МИНИСТЕРСТВО ОБРАЗОВАНИЯ ТУЛЬСКОЙ ОБЛАСТИ



Государственное профессиональное образовательное учреждение

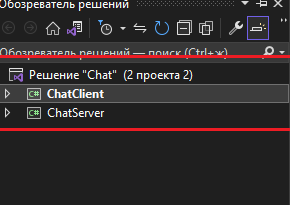
Тульской области

«ДОНСКОЙ КОЛЛЕДЖ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ»

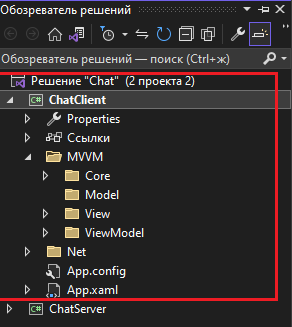
Методические указания к практической работе №15

«Разработка приложения - Чат»

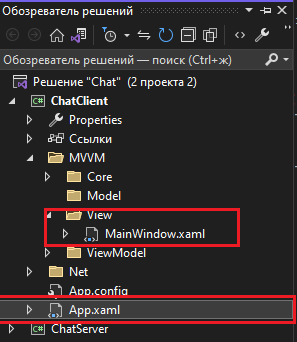
Так как наш чат будет содержать, как клиентскую так и серверную часть создадим в одном решении два проект: 1) Wpf – ChatClient; 2) ChatServer – Консольное приложение.

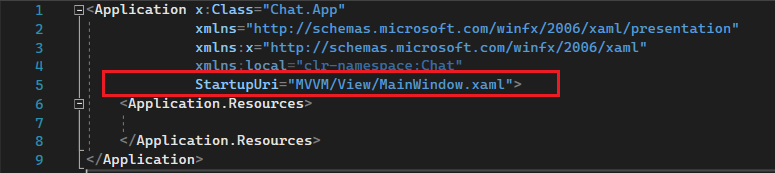


Далее переходим в ChatClient и создаем необходимые директивы: «MVVM», «View», «Model», «ViewModel», «Core», «Net».

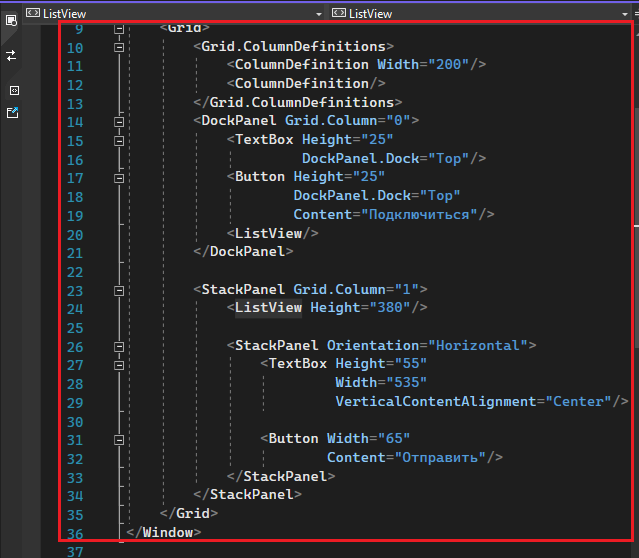


Теперь переносим «MainWindow.xaml» в директорию «View» и изменяем в файле «App.xaml» путь к MainWindow.xaml». После пересоберите и запустите проект.

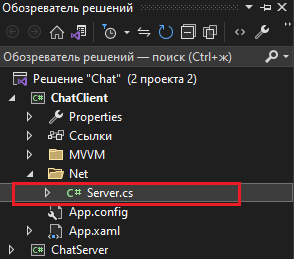


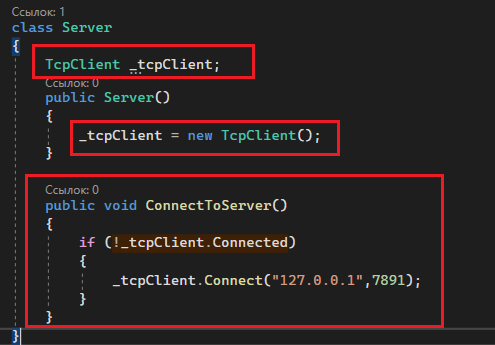


Далее переходим в «MainWindow.xaml» и делаем небольшую разметку. После пересоберите и запустите проект.

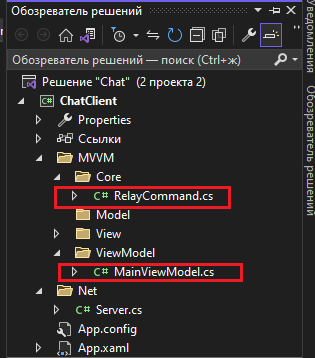


Переходим в директорию «Net» и создаем новый класс «Server». В этом классе создаем поле с типом TcpClient, конструктор, где инициализируем созданное поле и метод ConnectToServer в котором будет происходить соединение (\_tcpClient.Connect("127.0.0.1",7891)).

