# 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.

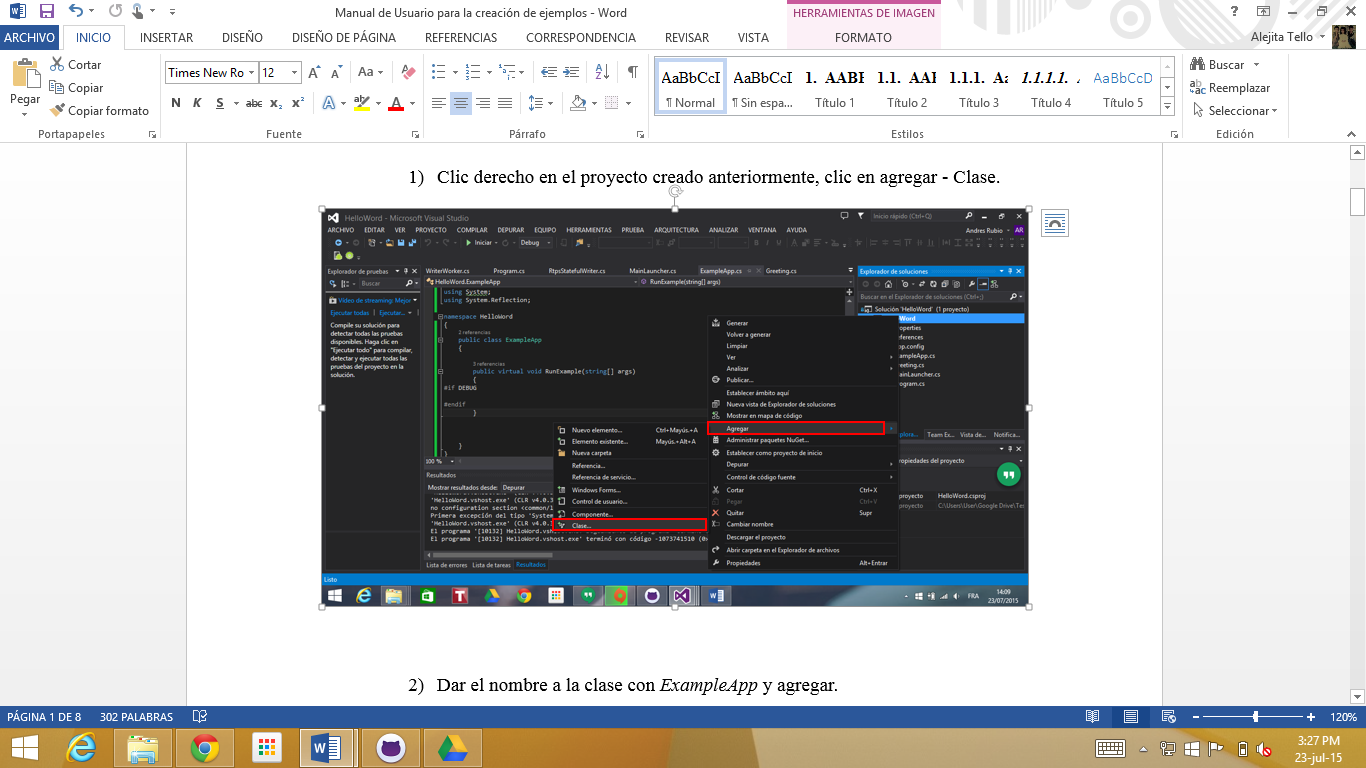
## PROCESO DE CREACIÓN DE UN ejemplo “hola mundo”

