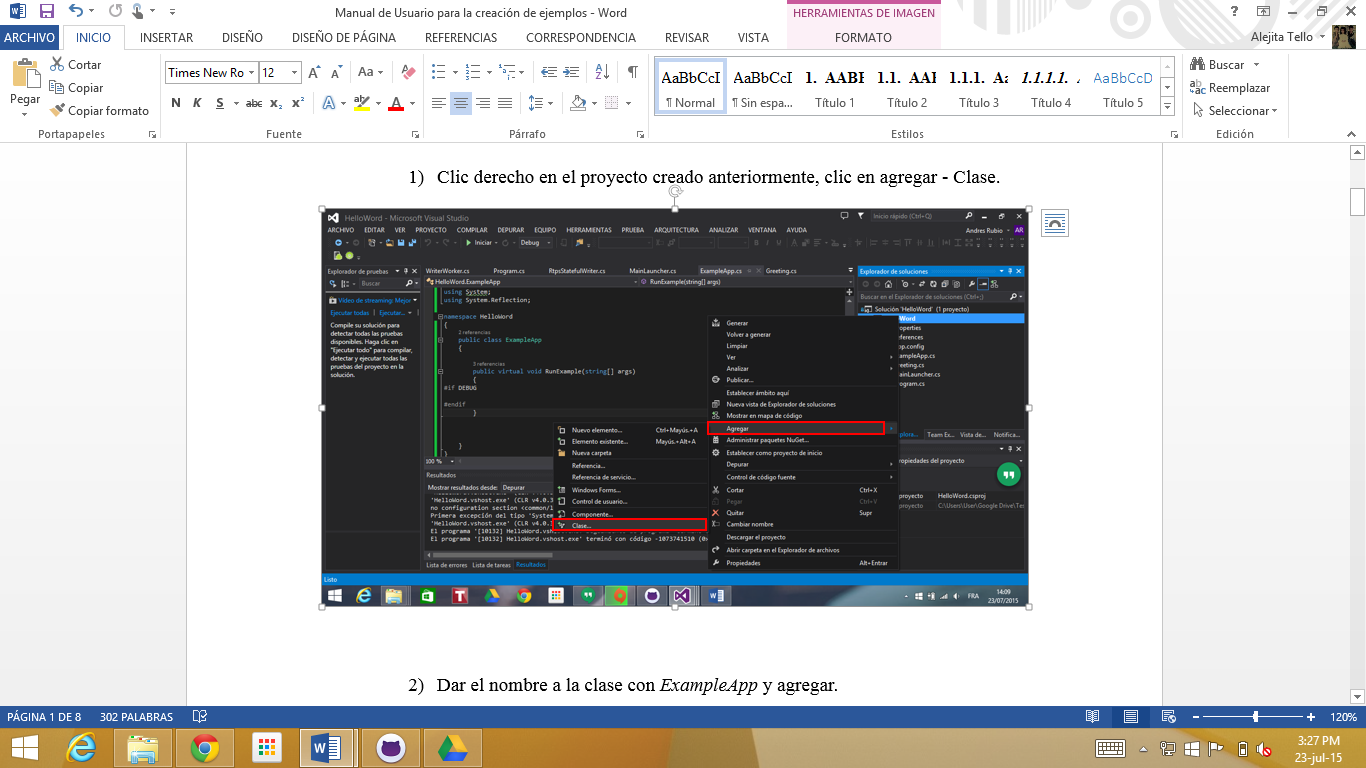


1. Dentro de la clase *Greeting* agregar las instancias básicas de los atributos de las clases.

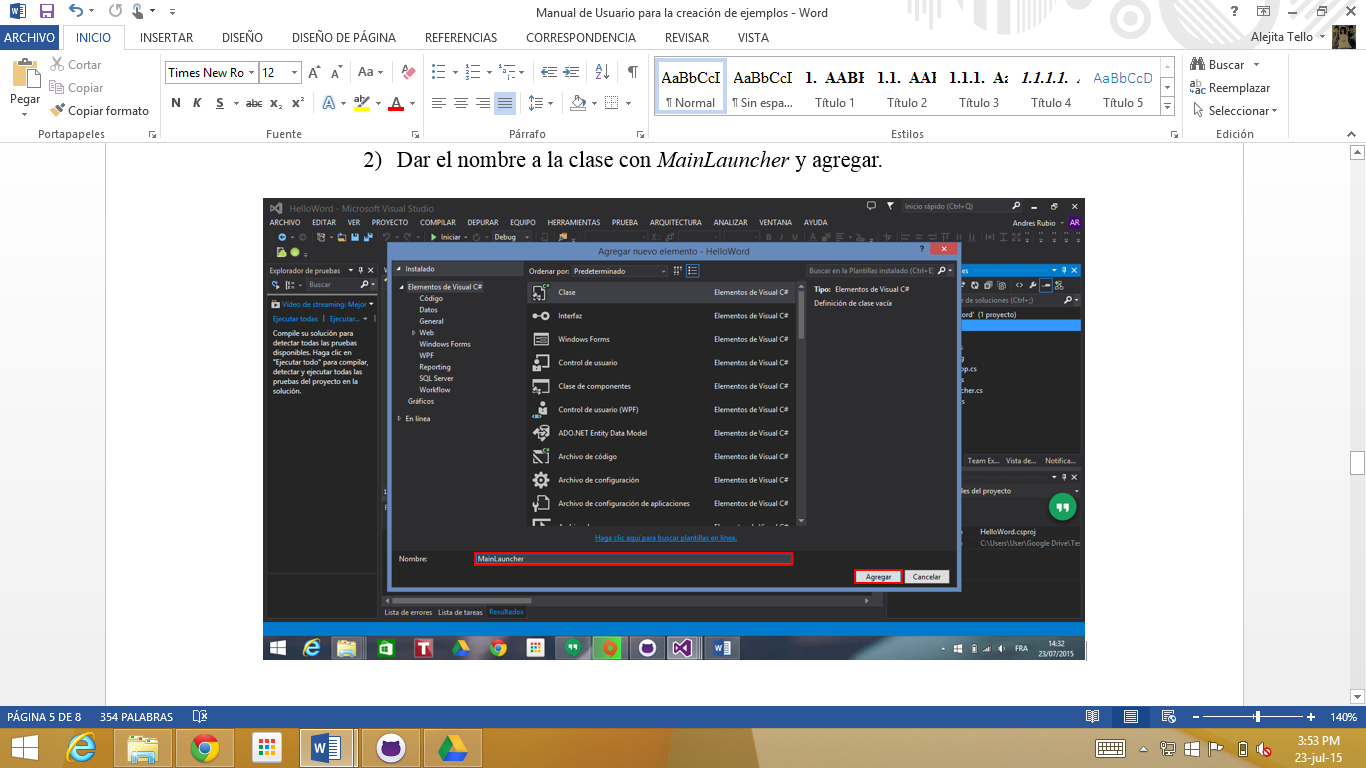


