

293

TGPSI

**Projecto Tecnológico** ***leChat***

Relatório

Nº 103629 – David Samuel Pereira Silva

Nº 104289 – Diogo Alexandre Flores Simões de Jesus



Identificação dos Alunos

David Samuel Pereira Silva

Idade: 18

Turma: 293

Correio electrónico: zgrav@null.net

Número de contacto: +351918586295

Diogo Alexandre Jesus

Idade: 16

Turma: 293

Correio electrónico: evilmonstah@innocent.com

Número de contacto: +351915345721

Índice

Conteúdo

[Introdução 4](#_Toc329069963)

[Objectivos do Projecto 5](#_Toc329069964)

[Proposta do Projecto 6](#_Toc329069965)

[Funcionalidades 7](#_Toc329069966)

[Multithreading 7](#_Toc329069967)

[Sockets 7](#_Toc329069968)

[Servidor 9](#_Toc329069969)

[Cliente 15](#_Toc329069970)

[*Stresstester* 18](#_Toc329069971)

[Historial do Programa 20](#_Toc329069972)

[Tecnologias usadas 23](#_Toc329069973)

[Conclusão 24](#_Toc329069974)

[Agradecimentos 25](#_Toc329069975)

# Introdução

No âmbito do 2º ano do curso de TGPSI (Técnico de Gestão e Programação de Sistemas Informáticos), foi-nos pedido para desenvolver um Projecto Tecnológico e proceder à apresentação do mesmo no final do respectivo ano lectivo.

Este projecto é constituído por duas aplicações baseadas em alguma matéria adquirida ao longo do curso e também por exploração própria. Tem como objectivo, demonstrar essas competências e expor o desenvolvimento de um projecto um pouco mais complexo utilizando *Threads* e *Sockets*.

O tempo dado para a realização e documentação do projecto foi sensivelmente a volta de 2 meses. Tendo começado com as propostas a 18 de Abril de 2012, a data de entrega do projecto foi definida para o dia 26 de Junho de 2012.

O tema e funcionalidades do projecto são deixados a cargo dos alunos. Optamos pela criação de uma espécie de Chat por LAN.

Ao longo desta Introdução, iremos abordar os seguintes aspectos:

* Objectivos do projecto;
* Proposta do projecto;
* Tecnologias e recursos utilizados;
* Contributos do projecto;
* Organização deste relatório;

# Objectivos do Projecto

O tema abordado (Chat via LAN) apresenta inúmeras características que podem ser bastante desenvolvidas se abordadas devidamente, se não poderão surgir várias complicações e pode não ser tao amigável ao utilizador.

Apos termos visto alguns programas e as suas funcionalidades, foi notado que alguns não continham algumas funcionalidades cruciais para estarem desenvolvidos a 100% e foi isso que tentamos fazer com o nosso projecto.

Os principais objectivos deste projecto são:

* Servidor & Cliente
* Uma lista de utilizadores
* Sistema de mensagens e a sua data/hora de recepção
* Histórico
* Notificações
* Transferência de ficheiros

Implementações futuras:

* Imagem de apresentação para cada utilizador
* *Emoticons*
* Formatação de fontes
* Temas

# Proposta do Projecto

Na proposta foram definidos os objectivos principais (pág. 4) e tendo em conta os objectivos, foram definidos as datas de início e fim de cada fase de desenvolvimento e o cronograma.

A proposta foi entregue a 8 de Maio de 2012.

|  |  |  |  |
| --- | --- | --- | --- |
| Fase | Data de início | Data de Fim | Horas/Dias |
| Planeamento | 18-04-2012 | 20-04-2012 | 48/2 |
| Proposta | 08-05-2012 | 08-05-2012 | 24/1 |
| Análise | 19-04-2012 | 20-04-2012 | 48/2 |
| Implementação | 19-04-2012 | 25-06-2012 | 1632/68 |
| Documentação | 25-06-2012 | 25-06-2012 | 24/1 |
| Apresentação | ? | ? | ? |

Tabela 1 - Datas relativamente às fases de desenvolvimento

A *Tabela 1* mostra as previsões para as datas de início e fim de cada fase de desenvolvimento e o seu número de horas.

Embora a tabela seja útil, é necessário um cronograma detalhado para que visualmente se possa interpretar o tempo definido para cada fase.