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

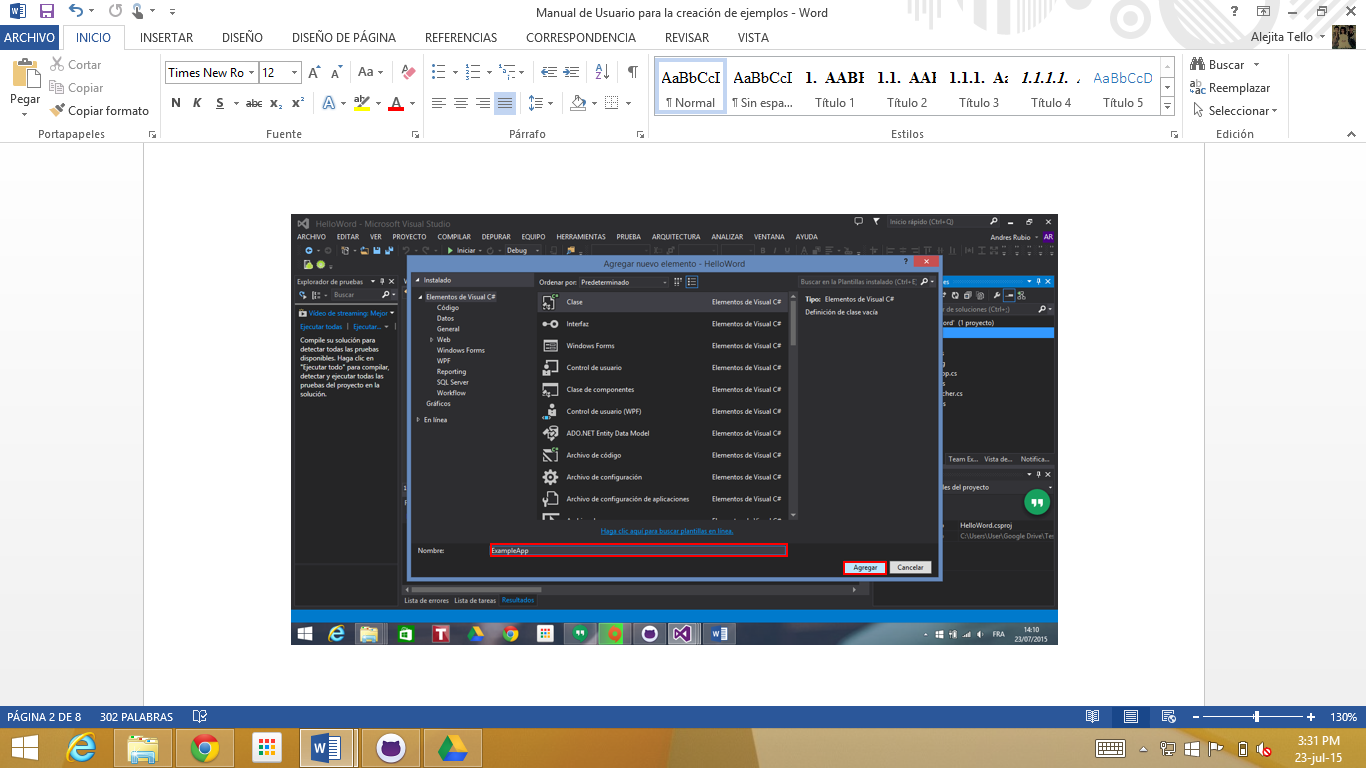
### Creación de clases adicionales

#### Creación