

Practico N°3 Comunicación Serial y Socket

Ejercicio 2: “Comunicación socket (Cliente-Servidor)”

Objetivo: Familiarizar al alumno con la comunicación a través de sockets, en una aplicación cliente-servidor.

2. Creación de aplicación Servidor

Para crear la aplicación “servidor” debe seguir los siguientes pasos:

2.1 Creando el Proyecto

Cree un nuevo proyecto de tipo Aplicación de Windows Forms.

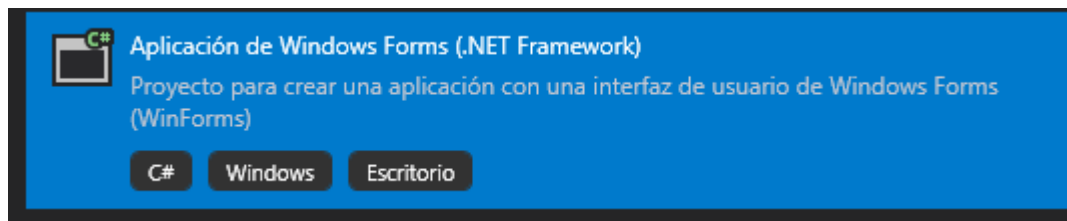


Figura 3. Creación de proyecto

2.2 Incorporación de directivas

Haciendo clic sobre el **Form** principal, se abrirá una sección de código en la que debe agregar las siguientes bibliotecas:

```
using System.Windows.Forms;
using System.IO.Ports;
using System.IO;
```

2.3 Activador de conexiones

Añada **Buttons**, **Labels** y **TextBox's** al Form principal guiándose por la figura 4.

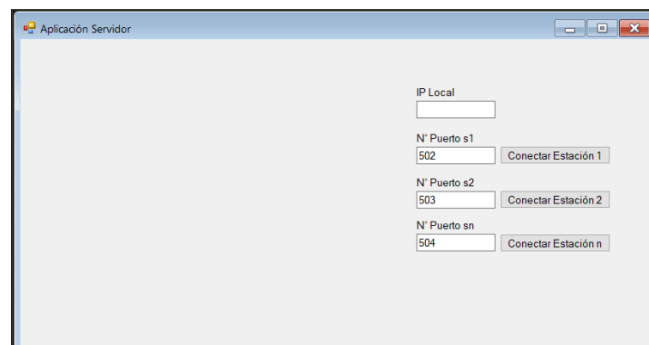


Figura 4: Primeros Controles en el Form Principal

Una vez agregados los controles, edite algunos atributos de los objetos gráficos, guiándose por la siguiente tabla:

Propiedad	Button	Button	Button	TextBox	TextBox	TextBox	TextBox	Label	Label	Label	Label
Name	B_C_s1	B_C_s2	B_C_sn	IP_local	N_puerto_s1	N_puerto_s2	N_puerto_sn	l_IP_local	l_P_s1	l_P_s2	l_P_sn
Text	Conectar Estación 1	Conectar Estación 2	Conectar Estación n		502	503	504	IP Local	N° Puerto s1	N° Puerto s2	N° Puerto sn

Luego haga doble clic en el botón “**Conectar Estación 1**”, “**Conectar Estación 2**” y “**Conectar Estación n**”, para poder programar sus funciones, añadiendo el siguiente código:

Botón “Conectar Estación 1”:

```
1 reference
private void B_C_s1_Click(object sender, EventArgs e)
{
    C_s1 = new Thread(C_Estacion_1);

    C_s1.Start();
    B_C_s1.Text = "Conexion s1 en espera";
}
```

Botón “Conectar Estación 2”:

```
1 reference
private void B_C_s2_Click(object sender, EventArgs e)
{
    C_s2 = new Thread(C_Estacion_2);

    C_s2.Start();
    B_C_s2.Text = "Conexion s2 en espera";
}
```

Botón “Conectar Estación n”:

```
1 reference
private void Button1_Click(object sender, EventArgs e)
{
    C_sn = new Thread(C_Estacion_n);

    C_sn.Start();
    B_C_sn.Text = "conexion sn en espera";
}
```

2.4 Envío de mensajes

Añada controles gráficos **Button**, **Label** y **TextBox** al Form principal, de acuerdo con la figura 5.

