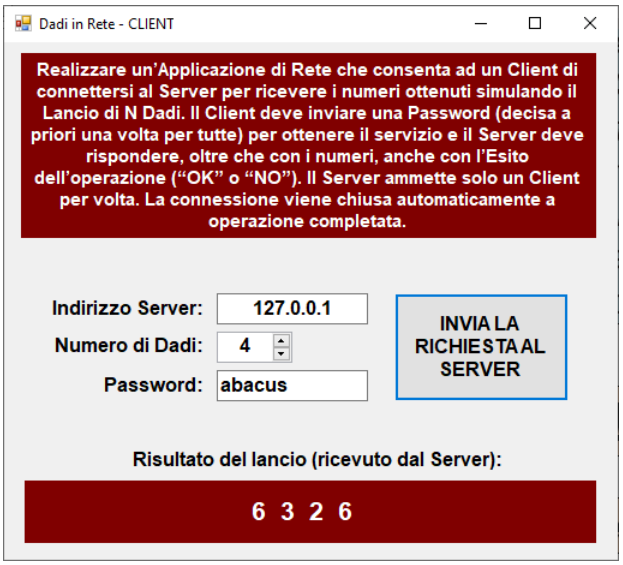
Esempio Pilota: **DadiInReteJsonConnApriChiudi**

*Realizzare un'**Applicazione di Rete** che consenta ad un **Client** di connettersi al **Server** per ricevere i numeri ottenuti simulando il **Lancio di N Dadi**. Il Client deve inviare una **Password** (decisa a priori una volta per tutte) per ottenere il servizio e il Server deve rispondere, oltre che con i numeri, anche con l'**Esito** dell'operazione ("**OK**" o "**NO**"). Il Server ammette solo un Client per volta. La connessione viene chiusa automaticamente a operazione completata.*