de la clase ExampleApp.cs

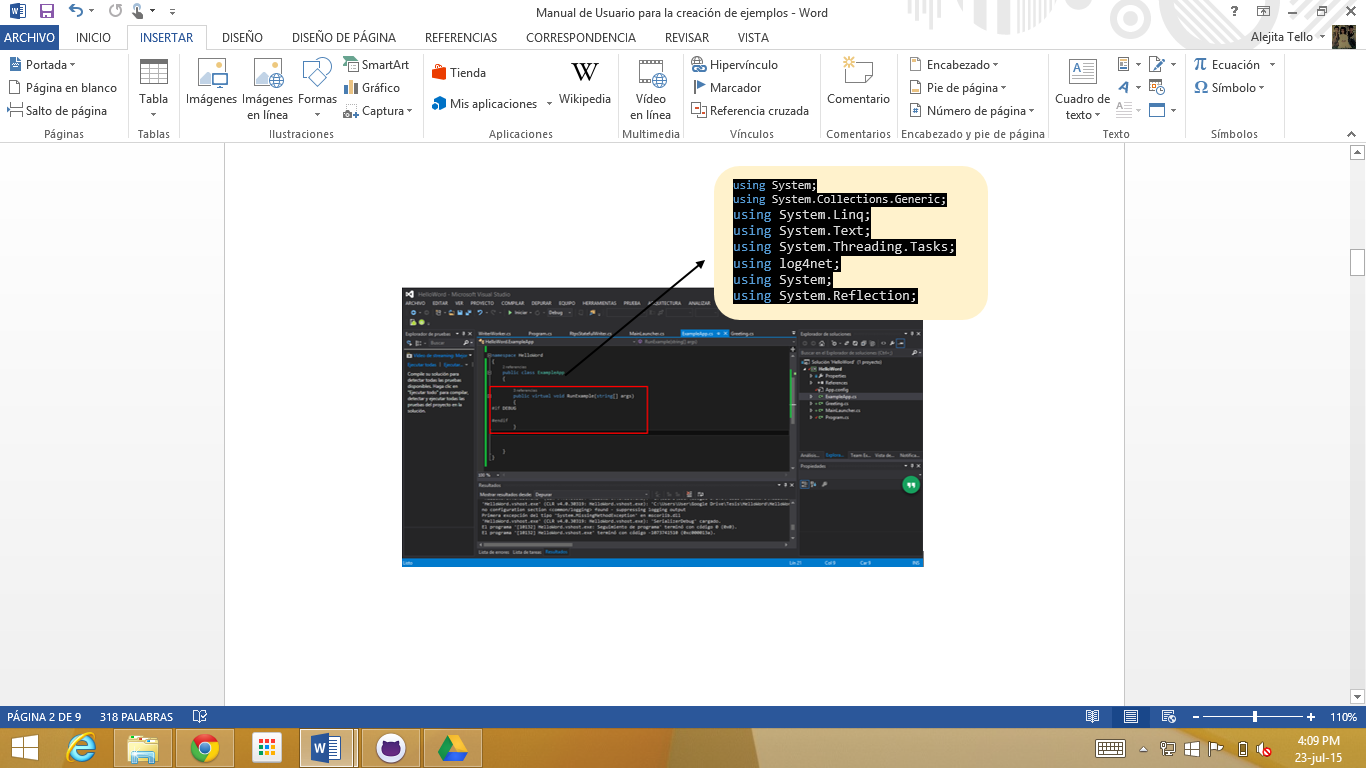
1. Clic derecho en el proyecto creado anteriormente, clic en agregar - Clase.