#### Creación de la clase MainLauncher

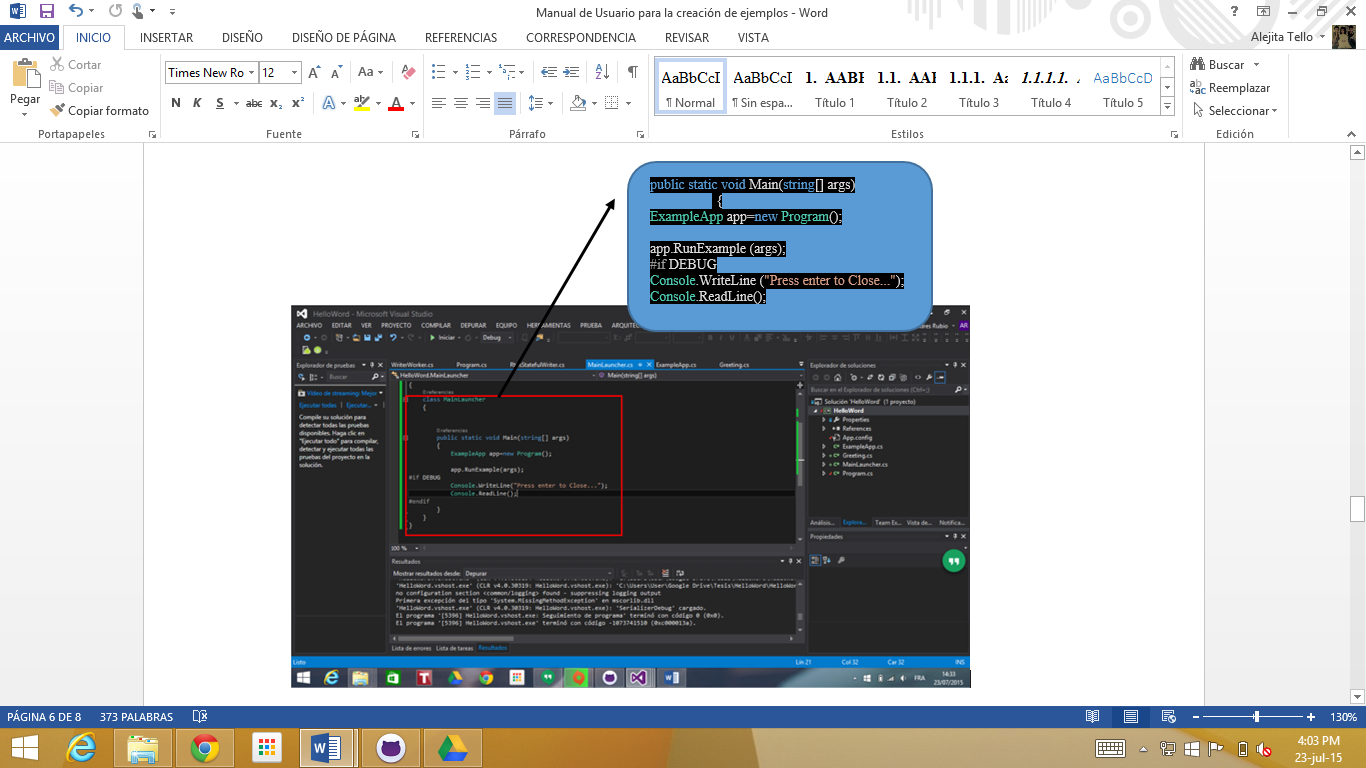
1. Clic derecho en el proyecto creado anteriormente, clic en agregar - Clase.