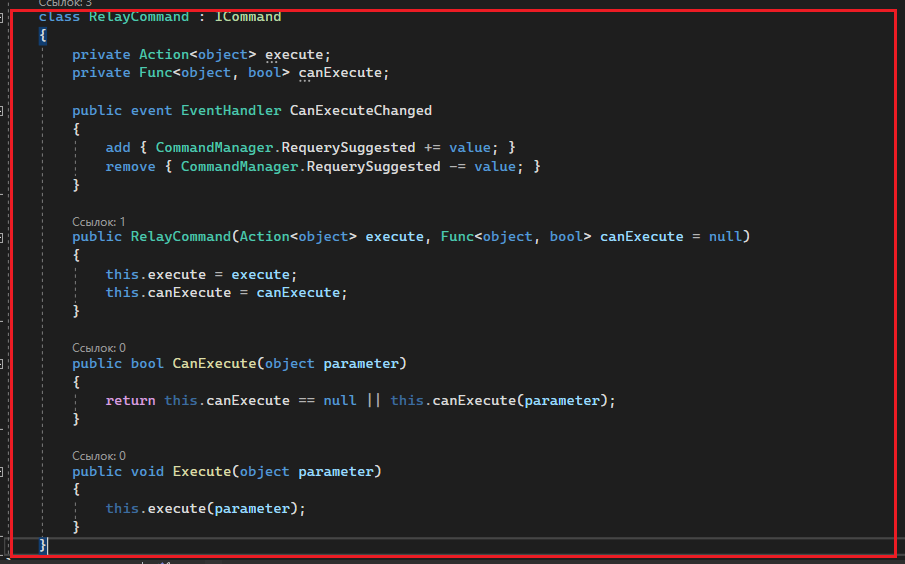




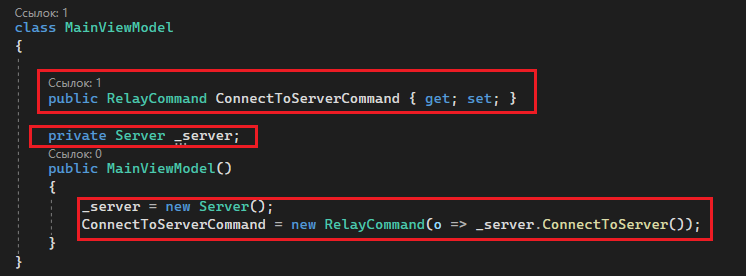
В директории «ViewModel» создаем новый класс «MainViewModel». В директории «Core» создадим класс «RelayCommand» и реализуем интерфейс «ICommand».



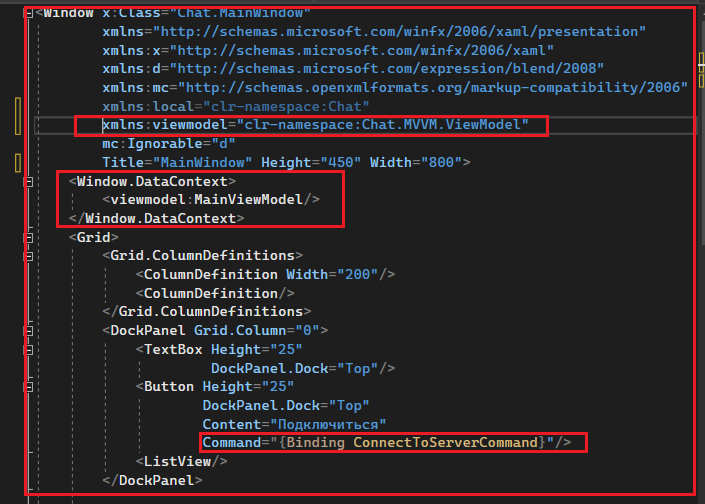
Сперва перейдем к реализации класса «RelayCommand».



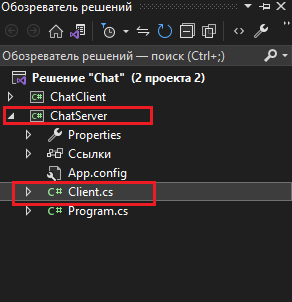
После реализации класса «RelayCommand» переходим к логике класс «MainViewModel». Создадим конструктор класса, свойства «ConnectToServerCommand» и приватную переменную «\_server». В конструкторе произведем их реализацию.

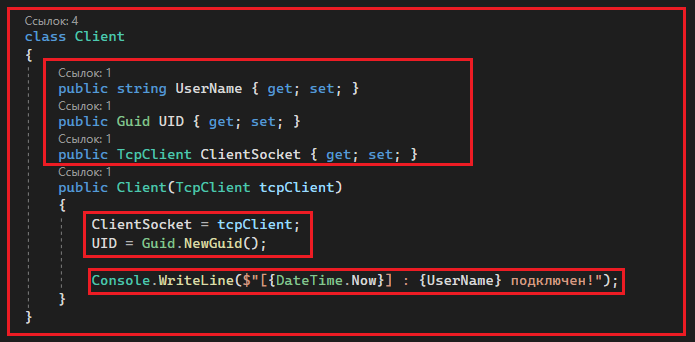


Переходим в «MainWindow.xaml» и в качестве «DataContext» устанавливаем «MainViewModel» (**Необходимо импортировать пространство имен!**). После переходим к кнопке и устанавливаем команду «ConnectToServerCommand» (далее пересоберите и запустите проект).

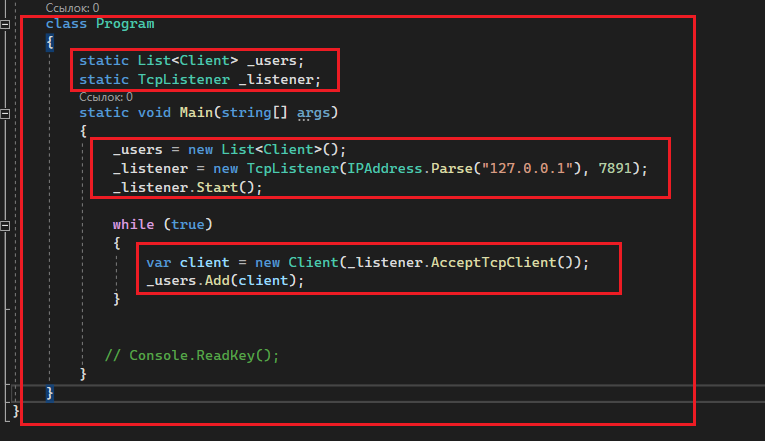


Далее необходимо настроить сервер, для этого переходим в консольное приложение «ChatServer» и создаем класс «Client». В классе создаём свойство «UserName», «UID», «ClientSocket» и конструктор с параметрами. В конструкторе инициализируем «ClientSocket» и «UID» и выводим с помощью «Console.WriteLine» информационную строку.