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

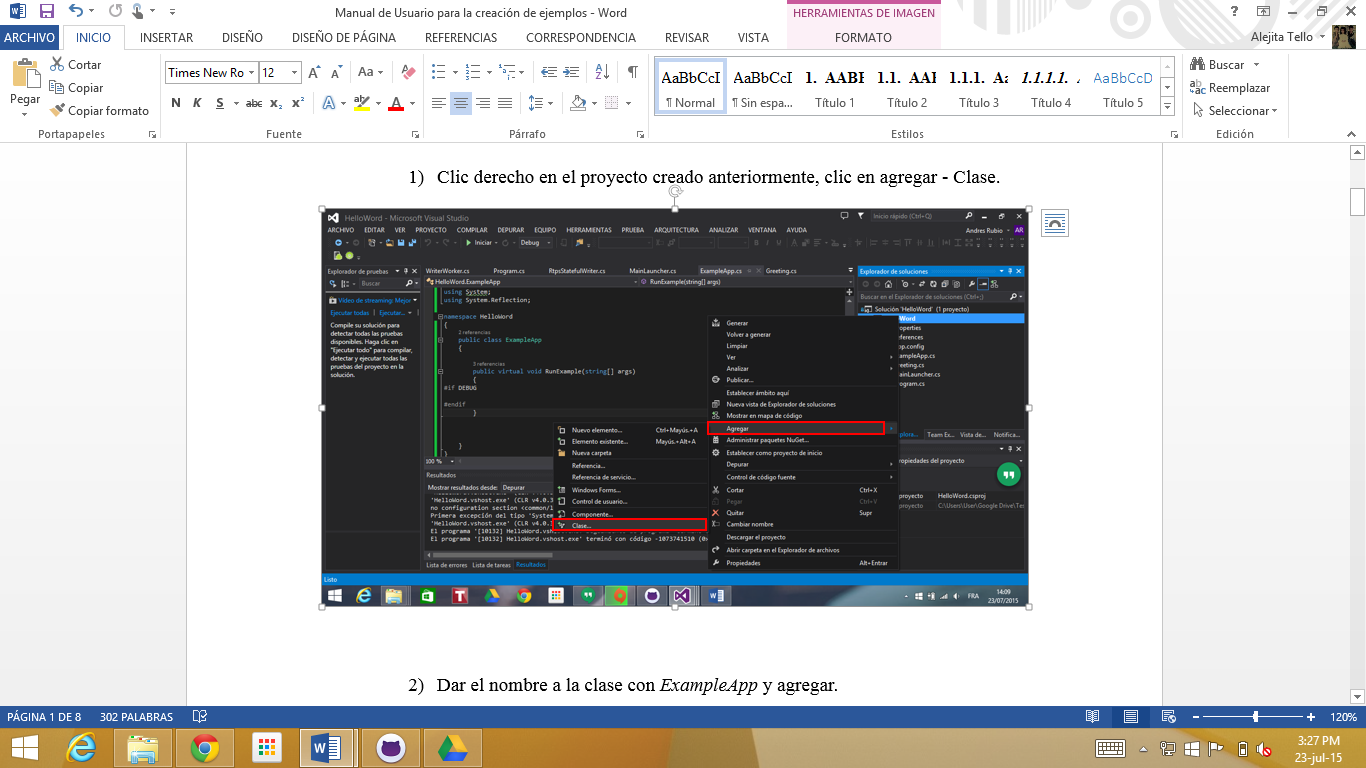


1. Dentro de la clase *ExampleApp* crear el método *Public Virtual RunExample*, como se muestra en la figura. Verificar que las siguientes referencias se encuentren añadidas.