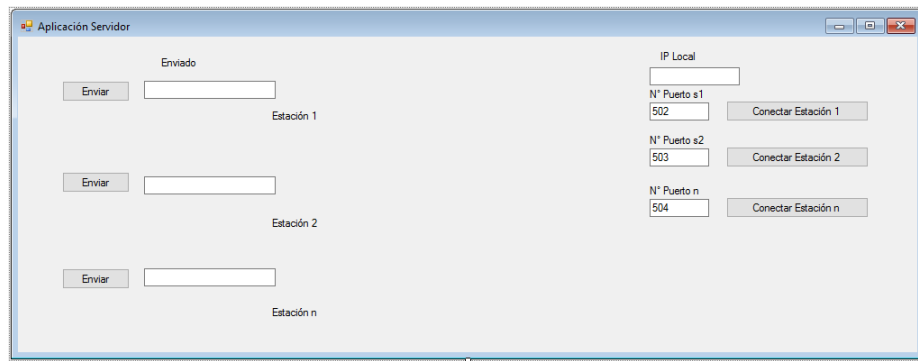


Figura 5. Controles de envío de mensajes

Una vez agregados los controles, edite algunos atributos de los objetos gráficos de la siguiente manera:

Propiedad	Button	Button	Button	TextBox	TextBox	TextBox	Label	Label	Label	Label
Name	B_enviar_s1	B_enviar_s2	B_enviar_sn	Mensaje_E_S1	Mensaje_E_S2	Mensaje_E_Sn	l_envio	l_s1	l_s2	l_sn
Text	Enviar	Enviar	Enviar				Enviado	Estación 1	Estación 2	Estación n

Luego haga doble clic en el botón **“Enviar”** de estación 1, estación 2 y estación n, para poder programar su función con el siguiente código:

Botón “Enviar” Estación 1:

```
1 reference
private void B_enviar_s1_Click(object sender, EventArgs e)
{
    byte_s1_Enviar = System.Text.Encoding.ASCII.GetBytes(Mensaje_E_S1.Text);
    stream_s1.Write(byte_s1_Enviar, 0, byte_s1_Enviar.Length);
}
```



UNIVERSIDAD DEL BÍO-BÍO
La Universidad de la Región del Biobío

Laboratorio de MAC-IPP
Profesor: Luís Vera Quiroga
Alumno ayudante: José I. Veloso Inzunza

Botón “Enviar” Estación 2:

```
1 reference
private void B_enviar_s2_Click(object sender, EventArgs e)
{
    byte_S2_Enviar = System.Text.Encoding.ASCII.GetBytes(Mensaje_E_S2.Text);
    stream_s2.Write(byte_S2_Enviar, 0, byte_S2_Enviar.Length);
}
}
```

Botón “Enviar” Estación n:

```
1 reference
private void B_enviar_sn_Click(object sender, EventArgs e)
{
    byte_Sn_Enviar = System.Text.Encoding.ASCII.GetBytes(Mensaje_E_sn.Text);
    stream_sn.Write(byte_Sn_Enviar, 0, byte_Sn_Enviar.Length);
}
}
```

2.5 Recepción de mensajes

Añada controles gráficos **Button**, **Label** y **TextBox** al Form principal tal como la figura 6.

