Министерство Образования Республики Беларусь Белорусский Государственный Университет Информатики и Радиоэлектроники
Кафедра Информатики

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К КУРСОВОМУ ПРОЕКТУ

Дисциплина: Программирование

Тема: Сетевой чат для Windows

Принял:

Козуб Виктор Николаевич

Выполнил: студент гр.653501

Заливако Андрей Андреевич

Минск 2017

Содержание

1. Введение

Конечной целью этого проекта является создание Сетевого Чата для платформы Windows. Простой сетевой чат для различной категории потребителей с достаточной для большинства пользователей функциональностью. В функциональность Сетевого Чата входит: прием хранение и передача текстовой информации и файлов по сети, а также сохранение истории сообщений всех клиентов, подключённых к серверу. Дизайн приложения довольно простой и неброский, но удобен для использования любому типу пользователей.

В ходе работы над этим проектом на практике понадобятся навыки по основам разработки приложений в Windows Forms на языке программирования C#. Будет использована среда разработки Microsoft Visual Studio, так как эта среда используется в данной дисциплине. Возможно, будут выявлены проблемы, характерные для работы над подобными приложениями, тем самым будет воспроизведён поиск необходимой информации для их решения. Это должно упростить более глубокое изучение данной системы в будущем.

2. Сравнение с программой обмена сообщениями ICQ

2.1. Функциональность

Сетевой чат значительно уступает продукту ICQ в функциональности. Так как Сетевой чат был разработан в очень короткие сроки, в нём содержится только базовый функционал по передаче, приёму и отправке сообщений и файлов, без различных расширений, которыми можно воспользоваться на ICQ.

2.2. Интерфейс

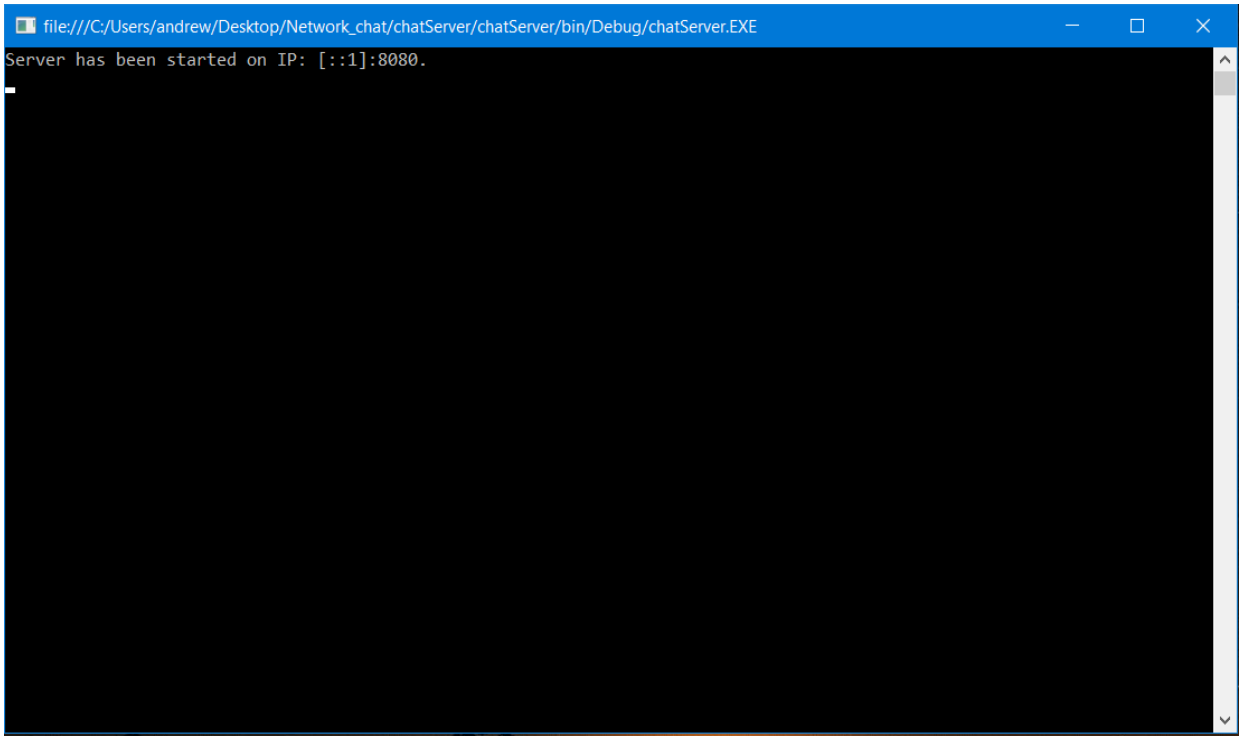
Обе программы обладают достаточно дружелюбным интерфейсом. Однако стоит отметить, что в связи с повышенной функциональностью аналог ICQ вызывает чувство некоторой нагромождённости, чего нельзя сказать о Сетевом Чате. Также стоит отметить то, что интерфейс Сетевого Чата очень прост, что позволяет пользователю без затруднений разобраться с функциональностью программы. Что касается языков интерфейса, для Сетевого Чата разработан только русский интерфейс, в то время, как на ICQ имеются и другие языки.