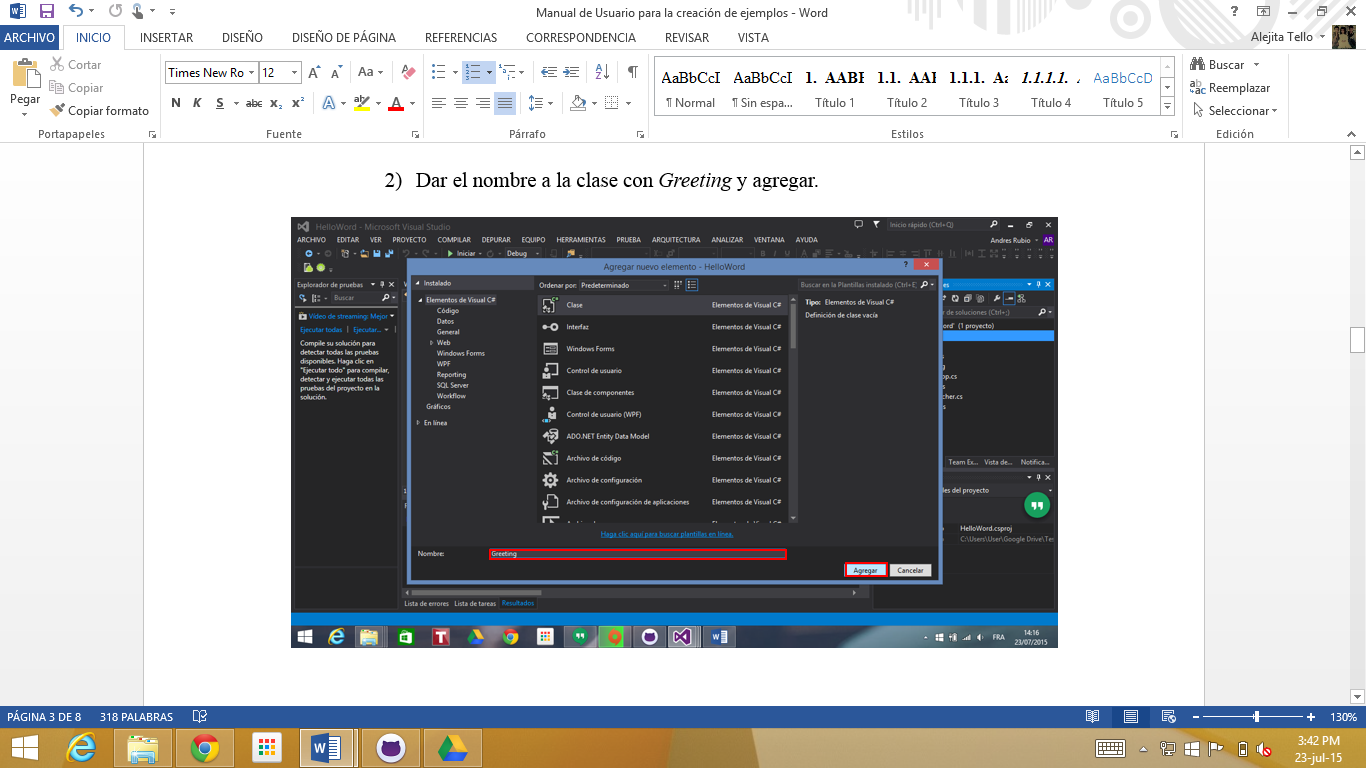


#### Creación de la clase Greeting.cs

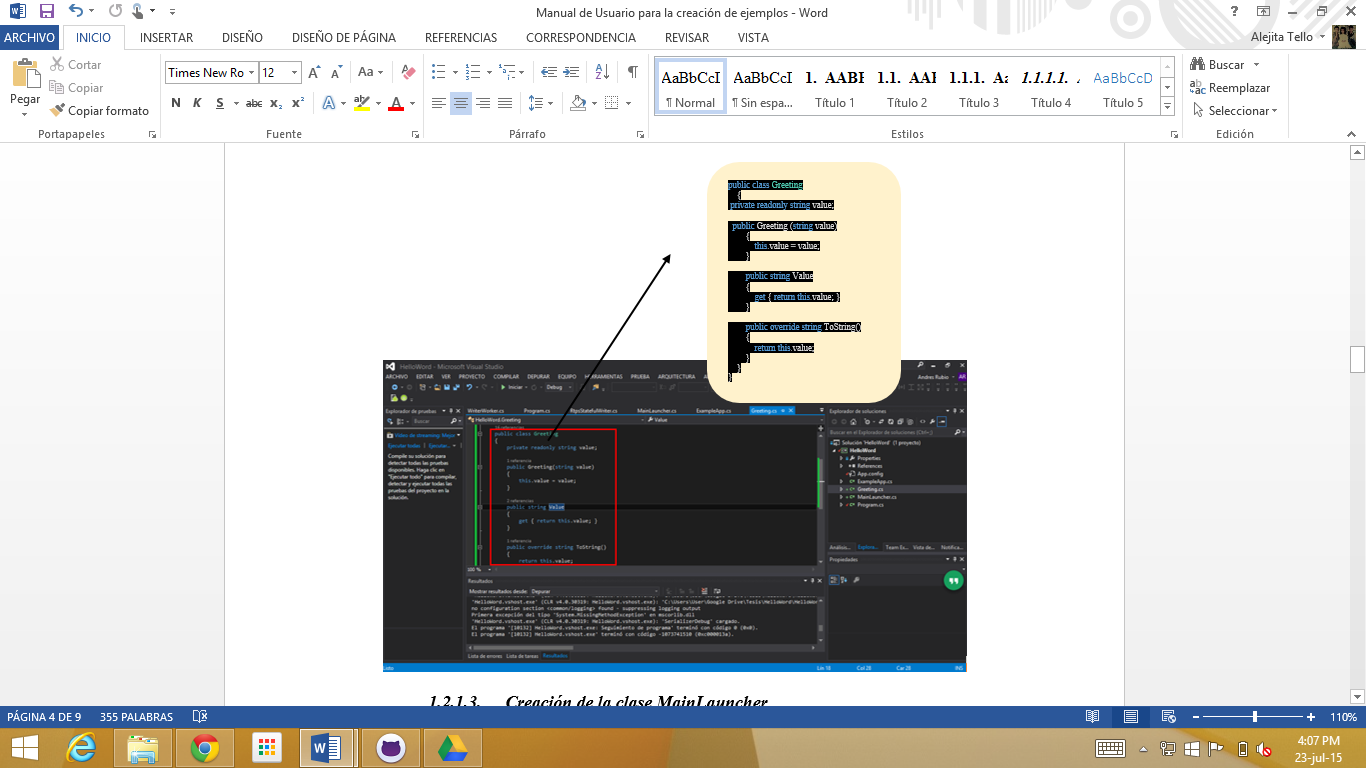
1. Clic derecho en el proyecto creado anteriormente, clic en agregar - Clase.