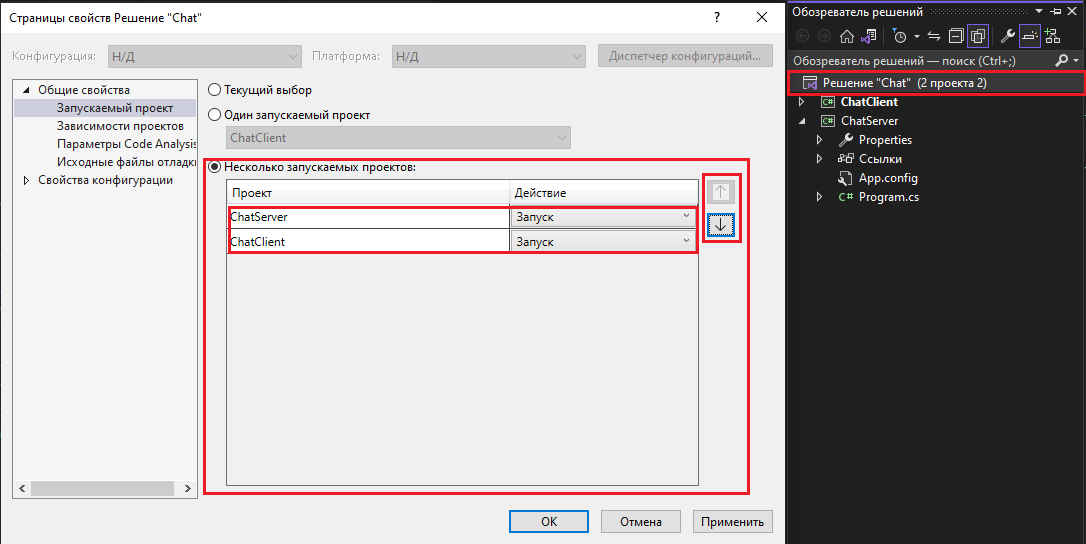


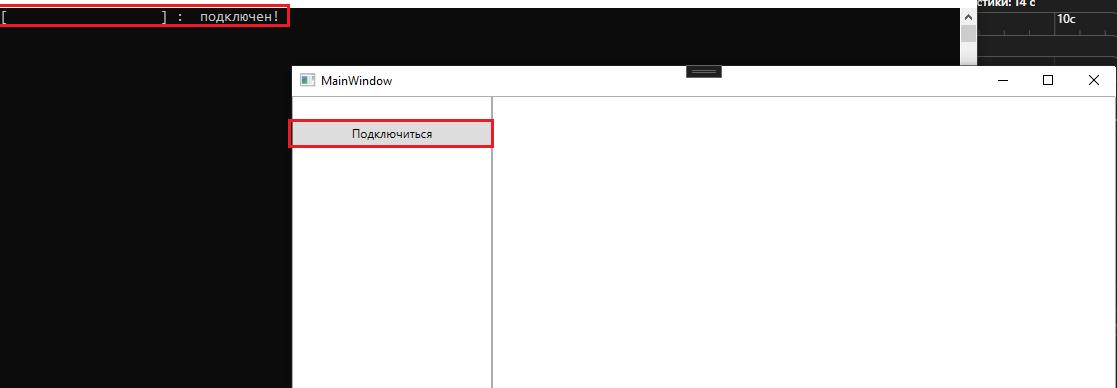


Переходим в класс «Program». Создаем статическую коллекцию «\_users» и типизируем ее классом «Client». После создаем статическую переменно «\_listener» типа «TcpListener». В методе «Main» инициализируем коллекцию, а переменной «\_listener» передаем в качестве параметра IP-адрес и порт("127.0.0.1", 7891,(после необходимо использовать метод «Start»)). Теперь создадим цикл «while» в нем создадим переменную «client», где в качестве параметра конструктора передадим «\_listener.AcceptTcpClient()» (AcceptTcpClient — Это блокирующий метод, возвращающий TcpClient, который можно использовать для отправки и получения данных) и добавляем эту переменную в коллекцию «\_users» с помощью метода «Add».

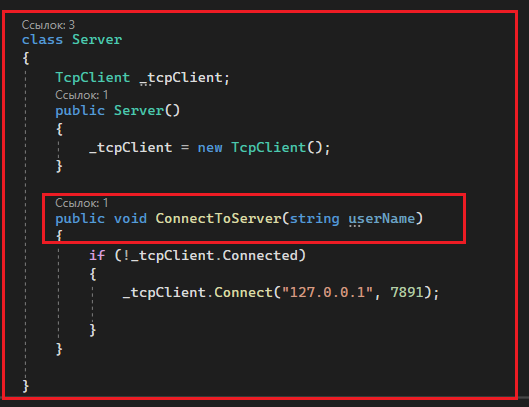


После необходимо перейти в «Решение» и настроить запуск нескольких проектов (далее пересоберите и запустите проект).