2.3. История

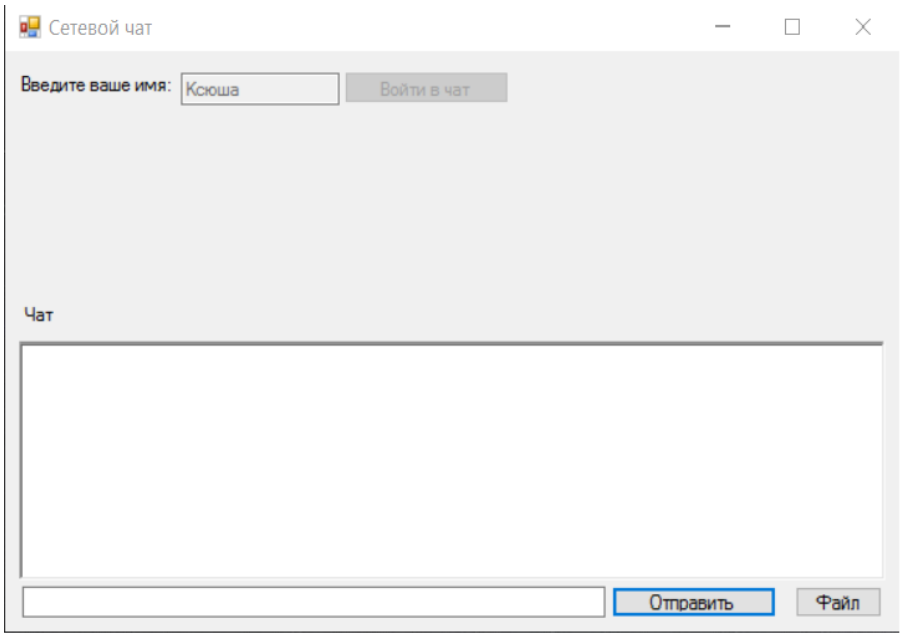
15 ноября 1996 года Арик Варди, Яир Голдфингер, Сефи Вигисер и Амнон Амир, старшеклассники из Тель-Авива (Израиль), основали компанию Mirabilis и создали интернет-пейджер ICQ. Программное обеспечение изначально распространялось бесплатно (в отличие от конкурентов). Число пользователей росло лавинообразно. Mirabilis предлагала IM не только частным пользователям, но и корпоративным клиентам. В 1998 году компания была выкуплена американской корпорацией AOL за 407 млн долларов и была преобразована в часть отделения Time Warner — ICQ, Incorporated.

Что же касается Сетевого Чата, на его разработку было потрачено около трёх недель. В случае необходимости работа над ним может быть продолжена, так как приложение довольно легко поддается расширению за счет небольшого количества составляющих.