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

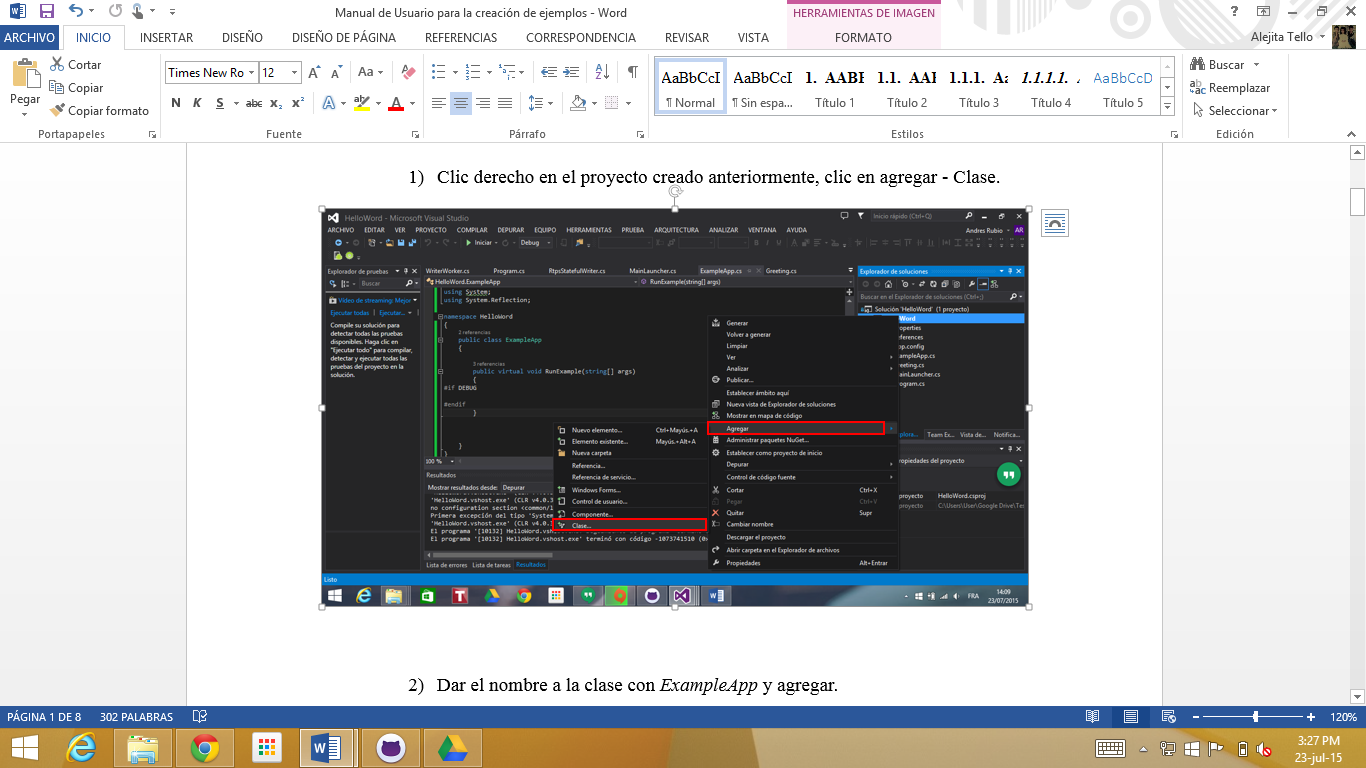


1. Dentro de la clase *Greeting* agregar las instancias básicas de los atributos de las clases.

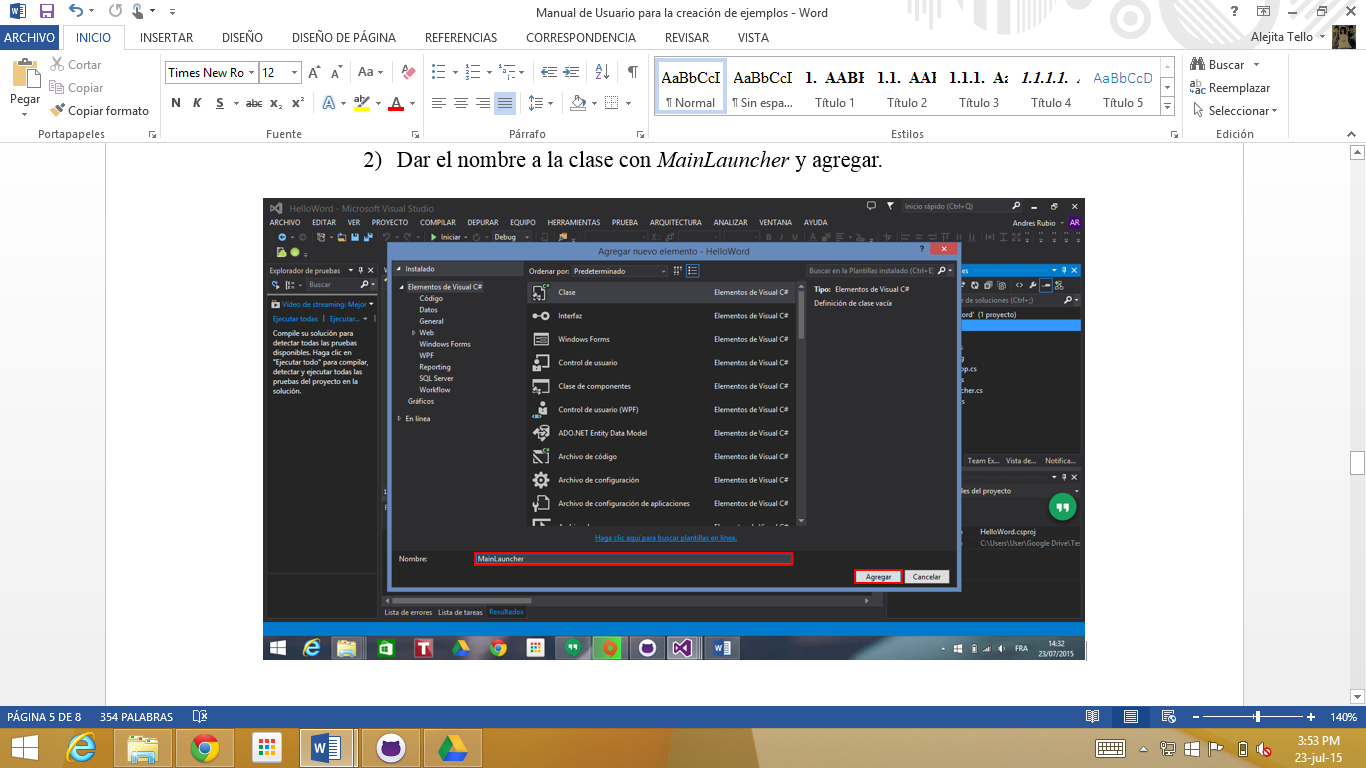


#### Creación de la clase MainLauncher

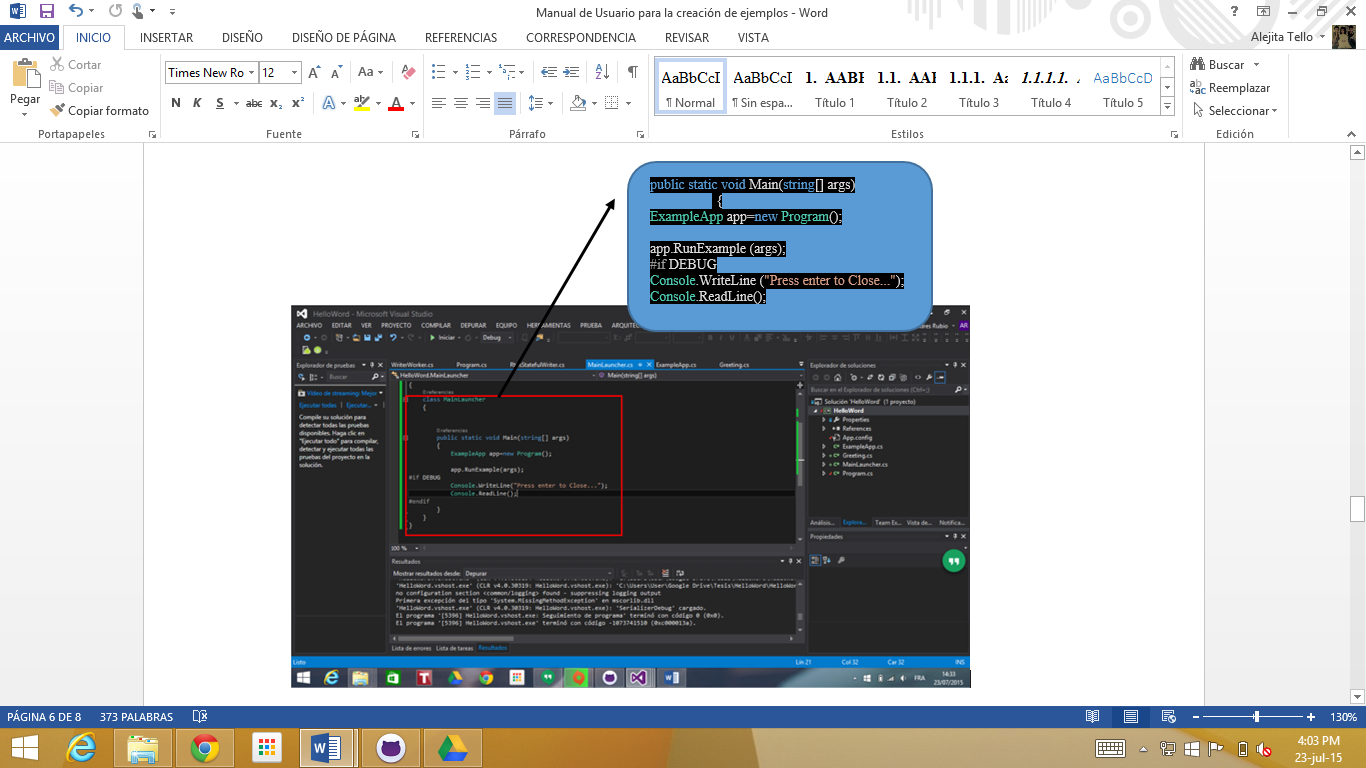
1. Clic derecho en el proyecto creado anteriormente, clic en agregar - Clase.



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).



#### Modificación del archivo de configuración App.config

El archivo de configuración se modifica de acuerdo a los siguientes parámetros. La explicación de la utilización de la configuración se encuentra especificada en el archivo denominado DESCRIPCIÓN ARCHIVO DE CONFIGURACIÓN.docx

<?xml version="1.0" encoding="utf-8" ?>

<configuration>

<configSections>

<section name="DDS" type="Doopec.Configuration.DDSConfigurationSection, Doopec, Version=0.1.\*, Culture=neutral, PublicKeyToken=null"/>

<section name="RTPS" type="Doopec.Configuration.RTPSConfigurationSection, Doopec, Version=0.1.\*, Culture=neutral, PublicKeyToken=null"/>

</configSections>

<DDS xmlns="urn:Configuration" vendor="Doopec" version="2.1">

<!--

This is just a minimal sample configuration file that shows how to declare

the configuration sections.

Because an XML Schema Definition (XSD) is generated for each configuration

section, it should be trivial to edit these files because you have

IntelliSense on the XML definition.