The screenshot shows a Windows application titled "Aplicación Servidor". The interface is divided into two main sections. On the left, there are three identical sets of controls for "Estación 1", "Estación 2", and "Estación n". Each station has an "Enviar" button, a text input field, and a "Recibir" button, with a label below the input field. On the right, there is a section for "IP Local" with a text input field. Below this, there are three rows for "N° Puerto s1", "N° Puerto s2", and "N° Puerto n", each with a text input field and a "Conectar Estación" button. The input fields for ports s1 and s2 contain the values "502" and "503" respectively, while the input field for port n contains "504".

Figura 6. Controles de recepción de mensajes

Una vez agregados los controles, edite algunos atributos de los objetos gráficos de la siguiente manera:

Propiedad	Button	Button	Button	TextBox	TextBox	TextBox	Label
Name	B_Recibir_S1	B_Recibir_S2	B_Recibir_Sn	M_s1_recibido	M_s2_recibido	M_sn_recibido	l_recepción
Text	Recibir	Recibir	Recibir				Recibido

Luego haga doble clic en el botón **“Recibir”** estación 1, estación 2 y estación n, para poder programar su función con el siguiente código:

Botón “Recibir” Estación 1:

```
1 reference
private void B_Recibir_S1_Click(object sender, EventArgs e)
{
    S1_R = new Thread(Recibir_Estacion_1);
    S1_R.Start();
}
```

Botón “Recibir” Estación 2:

```
1 reference
private void B_Recibir_S2_Click(object sender, EventArgs e)
{
    S2_R = new Thread(Recibir_Estacion_2);
    S2_R.Start();
}
```

Botón “Recibir” Estación n:

```
1 reference
private void B_Recibir_Sn_Click(object sender, EventArgs e)
{
    Sn_R = new Thread(Recibir_Estacion_n);
    Sn_R.Start();
}
```



UNIVERSIDAD DEL BÍO-BÍO
La Universidad de la Región del Biobío

Laboratorio de MAC-IPP
Profesor: LuíS Vera Quiroga
Alumno ayudante: José I. Veloso Inzunza

2.6 Crear funciones necesarias

Función “GetlocalIp”:

```
1 reference
public String GetlocalIp()
{
    IPEndPoint host;
    host = Dns.GetHostEntry(Dns.GetHostName());
    foreach (IPAddress ip in host.AddressList)
    {
        if (ip.AddressFamily == AddressFamily.InterNetwork)
            return ip.ToString();
    }

    return "127.0.0.1";
}
```

Botón “Enviar” Estación 1:

```
1 reference
public void C_Estacion_1()
{
    try
    {
        tcpListener_s1 = new TcpListener(IPAddress.Any, Convert.ToInt32(N_puerto_s1.Text));
        // correccion estacion 1
        tcpListener_s1.Start();
    }
    catch (Exception e_s1)
    {
        output = "ERROR: " + e_s1.ToString();
        MessageBox.Show(output);
    }

    try
    {
        tcpClient_s1 = tcpListener_s1.AcceptTcpClient();
        stream_s1 = tcpClient_s1.GetStream();
    }
    catch (Exception e_s1)
    {
        output = "Error: " + e_s1.ToString();
        MessageBox.Show(output);
    }

    B_C_s1.Text = "conexión s1 establecida";
}
```



UNIVERSIDAD DEL BÍO-BÍO
La Universidad de la Región del Biobío

Laboratorio de MAC-IPP
Profesor: Luís Vera Quiroga
Alumno ayudante: José I. Veloso Inzunza

Función “C_Estacion_2”:

```
1 reference
public void C_Estacion_2()
{
    try
    {
        tcpListener_s2 = new TcpListener(IPAddress.Any, Convert.ToInt32(N_puerto_s2.Text)); // conexión con estación 2
        tcpListener_s2.Start();
    }
    catch (Exception e_s2)
    {
        output = "Error: " + e_s2.ToString();
        MessageBox.Show(output);
    }

    try
    {
        tcpClient_s2 = tcpListener_s2.AcceptTcpClient();
        stream_s2 = tcpClient_s2.GetStream();
    }
    catch (Exception e_s2)
    {
        output = "Error: " + e_s2.ToString();
        MessageBox.Show(output);
    }

    B_C_s2.Text = "conexión s2 establecida";
}
}
```