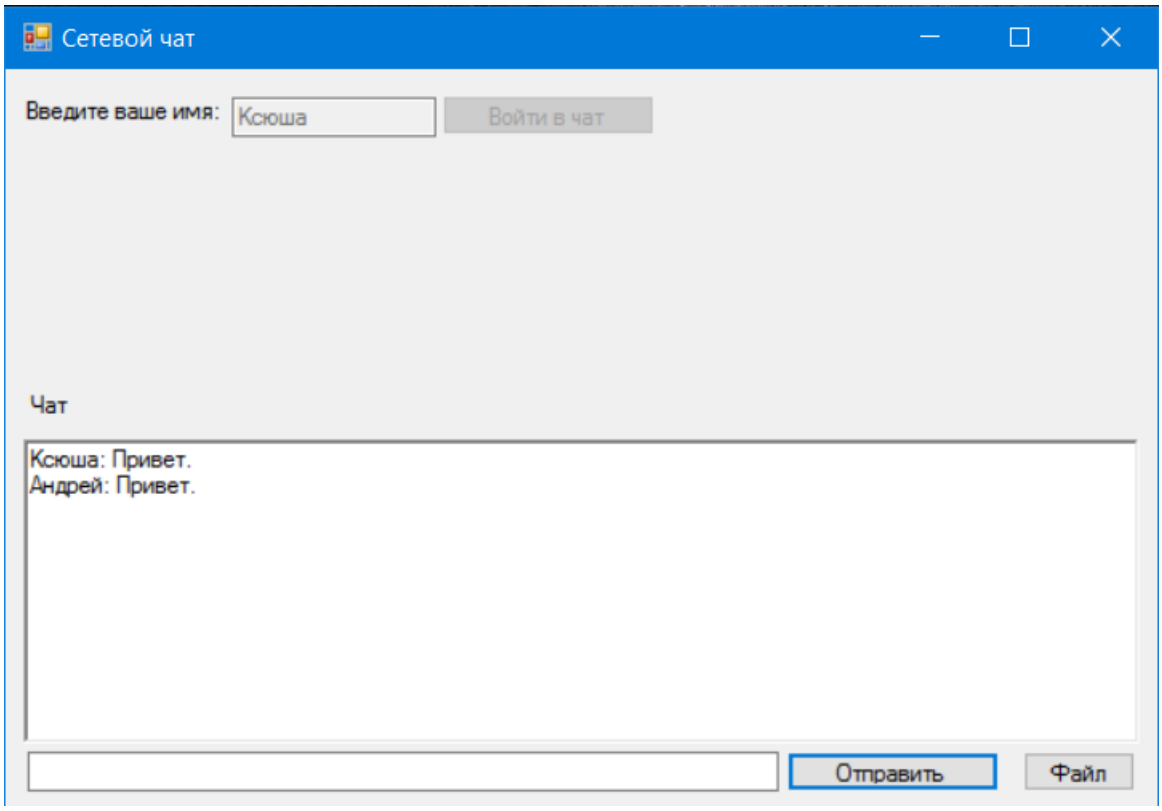
3. Скриншоты и описание



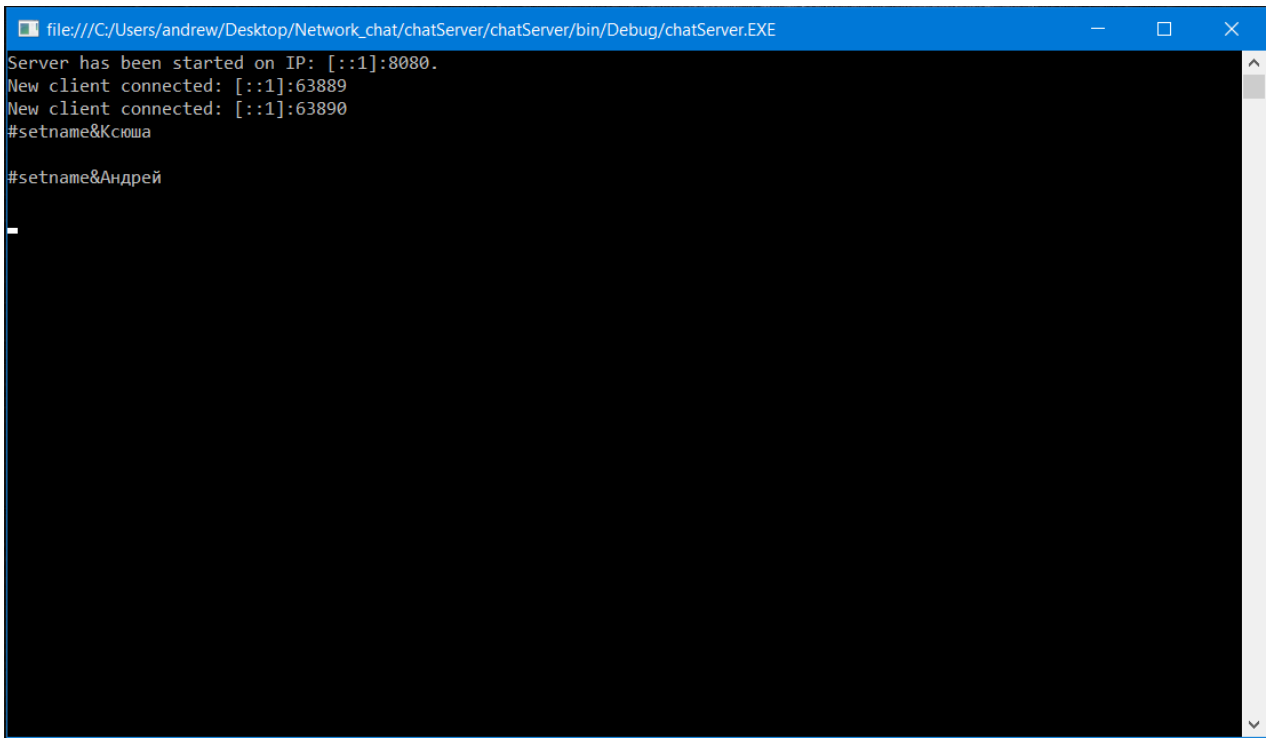
Сервер Сетевого Чата открывается в консольном приложении.