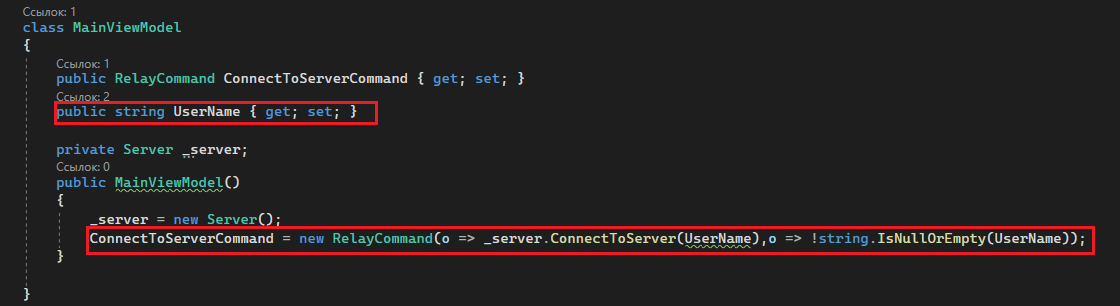


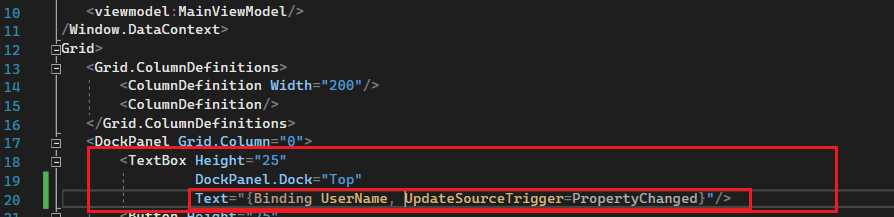


Далее нам нужно выводить имя пользователя, который подключился к серверу. Для это переходим в файл «Server.cs» и в метод «ConnectToServer» передаем параметр тип «string», «userName».

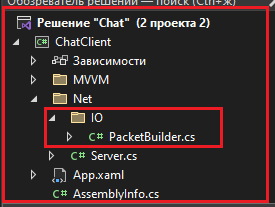


Также не забываем, что данный метод мы вызываем в «MainViewModel» и чтобы небыло ошибки нужно передать дополнительный параметр. Для этого создаём свойство «UserName» и в конструкторе свойтву «ConnectToServerCommand» передаем новый параметр «o => \_server.ConnectToServer(UserName),o => !string.IsNullOrEmpty(UserName)», незабываем что «UserName» мы будем получать с «TextBox» в «MainWindow», а для этого необходимо сделать привязку данных с обновление триггера. Переходим в файл «MainWindow.xaml» и «TextBox» в свойстве «Text» задаем следующий параметр «{Binding UserName, UpdateSourceTrigger=PropertyChanged}"».

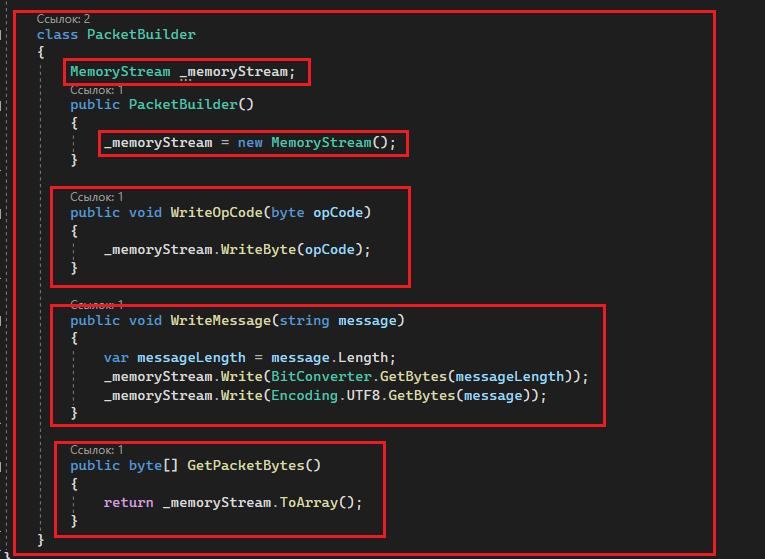




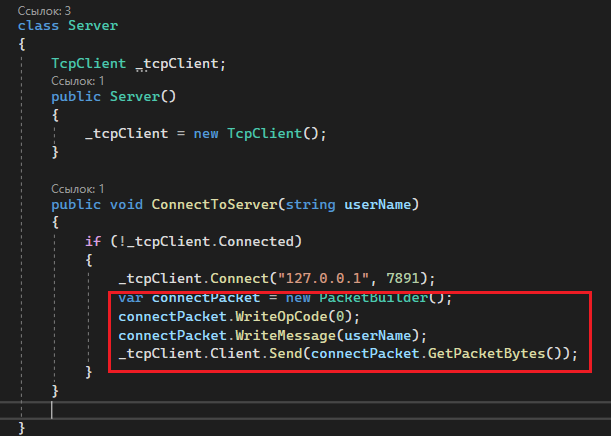
Далее необходимо создать пакет и отправить его на сервер. Для этого в директории «Net» создаём еще одну директории «OI» и в ней создаём класс «PacketBuilder».