-->

<boostrapType name="default" type="Doopec.Dds.Core.BootstrapImpl, Doopec"/>

<domains>

<domain name="Servidor" id="0">

<transportProfile name="defaultRtps"/>

<qoSProfile name="defaultQoS"/>

<guid kind="Fixed" val="7F294ABE-33F2-40B9-BFF5-7D9376EA061C"/>

</domain>

<domain name="Servidor" id="3">

<transportProfile name="defaultRtps"/>

<qoSProfile name="defaultQoS"/>

<guid kind="Fixed" val="7F294ABE-33F2-40B9-BFF5-7D9376EA061C"/>

</domain>

<domain name="Cliente1" id="1">

<transportProfile name="defaultRtps"/>

<qoSProfile name="defaultQoS"/>

<guid kind="Fixed" val="7F294ABE-33F2-40B9-BFF5-7D9376EA061C"/>

</domain>

<domain name="Cliente2" id="2">

<transportProfile name="defaultRtps"/>

<qoSProfile name="defaultQoS"/>

<guid kind="Fixed" val="7F294ABE-33F2-40B9-BFF5-7D9376EA061C"/>

</domain>

</domains>

<logLevel levelMin="DEBUG" levelMax="FATAL"/>

<qoSProfiles>

<qoSProfileDef name="defaultQoS">

<domainParticipantFactoryQos name="defaultDomainParticipantFactoryQoS">

<entityFactory autoenableCreatedEntities="true"/>

</domainParticipantFactoryQos>

<domainParticipantQos name="defaultDomainParticipantQoS">

<entityFactory autoenableCreatedEntities="true"/>

<userData value=""/>

</domainParticipantQos>

<topicQoS name="defaultTopicQoS">

<topicData value=""/>

<deadline period="100"/>

<durability kind="VOLATILE"/>

</topicQoS>

<publisherQoS name="defaultPublisherQoS">

<entityFactory autoenableCreatedEntities="true"/>

<groupData value=""/>

<partition value=""/>

<presentation accessScope="INSTANCE" coherentAccess="true" orderedAccess="true"/>

</publisherQoS>

<subscriberQoS name="defaultSubscriberQoS">

<entityFactory autoenableCreatedEntities="true"/>

<groupData value=""/>

<partition value=""/>

<presentation accessScope="INSTANCE" coherentAccess="true" orderedAccess="true"/>

</subscriberQoS>

<dataWriterQoS name="defaultDataWriterQoS">

<deadline period="1"/>

<destinationOrder kind="BY\_SOURCE\_TIMESTAMP"/>

<durability kind="VOLATILE"/>

<durabilityService historyDepth="0" historyKind="KEEP\_LAST" maxInstances="1" maxSamples="1" maxSamplesPerInstance="1" serviceCleanupDelay="100"/>

<history kind="KEEP\_LAST" depth="1"/>

<latencyBudget duration="100"/>

<lifespan duration="100"/>

<liveliness kind="AUTOMATIC" leaseDuration="100"/>

<ownership kind="SHARED"/>

<ownershipStrength value="100"/>

<reliability kind="RELIABLE" maxBlockingTime="1000"/>

<resourceLimits maxInstances="1" maxSamples="1" maxSamplesPerInstance="1"/>

<transportPriority value="1"/>

<userData value=""/>

<writerDataLifecycle autodisposeUnregisteredInstances="true"/>

</dataWriterQoS>

<dataReaderQoS name="defaultDataReaderQoS">

<deadline period="1"/>

<destinationOrder kind="BY\_SOURCE\_TIMESTAMP"/>

<durability kind="VOLATILE"/>

<history kind="KEEP\_LAST" depth="1"/>

<latencyBudget duration="100"/>

<liveliness kind="AUTOMATIC" leaseDuration="100"/>

<ownership kind="SHARED"/>

<reliability kind="RELIABLE" maxBlockingTime="1000"/>

<resourceLimits maxInstances="1" maxSamples="1" maxSamplesPerInstance="1"/>

<readerDataLifecycle autopurgeDisposedSamplesDelay="1000" autopurgeNowriterSamplesDelay="1000"/>

<timeBasedFilter minimumSeparation="1000"/>

<userData value=""/>

</dataReaderQoS>

</qoSProfileDef>

</qoSProfiles>

</DDS>

<RTPS xmlns="urn:Configuration">

<!--

This is just a minimal sample configuration file that shows how to declare

the configuration sections.

Because an XML Schema Definition (XSD) is generated for each configuration

section, it should be trivial to edit these files because you have

IntelliSense on the XML definition.

-->

<transports>

<transport name="defaultRtps" type="Doopec.Rtps.RtpsTransport.RtpsEngine, Doopec">

<ttl val="1"/>

<discovery name="defaultDiscovery">

<resendPeriod val="30000"/>

<useSedpMulticast val="true"/>

<portBase val="7400"/>

<domainGain val="250"/>

<participantGain val="2"/>

<offsetMetatrafficMulticast val="0"/>

<offsetMetatrafficUnicast val="10"/>

<metatrafficUnicastLocatorList val="localhost"/>

<metatrafficMulticastLocatorList val="239.255.0.1"/>

</discovery>

<rtpsWriter>

<heartbeatPeriod val="1000"/>

<nackResponseDelay val="200"/>

<nackSuppressionDuration val="0"/>

<pushMode val="true"/>

</rtpsWriter>

<rtpsReader>

<heartbeatResponseDelay val="500"/>

<heartbeatSuppressionDuration val="0"/>

</rtpsReader>

</transport>

</transports>

</RTPS>

<appSettings>

<add key="org.omg.dds.serviceClassName" value="Doopec.Dds.Core.BootstrapImpl, Doopec" />

</appSettings>

<startup>

<supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.5.1" />

</startup>

</configuration>

### 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);

}

}

}

}

## resultado

En el siguiente gráfico se muestra el resultado de la compilación.