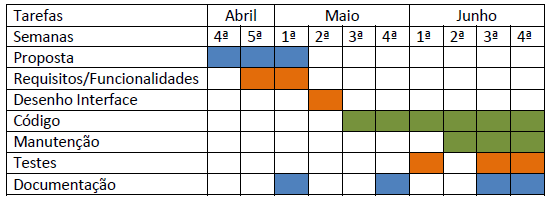


Tabela 2 - Cronograma

Legenda:

Azul – Documentação

Laranja – Análise

Verde – Implementação

# Funcionalidades

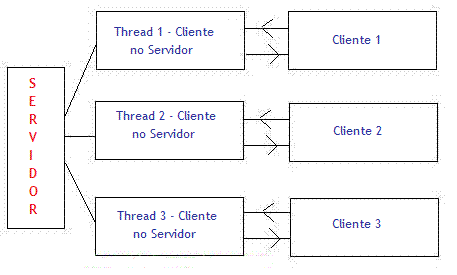
Permite a troca de mensagens e/ou ficheiros entre os utilizadores ligados ao Servidor, este também facilita a comunicação entre colegas de trabalhos que estejam espalhados pelo edifício e precisem de comunicar entre si, no entanto o programa também poderá ser usado num espaço de convivência.

## Multithreading

A programação com Socket multithread significa que um servidor multithread pode comunicar com mais de um cliente ao mesmo tempo usando a comunicação via socket.

A ideia básica por detrás da programação multithread com sockets é que sempre que servidor receber um pedido de ligação de cliente, o servidor criará uma Thread (independente) em separado para a solicitação de cada cliente.

Isso significa que para cada cliente, existe uma thread do cliente em separado. Assim, o cliente pode se comunicar de forma independente com sua própria Thread cliente no servidor.

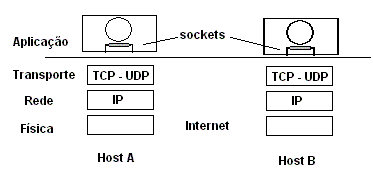
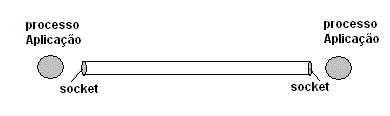


## Sockets

Um socket pode ser entendido como uma porta de um canal de comunicação que permite a um processo executando em um computador enviar/receber mensagens para/de outro processo que pode estar sendo executado no mesmo computador ou num computador remoto.

Os sockets permitem então a comunicação processo a processo da seguinte forma:

* Comunicação local : processos locais usando sockets locais
* Comunicação remota : processos remotos usando sockets em rede (TCP/IP)



Código

## Servidor

O servidor é o que aceita as ligações dos clientes e trata da informação e mensagens recebidas dos mesmos, iremos mostrar aqui as três funções mais importantes do Servidor.

Função Main **–** Esta função define a porta que fica à “escuta” de clientes e inicia o Servidor nessa porta. Recebe as ligações e redirecciona o cliente para a *hashtable* onde é atribuído uma posição ao cliente e guarda toda a informação correspondente ao mesmo. Inicia também a função dos comandos de Consola e o temporizador que actualiza a lista de clientes automaticamente. Também trata de reiniciar o Servidor automaticamente após três segundos se um valor no campo da porta não for adequado.