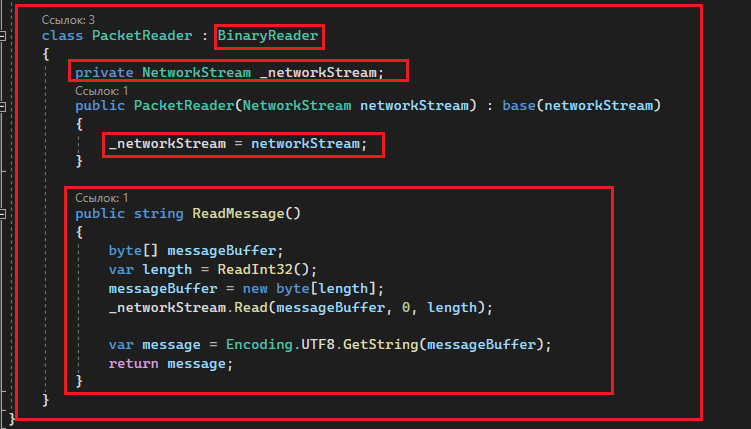
В этом классе необходимо создать поле с типом «MemoryStream» (MemoryStream - создает поток, резервным хранилищем которого является память.) и инициализировать его в конструкторе. Также создадим три метода: «WriteOpCode», «WriteMessage», «GetPacketBytes».



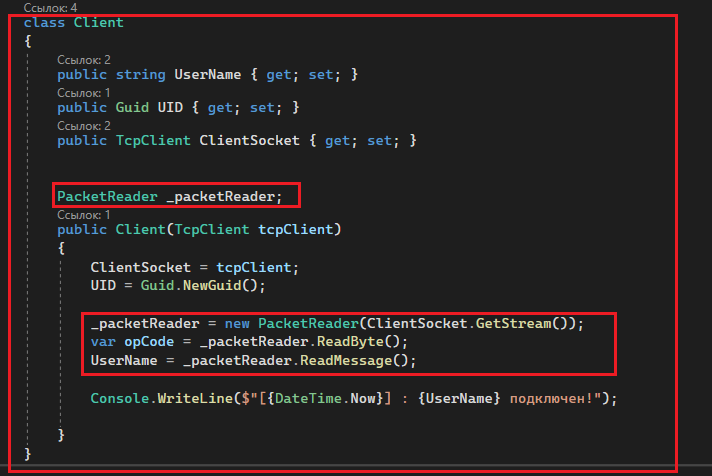
После переходим в файл «Server.cs» и отправляем наш пакет и именем пользователя на сервер.

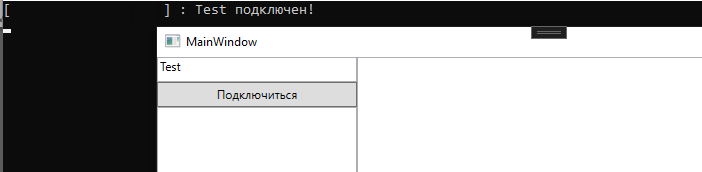


Далее переходим на сервер. Создаем новую директорию «Net» и в ней же еще одну директорию «IO» и в ней создаём класс «PacketReader». Класс PacketReader будет наследоваться от класса «BinaryReader» и будет содержать один метод «ReadMessage».

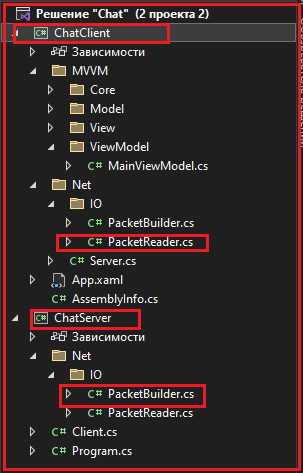


Теперь переходим в файл «Client.cs» создаем поле типа «PacketReader» в конструкторе инициализируем его и присваиваем свойству «UserName» сообщение из потока (далее пересоберите и запустите проект).

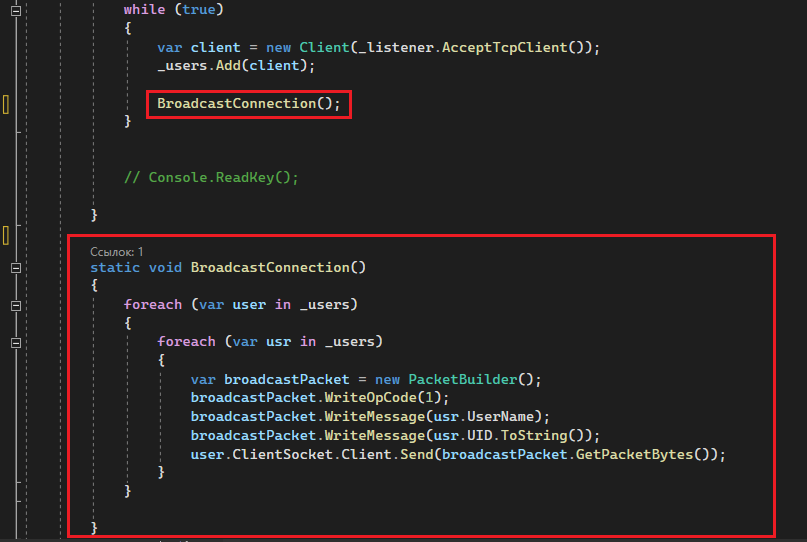




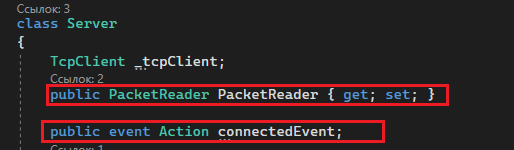
Далее создадим в приложении «ChatClient» в директории IO класс «PacketReader», а в приложении «ChatServer» в директории IO класс «PacketBuilder». Их логика будет идентична тем классам, которые мы создали ранее.