Función “C_Estacion_n”:

```
1 reference
public void C_Estacion_n()
{
    try
    {
        tcpListener_sn = new TcpListener(IPAddress.Any, Convert.ToInt32(N_puerto_sn.Text)); // conexión con otra estación
        tcpListener_sn.Start();
    }
    catch (Exception e_sn)
    {
        output = "Error: " + e_sn.ToString();
        MessageBox.Show(output);
    }

    try
    {
        tcpClient_sn = tcpListener_sn.AcceptTcpClient();
        stream_sn = tcpClient_sn.GetStream();
    }
    catch (Exception e_sn)
    {
        output = "Error: " + e_sn.ToString();
        MessageBox.Show(output);
    }

    B_C_sn.Text = "conexión sn establecida";
}
}
```



UNIVERSIDAD DEL BÍO-BÍO
La Universidad de la Región del Biobío

Laboratorio de MAC-IPP
Profesor: Luíís Vera Quiroga
Alumno ayudante: José I. Veloso Inzunza

Función “Recibir_Estacion_1”:

```
1 reference
public void Recibir_Estacion_1()
{
    while (true)
    {
        B_Recibir_S1.Text = "En espera...";
        Thread.Sleep(1000);
        stream_s1.Read(byte_S1_Recibir, 0, byte_S1_Recibir.Length);
        mstrMessage_s1 = Encoding.ASCII.GetString(byte_S1_Recibir, 0, byte_S1_Recibir.Length);
        mstrMessage_s1 = mstrMessage_s1.Substring(0, 5);
        B_Recibir_S1.Text = "Recibido";
        M_s1_recibido.Text = mstrMessage_s1;
    }
}
```

Función “Recibir_Estacion_2”:

```
1 reference
public void Recibir_Estacion_2()
{
    while (true)
    {
        B_Recibir_S2.Text = "En espera...";
        Thread.Sleep(1000);
        stream_s2.Read(byte_S2_Recibir, 0, byte_S2_Recibir.Length);
        mstrMessage_s2 = Encoding.ASCII.GetString(byte_S2_Recibir, 0, byte_S2_Recibir.Length);
        mstrMessage_s2 = mstrMessage_s2.Substring(0, 5);
        B_Recibir_S2.Text = "Recibido";
        M_s2_recibido.Text = mstrMessage_s2;
    }
}
```

Función “Recibir_Estacion_n”:

```
1 reference
public void Recibir_Estacion_n()
{
    while (true)
    {
        B_Recibir_Sn.Text = "En espera...";
        Thread.Sleep(1000);
        stream_sn.Read(byte_Sn_Recibir, 0, byte_Sn_Recibir.Length);
        mstrMessage_sn = Encoding.ASCII.GetString(byte_Sn_Recibir, 0, byte_Sn_Recibir.Length);
        mstrMessage_sn = mstrMessage_sn.Substring(0, 5);
        B_Recibir_Sn.Text = "Recibido";
        M_sn_recibido.Text = mstrMessage_sn;
    }
}
```




UNIVERSIDAD DEL BÍO-BÍO
La Universidad de la Región del Biobío

Laboratorio de MAC-IPP
Profesor: Luís Vera Quiroga
Alumno ayudante: José I. Veloso Inzunza

2.7 Crear variables globales

Nota: Para la creación de variables globales, deben ser declaradas dentro de la clase Form1.

```
public partial class Form1 : Form
```

