**12** **Многопользовательское приложение**

Задание 1. Автоматизация учета сотрудников

Листинг задачи:

public class ChatClient : IDisposable

{

private readonly string \_pipeName;

private NamedPipeClientStream \_pipe;

private StreamReader \_reader;

private StreamWriter \_writer;

private bool \_isConnected;

public event EventHandler<string> MessageReceived;

public ChatClient(string pipeName)

{

\_pipeName = pipeName;

}

public async Task ConnectAsync()

{

\_pipe = new NamedPipeClientStream(

".",

\_pipeName,

PipeDirection.InOut,

PipeOptions.Asynchronous);

await \_pipe.ConnectAsync(5000);

\_reader = new StreamReader(\_pipe);

\_writer = new StreamWriter(\_pipe) { AutoFlush = true };

\_isConnected = true;

\_ = Task.Run(ReceiveMessagesAsync);

}

private async Task ReceiveMessagesAsync()

{

while (\_isConnected)

{

try

{

var message = await \_reader.ReadLineAsync();

if (message != null)

{

MessageReceived?.Invoke(this, message);

}

}

catch

{

Disconnect();

}

}

}

public async Task SendAsync(string message)

{

if (\_isConnected)

{

await \_writer.WriteLineAsync(message);

}

}

private void Disconnect()

{

\_isConnected = false;

Dispose();

}

public void Dispose()

{

\_reader?.Dispose();

\_writer?.Dispose();

\_pipe?.Dispose();

GC.SuppressFinalize(this);

}

}

Таблица 1.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| Сообщение | Вывод сообщения |

Анализ результатов:

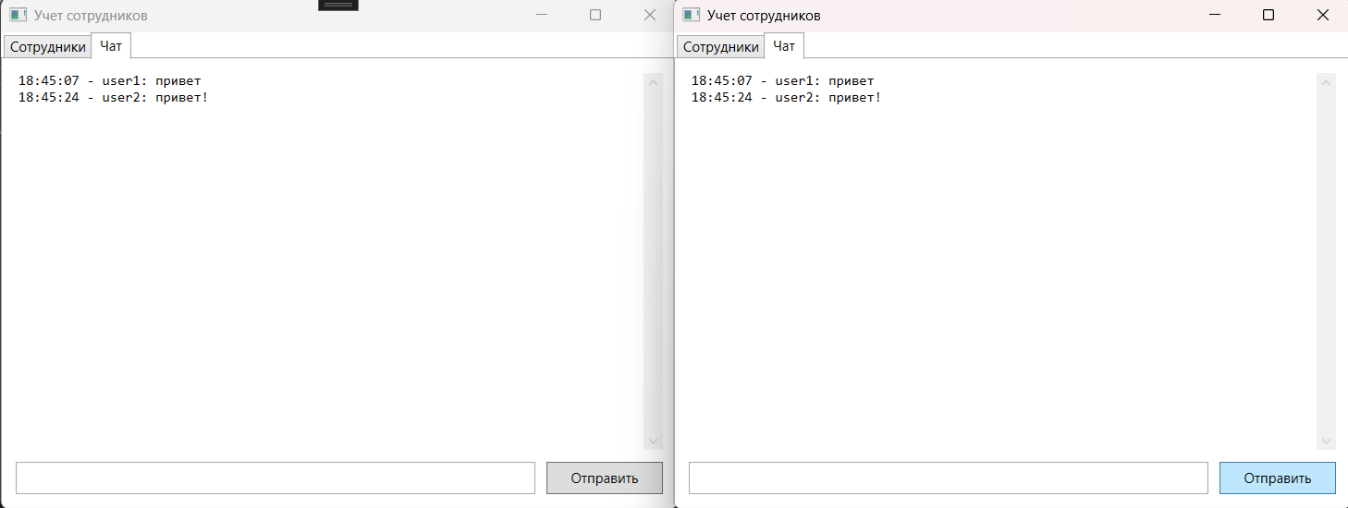


Рисунок 1.1 – Результат работы программы