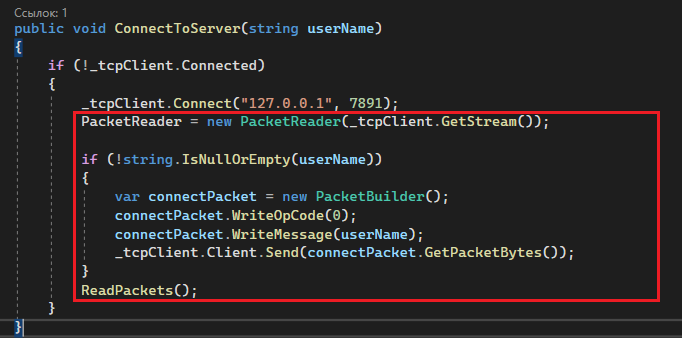


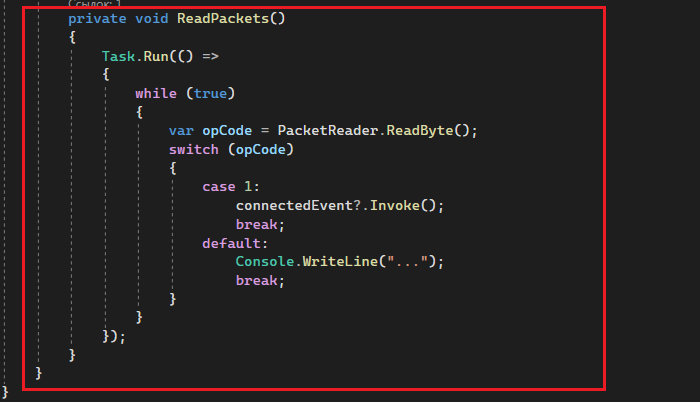
Следующая задача, которую нам необходимо решить — это вывод в «ListView» подключенных пользователей. Для этого переходим в приложение «ChatServer» в класс «Program.cs» и создаем метод «BroadcastConnection» который будет отправлять данные о подключенном пользователе.



Переходим в приложение «ChatClient нужно изменить класс «Server». Добавим два свойства (PacketReader, connectedEvent), один метод и сделаем проверку на «пустоту» поля «userName».

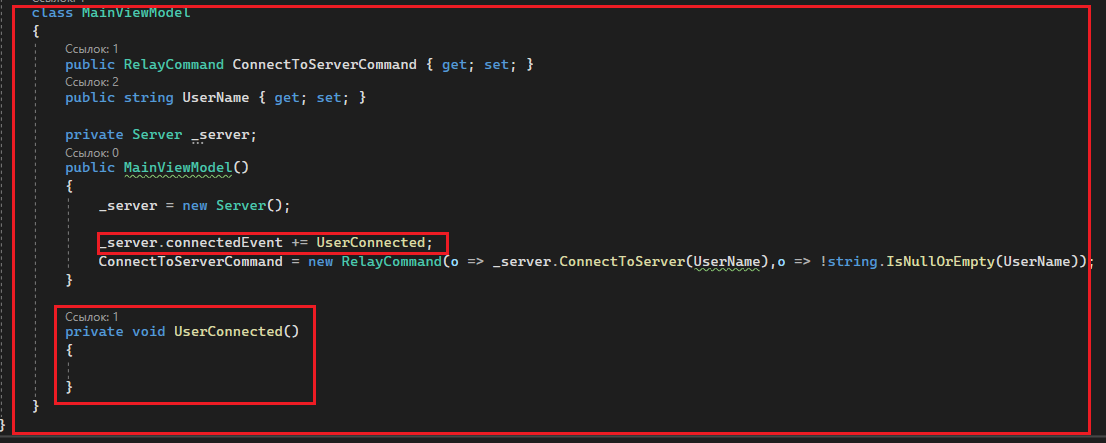




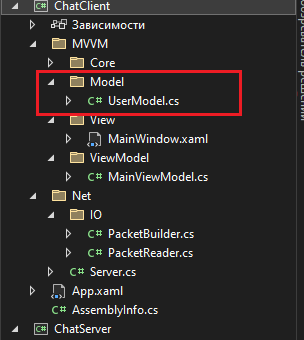


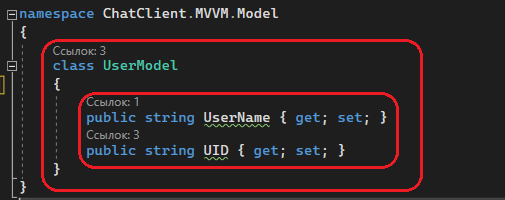
}

Далее переходим в файл «MainViewModel.cs» и в конструкторе делаем заготовку под событие.

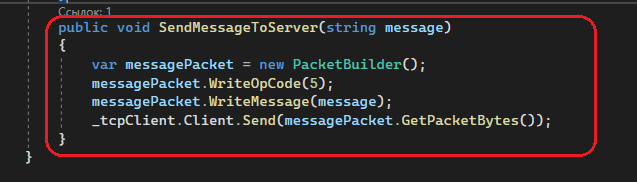


Переходим в директорию «Model» и создаём новый класс «UserModel.cs». В нем создаем два свойства: «UserName» и «UID».