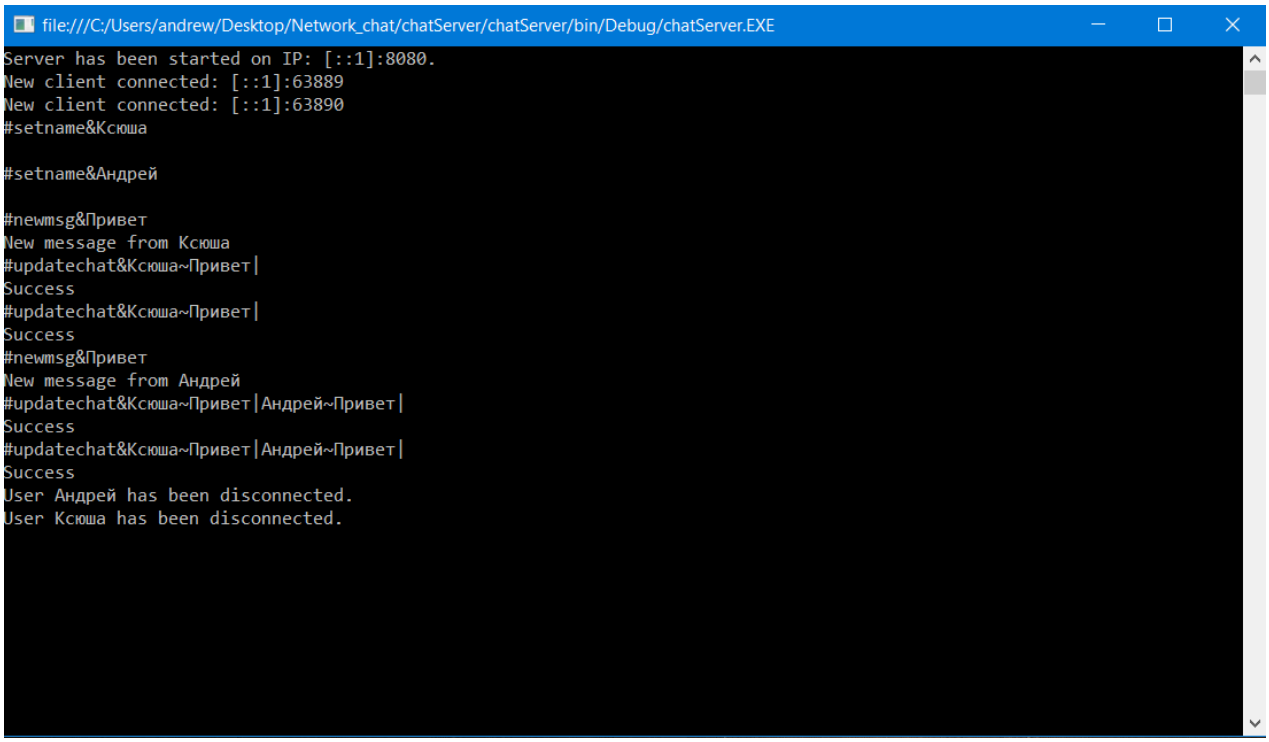
Интерфейс программы довольно простой и интуитивно понятный. Чтобы войти в чат, Вам необходимо ввести имя пользователя в окошко рядом с надписью: «Введите ваше имя», иначе вход не будет выполнен. Кнопка «Войти в чат» добавляет нового пользователя. Чтобы отправить сообщение, Вам потребуется ввести текст в поле