Variables:

```
TcpListener tcpListener_s1 = null;
TcpListener tcpListener_s2 = null;
TcpListener tcpListener_sn = null;

TcpClient tcpClient_s1 = null;
NetworkStream stream_s1 = null;
TcpClient tcpClient_s2 = null;
NetworkStream stream_s2 = null;
TcpClient tcpClient_sn = null;
NetworkStream stream_sn = null;

string output = "";

byte[] byte_s1_Recibir = new byte[256];
byte[] byte_s2_Recibir = new byte[256];
byte[] byte_sn_Recibir = new byte[256];
byte[] byte_s1_Enviar = new byte[256];
byte[] byte_s2_Enviar = new byte[256];
byte[] byte_sn_Enviar = new byte[256];

string mstrMessage_s1;
string mstrMessage_s2;
string mstrMessage_sn;

Thread C_s1;
Thread C_s2;
Thread C_sn;
Thread S1_R;
Thread S2_R;
Thread Sn_R;
```

2.8 Comunicación entre subprocesos e inicializaciones

Debido al trabajo con “hilos”. Se debe quitar la alerta de comunicación entre procesos. Además, inicialmente se necesita contar con la dirección IP local.

Se escriben las siguientes líneas de código en la función principal “**Form1()**”.

Quitar alerta subprocesos e Inicializaciones:

```
1 reference
public Form1()
{
    InitializeComponent();
    CheckForIllegalCrossThreadCalls = false;
    IP_local.Text = GetlocalIp();
}
1 reference
```

2.9 Ejecutar la aplicación

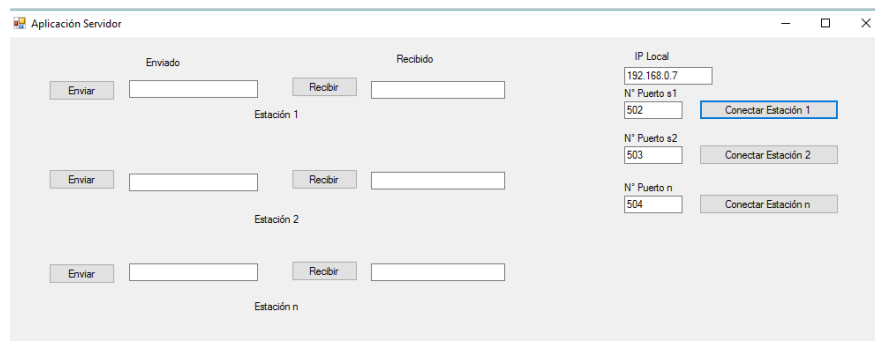


Figura 7. Forma final de aplicación Servidor

3. Creación de aplicación Cliente

Para crear la aplicación “cliente” debe seguir los siguientes pasos:

3.1 Crear el Proyecto

Cree un nuevo proyecto de tipo Aplicación de Windows Forms.

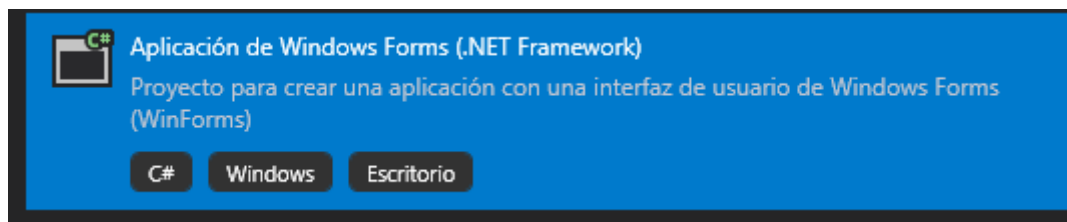


Figura 8. Creación de aplicación Cliente

3.2 Agregar directivas

Haga clic sobre el **Form** principal, se abrirá una sección de código en la que debe agregar las siguientes bibliotecas en el encabezado:

```
using System.Net;  
using System.Net.Sockets;  
using System.Threading;
```

3.3 Activación de conexión

Añada controles gráficos **Button**, **Label** y **TextBox** al Form principal de acuerdo con la figura 9.