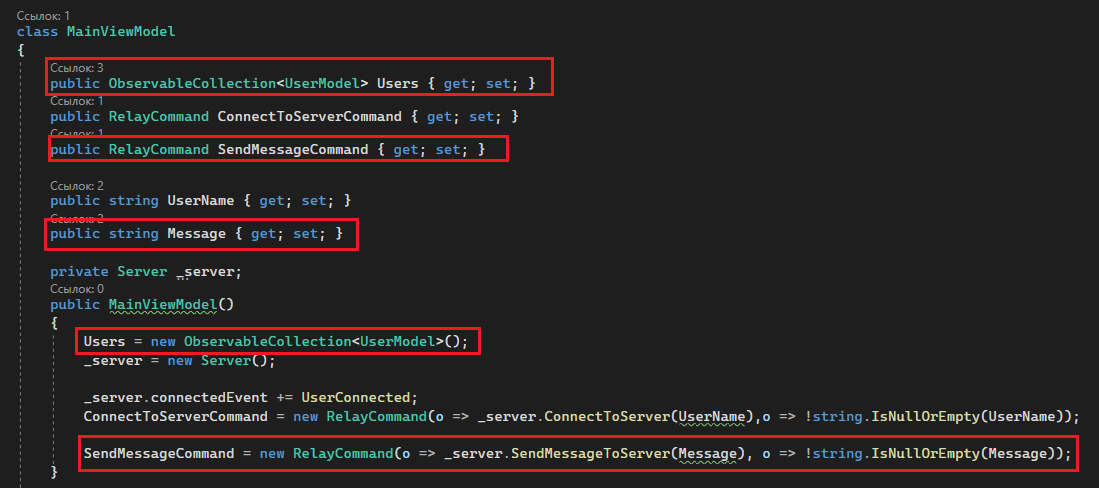


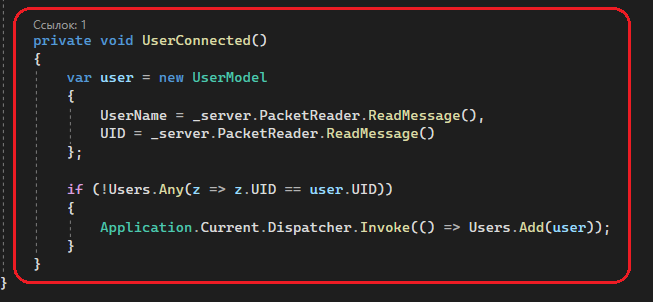


Далее переходим в файл «Server.cs» и в нем создаем метод «SendMessageToServer» который будет отправлять сообщения.

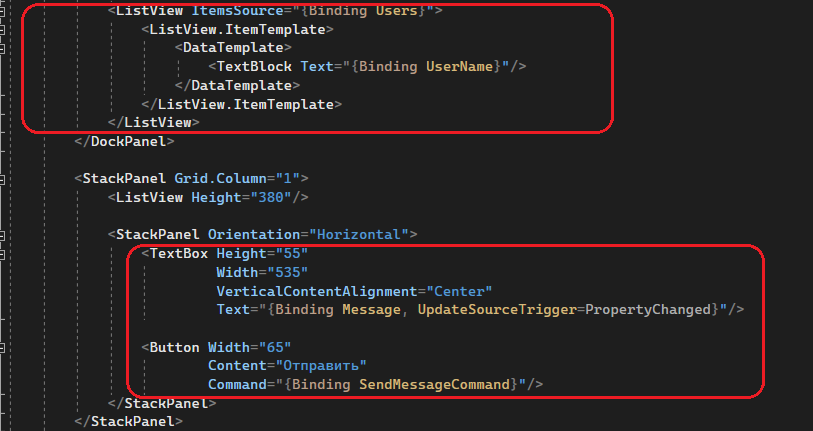


Теперь переходим в файл «MainViewModel.cs». В нем создадим три свойства (коллекцию «Users», «SendMessageCommand», «Message») и метод («UserConnected»).