Applicazione CLIENT

Definizione della Form del Client



Codice della Form del Client *(Linguaggio C#)*

```
using System;
using System.Windows.Forms;

using System.Net.Sockets;
using System.Text;  // … serve per le conversioni stringa/bytes

using System.Text.Json.Nodes;
using System.Text.Json;

namespace DadiCLN
{
   public partial class Form1 : Form
   {

      public Form1()
      {
         InitializeComponent();
      }

      TcpClient CN;
```

```
      private void plsInvia_Click (object sender, EventArgs e)
      {
         // connettiti al Server...
         CN = new TcpClient ( );
         CN.Connect (txtIP.Text, 57);

         // recupera i dati dalla form...
         int NumDadi = (int) nudNumeroDadi.Value;
         string Psw = txtPassword.Text;

         // costruisci un oggetto JsonObject con i dati...
         JsonObject JO = new JsonObject()
         {
            ["NumeroDadi"] = NumDadi,
            ["Password"] = Psw
         };

         // invia i dati serializzati al Server...
         Invia (CN, JO);

         // avvia il timer per l'attesa dati...
         tmrAD.Start();
      }
```

```
      private void tmrAD_Tick (object sender, EventArgs e)
      {
         // leggi i dati ricevuti (se ce ne sono)...
         JsonObject JO  =  Ricevi (CN);

         // se ha letto qualcosa...
         if  ( JO  !=  null  )
         {
            // recupera, dai dati ricevuti, l'esito dell'operazione…
            string Esito = (string) JO["Esito"];

            if ( Esito == "OK")
            {
               // recupera, dai dati ricevuti, il vett. con i lanci dei dadi
               JsonArray JA = (JsonArray)JO["VetDadi"];

               // visualizza tutti i lanci dei dadi…
               lblRisposta.Text = "";
               for (int K = 0; K <= JA.Count - 1; K++)
               {
                  lblRisposta.Text += ( (int) JA[K] ).ToString() + "  ";
               }
            }
            else
            {
               lblRisposta.Text = "PASSWORD NON RICONOSCIUTA";
            }

            // chiudi la connessione e ferma il timer...
            CN.Close ( );
            tmrAD.Stop ( );
         }
      }
```

```
/* i seguenti sottoprogrammi "Ricevi" e "Invia" sono generali
   e possono essere riutilizzati in altri progetti.

 "Ricevi" prevede, come parametro, una connessione (CN) aperta.
          Esso legge i dati (stringa JSON) dalla connessione, li
          deserializza e restituisce un oggetto JsonObject (JO)
          contenente i dati deserializzati.                    */

    private JsonObject Ricevi ( TcpClient CN )
    {
      JsonObject JO = null;

      // se la connessione e' aperta...
      if ( CN.Connected )
      {
        // se sono arrivati bytes da leggere...
        if ( CN.Available > 0 )
        {
          // ottieni il NetworkStream della connessione...
          NetworkStream NS = CN.GetStream ( );

          // dichiara un Vettore di Byte per leggere i bytes arrivati..
          int NumeroDiByteRicevuti = CN.Available;
          byte[ ] VetDati = new byte [ NumeroDiByteRicevuti] ;

          // leggi i dati arrivati e mettili nel Vettore...
          NS.Read ( VetDati, 0, NumeroDiByteRicevuti );

          // converti in stringa, il Vettore di Byte ricevuto...
          string StringaJson =
                UnicodeEncoding.Unicode.GetString ( VetDati );

          // deserializza i dati e ponili in un JsonObject...
         JO = JsonSerializer.
                        Deserialize<JsonObject>(StringaJson);
        }
      }

      return JO;
    }

/*
 "Invia" prevede, come parametri, una connessione (CN) aperta
         e un oggetto JsonObject (JO) con i dati da inviare.
         Esso serializza i dati presenti in JO e trasmette, sulla
        connessione, la stringa JSON ottenuta serializzando.
*/

    private void Invia ( TcpClient CN, JsonObject JO)
    {
      // se la connessione e' aperta...
      if ( CN.Connected )
      {
        // serializza i dati in una StringaJSON...
        string StringaJSON = JsonSerializer.Serialize ( JO );

        // ottieni il NetworkStream della connessione...
        NetworkStream NS = CN.GetStream ( );

        // converti, in Vettore di Byte, la stringa da inviare...
        byte[ ] VetDati =
           UnicodeEncoding.Unicode.GetBytes ( StringaJSON );

        // invia, sulla connessione, i dati presenti nel Vettore...
        NS.Write ( VetDati, 0, VetDati.Length);
      }
    }

  }
}
```

Applicazione SERVER

Definizione della Form del Server

Codice della Form del Server *(Linguaggio C#)*

```
using System;
using System.Windows.Forms;

using System.Net.Sockets;
using System.Text;
using System.Text.Json.Nodes;
using System.Text.Json;

namespace DadiSRV
{
   public partial class Form1 : Form
   {
      public Form1()
      {
         InitializeComponent();
      }

      Random R = new Random();

      TcpListener LST;
      TcpClient CN;

      const string PasswordGiusta = "abacus";
```

```
      private void Form1_Load (object sender, EventArgs e)
      {
         LST = new TcpListener ( 57 );
         LST.Start ( );

         tmrAC.Start();
      }
```

```
      private void tmrAC_Tick (object sender, EventArgs e)
      {
         if ( LST.Pending ( ) )
         {
            CN = LST.AcceptTcpClient ( );

            // ferma l'attesa per le richieste di connessione...
            LST.Stop ( );

            tmrAC.Stop();

            // attiva l'attesa dati dal client...
            tmrAD.Start();
         }
      }
```

```
      private void tmrAD_Tick (object sender, EventArgs e)
      {
         string Messaggio;

         // leggi i dati ricevuti (se ce ne sono)...
         JsonObject JO = Ricevi ( CN );

         // se ha letto qualcosa...
         if ( JO != null )
         {

            // recupera i dati ricevuti dal client...
            string Psw = (string) JO["Password"];
            int NumDadi = (int) JO["NumeroDadi"];

            // se la password ricevuta è quella giusta...
            if ( Psw == PasswordGiusta )
            {
               Messaggio = "Inviati " + NumDadi + " Dadi: ";

               // crea un JsonArray JA per serializzare i lanci dei dadi...
               JsonArray JA = new JsonArray ( );

               // simula i lanci e ponili in JA...
               for (int K = 0; K <= NumDadi - 1; K++)
               {
                  int LancioDelDado = R.Next ( 1, 7 );
                  JA.Add ( LancioDelDado );

                  Messaggio += LancioDelDado.ToString() + " ";
               }

               // crea l'oggetto JsonObject con i dati da inviare...
               JsonObject JOrisp = new JsonObject ( )
               {
                  ["Esito"] = "OK",
                  ["VetDadi"] = JA
               };

               // serializza e invia al client...
               Invia ( CN, JOrisp );
            }
            else
            {
               Messaggio = "Password Errata";

               // crea l'oggetto JsonObject con i dati da inviare...
               JsonObject JOrisp = new JsonObject ( )
               {
                  ["Esito"] = "NO",
                  ["VetDadi"] = null
               };
```

```
            // serializza e invia al client...
            Invia ( CN, JOrisp );
        }

        lblUltimoInvio.Text = Messaggio;

        // chiudi la connessione e ferma l'attesa dati...
        CN.Close ( );
        tmrAD.Stop ( );

        // riattiva l'attesa per nuove richieste di connessione...
        LST.Start ( );
        tmrAC.Start ( );
    }
}
```