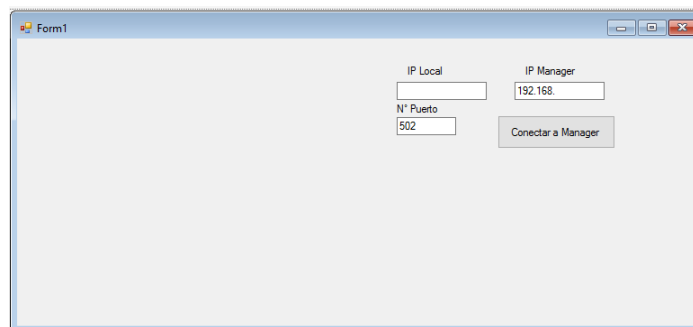


Figura 9. Primeros Controles en el Form Principal

Una vez agregados los controles, edite algunos atributos de los objetos gráficos de la siguiente manera:

Propiedad	Button	TextBox	TextBox	TextBox	Label	Label	Label
Name	B_C_M	IP_local	N_puerto	IP_manager	IP_l	N_p	IP_m
Text	Conectar a Manager		502	192.168.	IP Local	Nº Puerto	IP Manager

Luego haga doble clic en el botón “**Conectar a manager**”, para poder programar su función con el siguiente código:

Botón “Conectar a manager”:

```
C_Manager();
```

3.4 Envío y recepción de mensajes

Añada controles gráficos **Button**, **Label** y **TextBox** al Form principal de acuerdo con la figura 9.

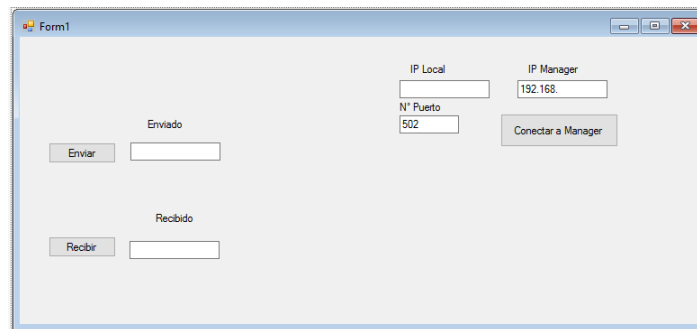


Figura 10. Controles de envío y recepción de mensajes

Una vez agregados los controles, edite algunos atributos de los objetos gráficos de la siguiente manera:

Propiedad	Button	Button	TextBox	TextBox	Label	Label
Name	B_Enviar_M	B_Recibir_M	Mensaje_E_M	Mensaje_R_M	l_E	l_R
Text	Enviar	Recibir			Enviado	Recibido



UNIVERSIDAD DEL BÍO-BÍO
La Universidad de la Región del Biobío

Laboratorio de MAC-IPP
Profesor: Luís Vera Quiroga
Alumno ayudante: José I. Veloso Inzunza

Luego haga doble clic en el botón “**Enviar**”, para poder programar su función con el siguiente código:

Botón “Enviar”:

```
1 reference
private void B_Enviar_M_Click(object sender, EventArgs e)
{
    byte_M_Enviar = System.Text.Encoding.ASCII.GetBytes(Mensaje_E_M.Text);
    stream_M.Write(byte_M_Enviar, 0, byte_M_Enviar.Length);
}
}
```

Botón “Recibir”:

```
1 reference
private void Button2_Click(object sender, EventArgs e)
{
    M_R = new Thread(Recibir_Manager);
    M_R.Start();
}
}
```

3.5 Crear funciones necesarias

Función “GetlocalIp”:

```
1 reference
public String GetlocalIp()
{
    IPEndPoint host;
    host = Dns.GetHostEntry(Dns.GetHostName());
    foreach (IPAddress ip in host.AddressList)
    {
        if (ip.AddressFamily == AddressFamily.InterNetwork)
            return ip.ToString();
    }

    return "127.0.0.1";
}
}
```