рядом с кнопкой «Отправить» и нажать на эту кнопку. Чтобы отправить файл, Вам потребуется нажать на кнопку «Файл» и выбрать файл, который Вы хотите отправить.



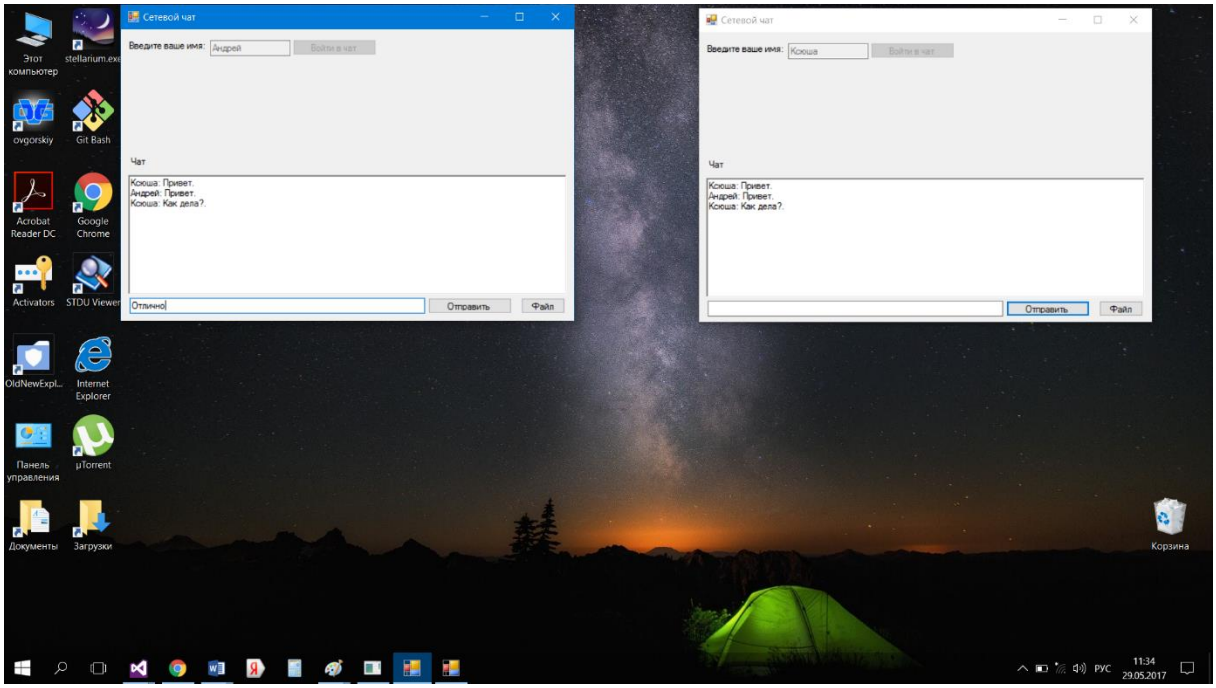
Два зарегистрированных пользователя пишут сообщения и отправляют сообщения, которые сохраняются в историю и которые мы можем увидеть на экране.