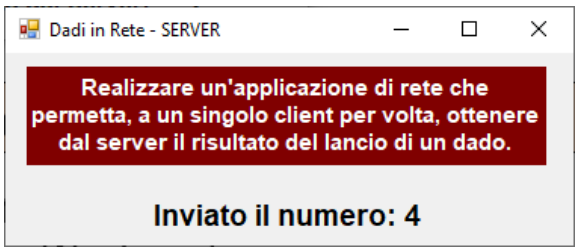
```
private JsonObject Ricevi ( TcpClient CN )
{
    ... il codice è identico a quello già visto nel Client ...
}
```

```
private void Invia ( TcpClient CN, JsonObject JO )
{
    ... il codice è identico a quello già visto nel Client ...
}
```

```
  }
}
```

Applicazione SERVER

Definizione della Form del Server



Codice della Form del Server *(Linguaggio C#)*

```
using System;
using System.Windows.Forms;

using System.Net.Sockets;

namespace DadiSRV
{
   public partial class Form1 : Form
   {
      public Form1()
      {
         InitializeComponent();
      }

      // crea l'oggetto R per generare numeri casuali ...
      Random R = new Random ( );

      TcpListener  LST; // gestisce l'attesa per richieste di conness.
      TcpClient  CN;      // gestisce la singola connessione

      private void Form1_Load (object sender, EventArgs e)
      {
         // imposta l'attesa sulla porta di rete N. 57 ...
         LST = new TcpListener (57);

         // attiva l'attesa per le richeste di connessione ...
         LST.Start ( );

         // avvia il Timer per verificare l'arrivo di richeste ...
         tmrAC.Start ( );
      }

      private void tmrAC_Tick (object sender, EventArgs e)
      {
         // se sono arrivate richieste di connessione ...
         if  (  LST.Pending ( )  )
         {
            // ottieni l'oggetto connessione dal TcpListener ...
            CN = LST.AcceptTcpClient ( );

            // ferma l'attesa di richeste e il relativo timer ...
            LST.Stop ( );
            tmrAC.Stop ( );

            // genera il lancio del dado (numero da 1 a 6) ...
            byte LancioDado = (byte) R.Next (1, 7);

            // ottieni l'oggetto flusso dati dalla connessione ...
            NetworkStream NS = CN.GetStream ( );

            // invia il numero sul flusso dati della connesione ...
            NS.WriteByte ( LancioDado );

            lblMessaggio.Text = "Inviato il numero: " + LancioDado;

            // Chiudi la connessione e riattiva l'attesa ...
            CN.Close ( );
            LST.Start ( );
            tmrAC.Start ( );
         }
      }

   }
}
```