UNIVERSIDAD DEL BÍO-BÍO
La Universidad de la Región del Biobío

Laboratorio de MAC-IPP
Profesor: Luís Vera Quiroga
Alumno ayudante: José I. Veloso Inzunza

Función “C_Manager”:

```
1 reference
public void C_Manager()
{
    try
    {
        TcpClient client = new TcpClient(IP_manager.Text, Convert.ToInt32(N_puerto.Text));
        stream_M = client.GetStream();
    }
    catch (Exception e_M)
    {
        output = "Error: " + e_M.ToString();
        MessageBox.Show(output);
    }

    B_C_M.Text = "Conectado a Manager";
}
1 reference
```

Función “Recibir_Manager”:

```
1 reference
public void Recibir_Manager()
{
    while (true)
    {
        B_Recibir_M.Text = "En espera...";
        Thread.Sleep(1000);
        stream_M.Read(byte_M_Recibir, 0, byte_M_Recibir.Length);
        mstrMessage = Encoding.ASCII.GetString(byte_M_Recibir, 0, byte_M_Recibir.Length);
        mstrMessage = mstrMessage.Substring(0, 5);
        B_Recibir_M.Text = "Recibido";
        Mensaje_R_M.Text = mstrMessage;
    }
}
```

3.6 Crear variables globales

Nota: Para la creación de variables globales, deben ser declaradas dentro de la clase Form1.

```
public partial class Form1 : Form
```

Variables:

```
string output = "";
NetworkStream stream_M = null;
byte[] byte_M_Recibir = new byte[256];
byte[] byte_M_Enviar = new byte[256];
string mstrMessage;
Thread M_R;
```

3.7 Comunicación entre subprocessos e inicializaciones

Debido al trabajo con “hilos”. Se debe quitar la alerta de comunicación entre procesos. Además, inicialmente se necesita contar con la dirección IP local.

Se escriben las siguientes líneas de código en la función principal **“Form1()”**.

Quitar alerta subprocessos e inicializaciones:

```
1 reference
public Form1()
{
    InitializeComponent();
    CheckForIllegalCrossThreadCalls = false;
    IP_local.Text = GetlocalIp();
}
```

3.8 Ejecutar la aplicación

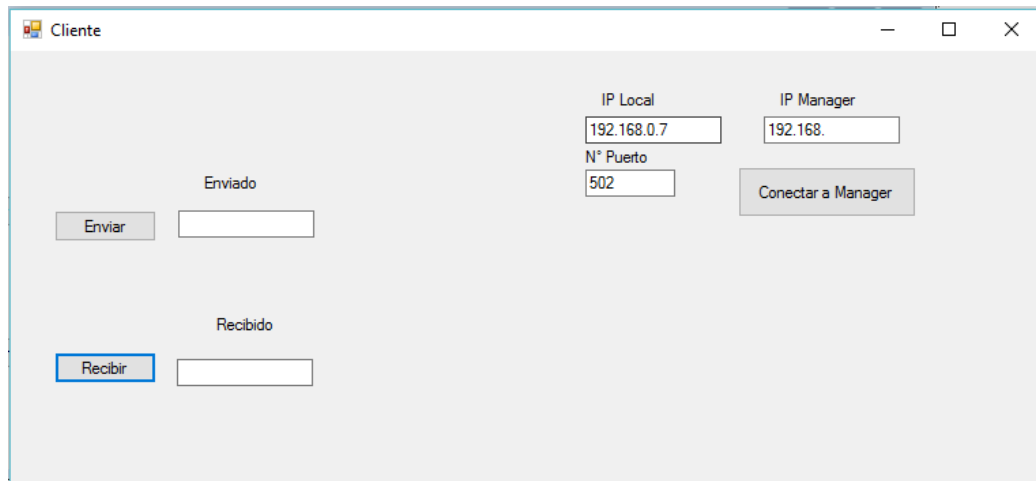


Figura 11. Forma final de aplicación Servidor

