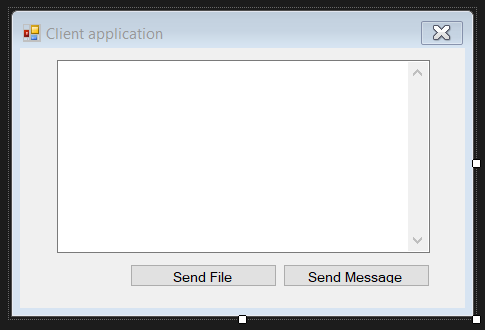
**Лабораторная работа №10**

**Ход работы:**

На форму ClientMainWindow добавляем кнопку Send File и openFileDialog.



Обработаем нажатие на кнопку SendFile. Для этого нужно создать новый метод SelectFile, который с помощью openFileDialog выберет файл и вернет полный путь до файла с его именем. Мы создаем нового клиента и выбираем файл. Как только путь до файла получен, мы должны вызвать клиентский метод по отправки файла на сервер (SendFileToServer).

private void OnFileBtnClick(object sender, EventArgs e)

{

Client client = new Client();

OperationResult res;

string path = SelectFile();

if (path != null) res = client.SendFileToServer(path);

else return;

labelRes.Text = res.Message;

timer.Interval = 2000;

timer.Start();

}

Для пересылки файла на сервер нужно создать новые подключение и сетевой поток, который будет прослушиваться листенером. Поскольку серверу нужно знать, что передается именно файл, то в первый байт потока нужно записывать значение false. Также на сервер необходимо переслать расширение файла. Далее побайтово записываем содержимое файла в массив, и этот массив байтов также передается в поток.

public OperationResult SendFileToServer(string path)

{

try

{

tcpClient = new TcpClient("127.0.0.1", 8080);

NetworkStream stream = tcpClient.GetStream();

byte[] data = File.ReadAllBytes(path);

BinaryFormatter formatter = new BinaryFormatter();

// Указываем, что передаётся файл

formatter.Serialize(stream, false);

// Указываем расширение файла

formatter.Serialize(stream, Path.GetExtension(path));

// Передаем сам файл

formatter.Serialize(stream, data);

stream.Close();

tcpClient.Close();

return new OperationResult(Result.OK, "Файл " + Path.GetFileName(path) + '.' + Path.GetExtension(path) + " отправлен.");

}

catch (Exception e)

{

return new OperationResult(Result.Fail, e.Message);

}

}

Так как теперь сервер может принимать сообщения или файл, то необходимо написать метод, который будет определять, что именно принимает сервер, и запускать соответствующие методы. Это метод ChooseData. В самом начале нужно проверить, не превышено ли допустимое количество подключений на сервере (в нашем случае их 3). Если всё в порядке, то метод продолжает работу. Из принимаемого потока по первому байту выясняем, что мы получили – файл (false) или сообщение (true).

private async Task<OperationResult> ChooseData()

{

try

{

if (cur\_cnt > max\_cnt) return new OperationResult(Result.Fail, "Server busy. Try connect Later");

TcpClient client = serverListener.AcceptTcpClient();

Interlocked.Increment(ref cur\_cnt);

NetworkStream stream = client.GetStream();

BinaryFormatter formatter = new BinaryFormatter();

bool operation = (bool)formatter.Deserialize(stream);

if (operation)

{

OperationResult result = await ReceiveMessageFromClient(stream);

if (result.Result == Result.Fail)

Console.WriteLine("Оибка: " + result.Message);

else

Console.WriteLine("Новое сообщение от клиента: " + result.Message);

}

else

{

OperationResult result = await ReceiveFileFromClient(stream);

if (result.Result == Result.Fail)

Console.WriteLine("Оибка: " + result.Message);

else

Console.WriteLine("Получен файл от клиента: " + result.Message);

}

stream.Close();

client.Close();

Interlocked.Decrement(ref cur\_cnt);

return new OperationResult(Result.OK, "");

}

catch (Exception e)

{

return new OperationResult(Result.Fail, e.Message);

}

}

Поскольку теперь обмен с сервером происходит через поток, то нужно переписать методы ReceiveMessageFromClient и SendMessageToClient. Суть первого метода в том, что полученные байты из потока декодируются обратно в сообщение, которое передается вызывающей стороне.

public async Task<OperationResult> ReceiveMessageFromClient(NetworkStream stream)

{

try

{

Console.WriteLine("Waiting for connections...");

StringBuilder recievedMessage = new StringBuilder();

byte[] data = new byte[256];

do

{

int bytes = stream.Read(data, 0, data.Length);

recievedMessage.Append(Encoding.UTF8.GetString(data, 0, bytes));

}

while (stream.DataAvailable);

stream.Close();

return new OperationResult(Result.OK, recievedMessage.ToString());

}

catch (Exception e)

{

return new OperationResult(Result.Fail, e.Message);

}

}

Напишем метод по приему файла от клиента. Из потока мы получаем расширение файла и методом GetNewPath формируем полный путь (включая название), по которому будет сохранен файл. Затем побайтово переносим содержимое файла из потока по новому пути.

public async Task<OperationResult> ReceiveFileFromClient(NetworkStream stream)

{

try

{

BinaryFormatter formatter = new BinaryFormatter();

// Получаем из потока расширение переданного файла

string extension = (string)formatter.Deserialize(stream);

// Получаем из потока побайтовое содержимое файла

byte[] data = (byte[])formatter.Deserialize(stream);

// Получаем новый путь

string newPath = GetNewPath(extension);

// Считываем всё побайтовое содержимое файла в файл newPass

File.WriteAllBytes(newPath, data);

// Увеличиваем количество загруженных файлов

Interlocked.Increment(ref file\_cnt);

stream.Close();

return new OperationResult(Result.OK, Path.GetFileName(newPath));

}

catch (Exception e)

{

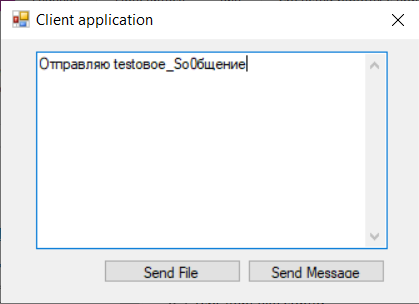
return new OperationResult(Result.Fail, e.Message);

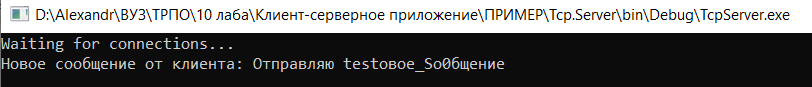
}

}

**Демонстрация**:

Отправим сообщение





Отправим файл

