# клиент и сервер на одном проекте но на разных формах

Создание клиент-серверного приложения в одном проекте с использованием Windows Forms (WinForms) в .NET Framework, где сервер и клиент находятся на разных формах. Ниже приведен пример, который объединяет сервер и клиент в одном проекте с использованием двух (трех) форм.

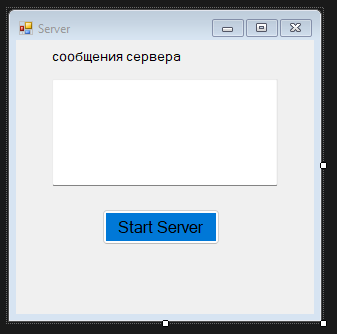
**Шаг 1: Создание проекта**

1. Откройте Visual Studio и создайте новый проект **Windows Forms App (.NET Framework)**.
2. Назовите проект, например, ClientServerApp.

**Шаг 2: Создание форм**

1. **Добавьте две формы** в проект:
   * **FormServer** — для сервера.
   * **FormClient** — для клиента.

**Шаг 3: Реализация серверной формы (FormServer)**



**1. Установка необходимых пространств имен**

В FormServer.cs добавьте следующие пространства имен:

using System;

using System.Net;

using System.Net.Sockets;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

**2. Создание интерфейса**

Добавьте на форму сервера элементы управления:

* **TextBox** (Name: textBoxMessages) для отображения сообщений сервера (многострочный).
* **Button** (Name: btnStartServer, Text: "Запустить сервер") для запуска сервера.

**3. Реализация кода для сервера**

Вот пример кода для сервера:

using System;

using System.Text;

using System.Net;

using System.Net.Sockets;

using System.Windows.Forms;

namespace Clients\_Server

{

public partial class Server : Form

{

private TcpListener listener;

public Server()

{

InitializeComponent();

}

private void buttonStartServer\_Click( object sender, EventArgs e )

{

listener = new TcpListener( IPAddress.Any, 5000 );

listener.Start();

textBoxMessages.AppendText( "Сервер запущен...\n" );

AcceptClients();

}

private async void AcceptClients()

{

while (true)

{

var client = await listener.AcceptTcpClientAsync();

textBoxMessages.AppendText( "Клиент подключен...\n" );

HandleClient( client );

}

}

private async void HandleClient( TcpClient client )

{

NetworkStream stream = client.GetStream();

byte[] buffer = new byte[1024];

int bytesRead;

while ((bytesRead = await stream.ReadAsync( buffer, 0, buffer.Length )) != 0)

{

string message = Encoding.UTF8.GetString( buffer, 0, bytesRead );

textBoxMessages.AppendText( $"Получено: {message}\n" );

}

client.Close();

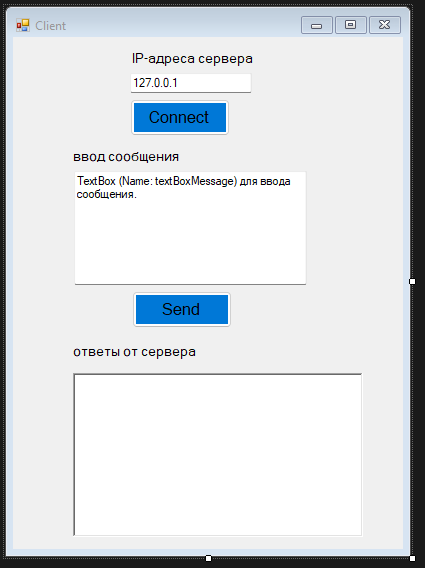
textBoxMessages.AppendText( "Клиент отключен...\n" );

}

}

}

**Шаг 4: Реализация клиентской формы (FormClient)**



**1. Установка необходимых пространств имен**

В FormClient.cs добавьте следующие пространства имен:

using System;

using System.Net.Sockets;

using System.Text;

using System.Windows.Forms;

**2. Создание интерфейса**

Добавьте на форму клиента элементы управления:

* **TextBox** (Name: textBoxIpAddress) для ввода IP-адреса сервера.
* **TextBox** (Name: textBoxMessage) для ввода сообщения.
* **Button** (Name: btnConnect, Text: "Подключиться") для подключения к серверу.
* **Button** (Name: btnSend, Text: "Отправить") для отправки сообщения.
* **RichTextBox** (Name: textBoxMessages) для отображения ответов от сервера.

**3. Реализация кода для клиента**

Вот пример кода для клиента:

using System;

using System.Net.Sockets;

using System.Text;

using System.Windows.Forms;

namespace Clients\_Server

{

public partial class Client : Form

{

private TcpClient client;

private NetworkStream stream;

public Client()

{

InitializeComponent();

}

private void buttonConnect\_Click( object sender, EventArgs e )

{

try

{

string ipAddress = textBoxIpAddress.Text;

client = new TcpClient( ipAddress, 5000 );

stream = client.GetStream();

richTextBoxMessages.AppendText( "Подключено к серверу...\n" );

}

catch (Exception ex)

{

richTextBoxMessages.AppendText( $"Ошибка: {ex.Message}\n" );

}

}

private async void buttonSend\_Click( object sender, EventArgs e )

{

if (stream != null && stream.CanWrite)

{

string message = textBoxMessage.Text;

byte[] data = Encoding.UTF8.GetBytes( message );

await stream.WriteAsync( data, 0, data.Length );

richTextBoxMessages.AppendText( $"Отправлено: {message}\n" );

}

}

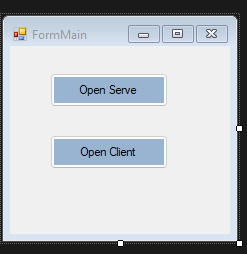
}

}

**Шаг 5: Главное окно приложения**

Затем создайте главное окно приложения (например, FormMain), которое будет содержать кнопки для открытия форм сервера и клиента.

**1. Создайте интерфейс для главного окна**



Добавьте на форму главного окна следующие элементы управления:

* **Button** (Name: btnOpenServer, Text: "Открыть сервер").
* **Button** (Name: btnOpenClient, Text: "Открыть клиент").

**2. Реализация кода для главного окна**

Вот пример кода для главного окна:

using System;

using System.Windows.Forms;

namespace Clients\_Server

{

public partial class FormMain : Form

{

public FormMain()

{

InitializeComponent();

}

private void buttonOpenServe\_Click( object sender, EventArgs e )

{

// Open Server, Открыть сервер

Server serverForm = new Server();

serverForm.Show();

}

private void buttonOpenClient\_Click( object sender, EventArgs e )

{

// Open Client, Открыть клиент

Client clientForm = new Client();

clientForm.Show();

}

}

}

**Шаг 6: Запуск приложения**

1. Убедитесь, что все формы и элементы управления правильно настроены.
2. Запустите главное окно приложения (FormMain).
3. Нажмите на кнопку для открытия серверной формы. Запустите сервер.
4. Затем откройте клиентскую форму и подключитесь к серверу, введя IP-адрес (например, 127.0.0.1 для локального сервера), и отправьте сообщение.