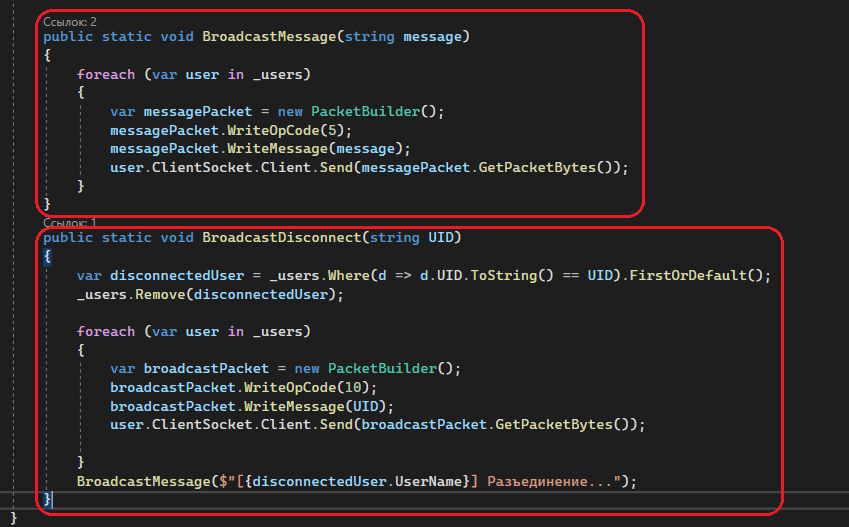




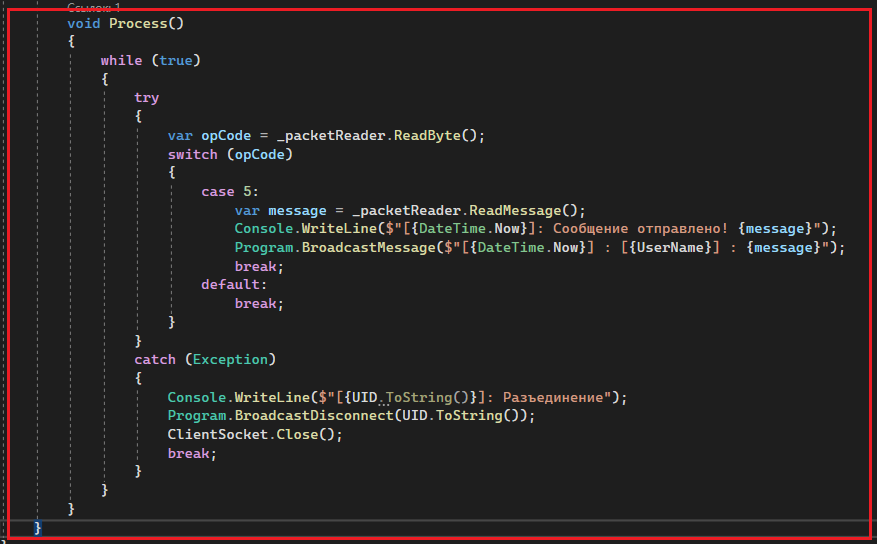
Далее переходим в «MainWindow.xaml» и через «Binding» связываем данные.

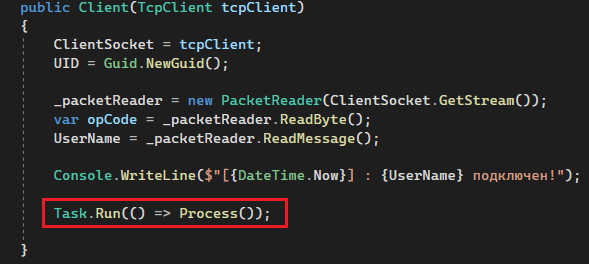


Переходим в приложение «ChatServer» в файл «Program.cs» и создадим два метода «BroadcastMessage» и «BroadcastDisconnect».

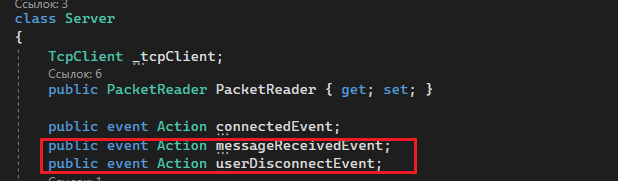


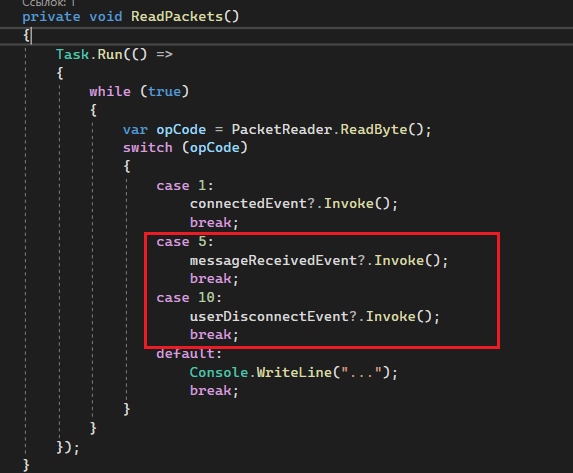
Далее переходим в файл «Client.cs» и создадим метод для отправки сообщения «Process» и вызовем в конструкторе «Client».



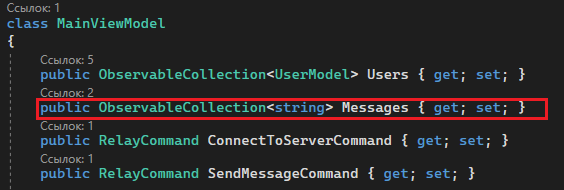


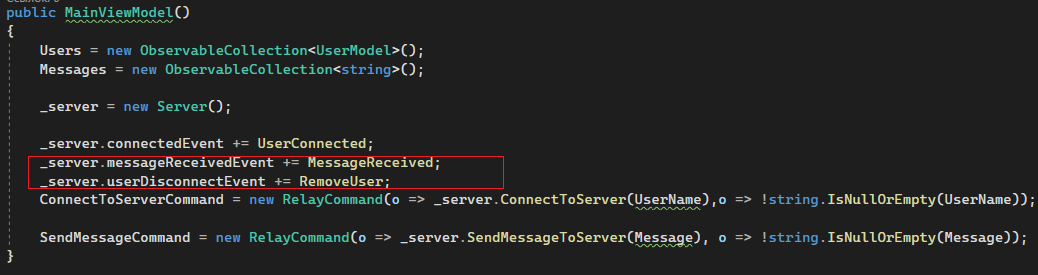
Переходим в приложение «ChatClient» в файл «Server.cs» и создадим два события и допишем метод «ReadPackets».

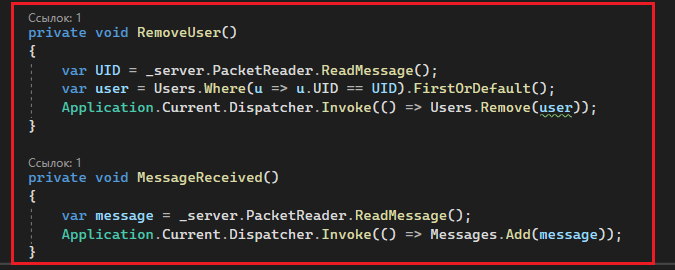




Далее переходим в файл «MainViewModel.cs». В этом файле необходимо создать: коллекцию «Messages», методы «RemoveUser» и «MessageReceived».







После переходим в файл «MainWindow.xaml» и с помощью привязки назначаем в качестве источника данных коллекцию «Messages». После пересоберите и запустите проект