В консольном приложении на сервере показываются все действия, связанные с программой-клиентом. То есть вход и регистрация пользователей, отправка сообщений и файлов, выход пользователей из программы и все возможные ошибки с связью сервера и клиентов.



Для выхода из программы Вам следует нажать на  в правом верхнем углу окошка.



4. Техническая информация

4.1. Средства разработки

Для разработки приложения использовалась система для построения клиентских приложений Windows с использованием Windows Forms.

Разработка велась в среде Visual Studio 2015, так как данная среда является новейшим полноценным средством работы с платформой .NET и поставляется бесплатно в редакции Community.

Socket Класс предоставляет широкий набор методов и свойств для сетевых взаимодействий. Socket Класс позволяет выполнять синхронный и асинхронную передачу данных с использованием любого из коммуникационных протоколов, перечисленных в ProtocolType перечисления.

4.2. Структура программы

Файлы сервера (проект chatServer):

Program.cs – тут содержится код программы.

ChatController.cs – класс с кодом на C#, который отвечает за управление элементами чата.

Client.cs – класс с кодом на C#, позволяющий осуществляет работу с клиентами, подключенными к серверу.

Server.cs – класс с кодом на C#, который содержит инструменты для добавления и удаления подключенных клиентов.

Файлы программы-клиента:

Program.cs – тут содержится код программы.

MainClient.cs – класс с кодом на C#, который содержит основную логику работы клиентского приложения (отправка, получение сообщений, обновление чата, подключение к серверу).

5. Описание работы программного кода

В начале требуется произвести запуск сервера.

Инициализация Socket компонента в котором передаются данные такие как IP адрес, порт, по которому будет происходить обмен данными.

```
public class Client
  {
      private string _userName;
      private Socket _handler;
      private Thread _userThread;
      XmlDocument history = new XmlDocument();

      public Client(Socket socket)
      {
          _handler = socket;
          _userThread = new Thread(listner);
          _userThread.IsBackground = true;
          _userThread.Start();

       }
```

После этого создается отдельный поток, в котором обрабатываются события подключение клиента к серверу, регистрация его имени, получение и отправка сообщений.

```
public static void NewClient(Socket handle)
      {
          try
          {
              Client newClient = new Client(handle);
              Clients.Add(newClient);
              Console.WriteLine("New client connected: {0}", handle.RemoteEndPoint);
          }
          catch (Exception exp) { Console.WriteLine("Error with addNewClient: {0}.",
     exp.Message); }
      }
```

В то же время в этом же потоке идет учет сообщений, которые прошли через сервер.

```
private const int _maxMessage = 100;
 public static List<message> Chat = new List<message>();

      public struct message
      {
          public string userName;
          public string data;

          public message(string name, string msg)
          {
              userName = name;
              data = msg;
          }
      }
```

```
        public static void AddMessage(string userName, string msg)
        {
            try
            {
                if (string.IsNullOrEmpty(userName) || string.IsNullOrEmpty(msg)) return;
                int countMessages = Chat.Count;
                if (countMessages > _maxMessage) ClearChat();
                message newMessage = new message(userName, msg);
                Chat.Add(newMessage);
                Console.WriteLine("New message from {0}", userName);
                Server.UpdateAllChats();
            }
            catch (Exception exp) { Console.WriteLine("Error with addMessage: {0}.",
exp.Message); }
          }
```