1. Dar el nombre a la clase con *MainLauncher* y agregar.



1. Dentro de la clase *MainLauncher* agregar el siguiente código para poder correr la clase *Program.cs* (Esta clase se crea automáticamente al generar el proyecto).



### Codificación de la lógica del programa en la clase *Program.cs*

1. Dentro de la clase *Program.cs,* adicionalmente a las referencias agregadas se debe agregar *using HelloWorld*.
2. A la declaración de la clase *Program.cs* se le extiende la clase *ExampleApp.cs,* por lo cual es necesario tener el método *RunExample*. Dentro del método *RunExample* primeramente inicializamos al participante, al topic y al *DomainParticipantFactory.*

public class Program : ExampleApp

{

public override void RunExample(string[] args)

{

base.RunExample(args);

DomainParticipantFactory factory = DomainParticipantFactory.GetInstance(Bootstrap.CreateInstance());

DomainParticipant dp = factory.CreateParticipant();

// Implicitly create TypeSupport and register type:

Topic<Greeting> tp = dp.CreateTopic<Greeting>("Greetings Topic");

1. A continuación se crea el Publicador y el *DataWriter.*

// Create the publisher

Publisher pub = dp.CreatePublisher();

/\* DataWriter<Greeting> dw = pub.CreateDataWriter(tp);

\*/

DataWriter<Greeting> dw = pub.CreateDataWriter<Greeting>(tp,

pub.GetDefaultDataWriterQos(),null, null);

1. Se crea el Suscriptor, el *DataReader* y el *DataReaderListener,* el cual servirá para la escucha de mensajes.

// Create the subscriber

Subscriber sub = dp.CreateSubscriber();

DataReaderListener<Greeting> ls = new MyListener();

/\*DataReader<Greeting> dr = sub.CreateDataReader(tp);\*/

DataReader<Greeting> dr = sub.CreateDataReader<Greeting>(tp,

sub.GetDefaultDataReaderQos(),

ls,

null );

/\*

// Now Publish some piece of data

Greeting data = new Greeting("Hello, World with DDS.");

Console.WriteLine("Sending data:\"{0}\"", data.Value);

dw.Write(data);

//and check that the reader has this data

dr.WaitForHistoricalData(10, TimeUnit.SECONDS);

1. Se crea el código de publicación de datos utilizando método del *DataWriter write (),* y se espera en el *DataReader* un tiempo de 1500 ms hasta la recepción del mensaje.

\*/

int i = 0;

// Now Publish some piece of data

//Greeting data = new Greeting("Hola Mundo"+ i.ToString());

for (i = 0; i < 1; i++)

{

Greeting data = new Greeting("Hola Mundo" + i.ToString());

Console.WriteLine("Sending data:\"{0}\"", data.Value);

dw.Write(data);

dr.WaitForHistoricalData(1500, TimeUnit.MILLISECONDS);

}

//and check that the reader has this data

//dr.WaitForHistoricalData(10000, TimeUnit.SECONDS);

dp.Close();

}

1. Dentro del *Listener* se genera un evento llamado *OnDataAvailable* que sirve para la escritura de mensajes.

private class MyListener : DataReaderAdapter<Greeting>

{

//private static readonly ILog log = LogManager.GetLogger(MethodBase.GetCurrentMethod().DeclaringType);

public override void OnDataAvailable(DataAvailableStatus<Greeting> status)

{

DataReader<Greeting> dr = status.GetSource();

SampleIterator<Greeting> it = dr.Take();

foreach (Sample<Greeting> smp in it)

{

// SampleInfo stuff is built into Sample:

// InstanceHandle inst = smp.GetInstanceHandle();

// Data accessible from Sample; null if invalid:

Greeting dt = smp.GetData();

Console.WriteLine("Received data:\"{0}\"", dt.Value);

}

}

}

}