Далее реализовывается клиентская часть.
При запуске клиент-приложения сразу адресуется форма, после чего в Socket передаются параметры для подключения IP адреса сервера и порта подключения.

```
    private delegate void printer(string data);
    private delegate void cleaner();
    printer Printer;
    cleaner Cleaner;
    private Socket _serverSocket;
    private Thread _clientThread;
    private const string _serverHost = "localhost";
    private const int _serverPort = 8080;

     public MainClient()
     {
         InitializeComponent();
         Printer = new printer(print);
         Cleaner = new cleaner(clearChat);
         connect();
         _clientThread = new Thread(listner);
         _clientThread.IsBackground = true;
         _clientThread.Start();
     }

     private void listner()
     {
         while(_serverSocket.Connected)
         {
             byte[] buffer = new byte[1024];
             int bytesRec = _serverSocket.Receive(buffer);
             string data = Encoding.UTF8.GetString(buffer, 0, bytesRec);
             if (data.Contains("#updatechat"))
             {
                 UpdateChat(data);
                 continue;
             }

         }
     }
```

При нажатии кнопки “Войти в чат” происходит отправка укомплектованного пакета с командой setname. Которая дает серверу понять, что данному клиенту присвоено имя.

```
private void enter_chat()
      {
          string Name = userName.Text;
          if (string.IsNullOrEmpty(Name)) return;
          send("#setname&" + Name);
          chatBox.Enabled = true;
          chat_msg.Enabled = true;
          chat_send.Enabled = true;
          userName.Enabled = false;
          enter.Enabled = false;
        }
```

После отправки сообщения на сервер поступает строка из byte которая переводится в string и разбивается на компоненты, такие как действия и сообщения. В то время, когда сервер получает одну из команд он возвращает updatechat который запускает процедуру обновления чата на стороне клиента.

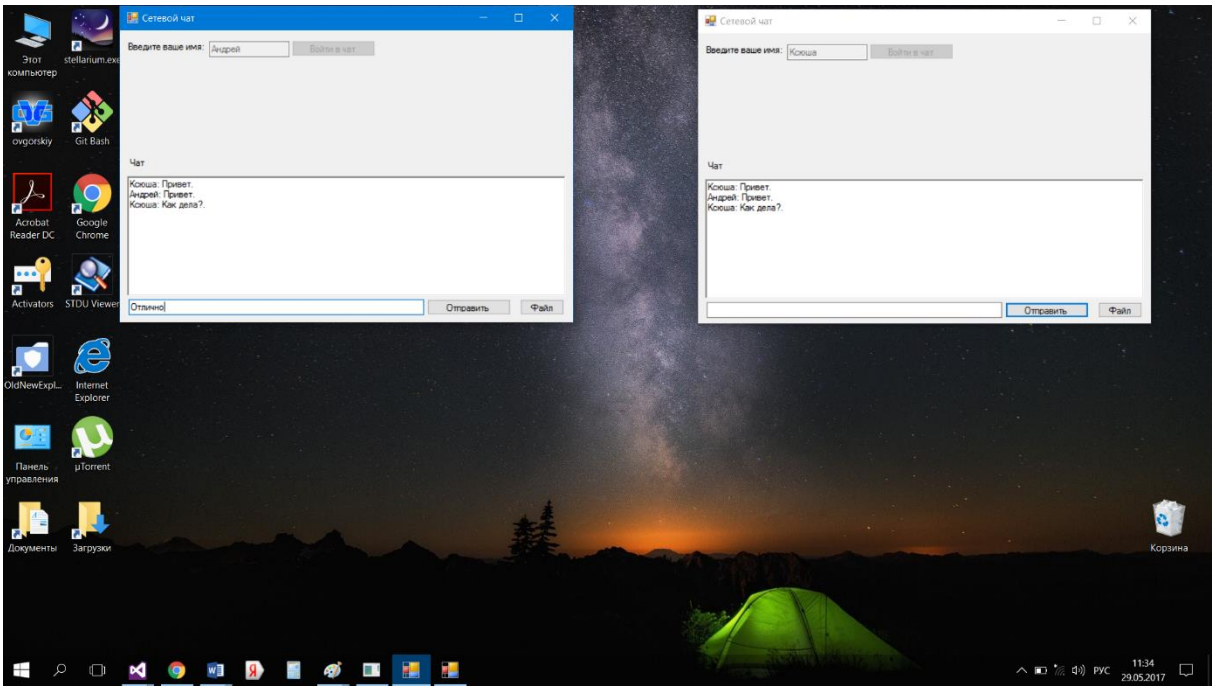
```
     private void UpdateChat(string data)
      {
          //#updatechat&userName~data|username~data
        clearChat();
        string[] messages = data.Split('&')[1].Split('|');
        File.WriteAllBytes("newfile.txt", Encoding.Default.GetBytes(data.Split('&')[1]));
        int countMessages = messages.Length;
        if (countMessages <= 0) return;
        for(int i=0; i<countMessages; i++)
        {
              try
              {
                  if (string.IsNullOrEmpty(messages[i])) continue;
                  print(String.Format("{0}: {1}.", messages[i].Split('~')[0],
messages[i].Split('~')[1]));
              }
              catch { continue; }
          }
        }
```

Пока работает сервер и клиент, действия будут повторяться. Два независимых приложения работают с интерфейсом, позволяющим им обмениваться между собой сообщениями в виде потока byte, тем самым позволяет синхронно выполнять поставленную задачу в виде обмена данными между пользователями.

6. Тестирование

Работа программы была проверена на операционной системе Windows 10. Как показали тесты, текст остается читаемым и графическое оформление вписывается в любую цветовую схему, предложенную Microsoft.

Также работа программы была проверена на всех возможных разрешениях экрана, что также подтвердило ее стабильность. Во время тестирования каких-либо недостатков в работе, к примеру, ошибок в ходе программы не было замечено.



7. Выводы

В ходе работы над проектом были изучены и закреплены на практике основные принципы работы в Windows Forms. Был выявлен ряд проблем, часть из которых были решены во время работы над приложением. Среди этих проблем: связь сервера с программой-клиентом, преобразование типов данных при передаче информации, разделение вводимой информации методом Split и некоторые другие.

При этом также были определены проблемы, требующие дальнейшего изучения. Например, сохранение чистоты и, соответственно, обслуживаемости кода. Также были выявлены некоторые проблемы с отправкой файлов, поэтому требуется небольшая доработка до полной эффективности этой функции. В данном проекте это не совсем получилось, из-за отсутствия какого-либо опыта работы в Windows Forms в прошлом, недостаточного времени, отведенного на проектирование программы и неполное знание средств выбранной системы. Также стала очевидной необходимость придерживаться заранее определенного стиля написания кода для облегчения его читаемости.

Несмотря на некоторые недостатки в программном коде, поставленные цели были достигнуты. Программа имеет ненавязчивый дизайн и определенный изначально функционал, работает стабильно, не зависимо от программного и аппаратного обеспечения. Недостатки, выявленные при написании кода, определяют направление изучения мною не только данной системы, но и принципов проектирования ПО в целом, поэтому проект считаю полезным и успешно завершенным.

Список литературы

1. https://msdn.microsoft.com/ru-ru/library/system.net.sockets.socket.
2. https://habrahabr.ru/post/228021/
3. http://www.cyberforum.ru/csharp-net/thread380300.html
4. https://metanit.com/sharp/net/4.4.php
5. https://professorweb.ru/my/csharp/web/level4/4_7.php