|  |
| --- |
| #Region "TCP/IP protocol and Client handling"  Sub Main()  Console.Title = "leChat Server"  Dim port As String  Console.WriteLine("Please insert a port value.")  Console.WriteLine("If no port value is entered, the default will be 8888.")  port = ReadLine()  If String.IsNullOrEmpty(port) Then  port = "8888"  End If  If IsNumeric(port) = True Then  '---timer---  listtimer.AutoReset = True  listtimer.Interval = 60000  AddHandler listtimer.Elapsed, AddressOf tick  listtimer.Enabled = True  listtimer.Start()  '---timer end---  Try  Dim server As New TcpListener(port)  Dim client As TcpClient  Dim IP = GetIPv4Address()  Console.Clear()  Dim conShellThread As Threading.Thread = New Threading.Thread(AddressOf conShell)  server.Start()  message("Server up @ " + IP)  message("Port: " + port)  conShellThread.Start()  count = 0  While (True)  count += 1  client = server.AcceptTcpClient()  Dim bytes(10024) As Byte  Dim netwrkStream As NetworkStream = client.GetStream() ' Network and Client data Stream  netwrkStream.Read(bytes, 0, CInt(client.ReceiveBufferSize)) ' Reads data received from client.  dataClnt = System.Text.Encoding.ASCII.GetString(bytes)  If dataClnt.IndexOf("$") < 0 Then  clntDisc = True  Exit Sub  Else  dataClnt = dataClnt.Substring(0, dataClnt.IndexOf("$"))  Dim orgnick As String  Dim countuser As Integer  Dim orgnickbackup As String  orgnick = dataClnt  countuser = countuser + 1  orgnickbackup = orgnick  If countuser > 1 AndAlso orgnick = dataClnt Then  If clList.Contains(orgnickbackup) Then  dataClnt = dataClnt & "\_" & countuser  orgnick = ""  message("Multiple nickname found: " & orgnickbackup)  clList(dataClnt) = client  bcast(dataClnt + " " + "(" + Convert.ToString(count) + ")" + " " + "joined " + DateAndTime.Now, dataClnt, False)  message(dataClnt + " " + "(" + Convert.ToString(count) + ")" + " " + "joined the lobby. " + DateTime.Now)  bcast("This nickname is in use by another client.", Nothing, False)  Dim clienth2 As New handleClnt  clienth2.startClient(client, dataClnt, clList)  countuser = 0  Continue While  End If  End If  If userlist.Contains(orgnickbackup) Then  For i = 0 To userlist.Count - 1  If userlist.Item(i).ToString.Contains(orgnickbackup) Then  countuser = countuser + 1  End If  Next  dataClnt = dataClnt & "\_" & countuser  orgnick = ""  message("Multiple nickname found: " & orgnickbackup)  clList(dataClnt) = client  bcast(dataClnt + " " + "(" + Convert.ToString(count) + ")" + " " + "joined " + DateAndTime.Now, dataClnt, False)  message(dataClnt + " " + "(" + Convert.ToString(count) + ")" + " " + "joined the lobby. " + DateTime.Now)  bcast("This nickname is in use by another client.", Nothing, False)  Dim clienth3 As New handleClnt  clienth3.startClient(client, dataClnt, clList)  countuser = 0  Continue While  Else  clList(dataClnt) = client  bcast(dataClnt + " " + "(" + Convert.ToString(count) + ")" + " " + "joined " + DateAndTime.Now, dataClnt, False)  message(dataClnt + " " + "(" + Convert.ToString(count) + ")" + " " + "joined the lobby. " + DateTime.Now)  Dim clienth As New handleClnt  clienth.startClient(client, dataClnt, clList) ' Client handler  End If  End If  End While  Catch ex As IOException  ' message("Socket not disposed properly.")  clntDisc = True  End Try  Else  Console.WriteLine()  Console.WriteLine("An incorrect port value has been set. Program restarting in 3 seconds.")  If Directory.Exists(Application.StartupPath & "\Log") = False Then  Call Directory.CreateDirectory(Application.StartupPath & "\Log")  End If  My.Computer.FileSystem.WriteAllText("Log\Server Log " + FormatDateTime(DateAndTime.Today, DateFormat.LongDate) + ".log", DateTime.UtcNow + " >> " + "Server restarted!" + Environment.NewLine, True)  Thread.Sleep(3000)  Application.Restart()  End If  End Sub  #End Region |

Função de Broadcast – Esta função trata de receber as mensagens do cliente e de as difundir por todos os clientes conectados.

|  |
| --- |
| #Region "Broadcasting"  Private Sub bcast(ByVal msg As String, ByVal userName As String, ByVal msgflag As Boolean)  Dim Item As DictionaryEntry  Try  For Each Item In clList  Dim bcastSocket As TcpClient 'Broadcast Socket  bcastSocket = CType(Item.Value, TcpClient)  Dim bcastStream As NetworkStream = bcastSocket.GetStream()  Dim bcastBytes As [Byte]() 'Broadcast Length received on bcastStream.Write  If msgflag = True Then  bcastBytes = Encoding.ASCII.GetBytes(DateTime.Now + " " + userName + " says : " + msg)  msg2 = msg  Else  bcastBytes = Encoding.ASCII.GetBytes(msg)  End If  bcastStream.Write(bcastBytes, 0, bcastBytes.Length)  bcastStream.Flush()  Next  Catch ex As IOException  ' message("Socket not disposed properly.")  clntDisc = True  Catch ex2 As ObjectDisposedException  ' message("Object not disposed properly.")  clntDisc = True  Catch ex3 As InvalidOperationException  ' message("Socket not disposed properly.")  clntDisc = True  End Try  End Sub  #End Region |

**Função Tratamento de Cliente** – Esta função trata em guardar a ligação e os dados do Cliente (Nome, IP) numa *hashtable* e trata de receber e redireccionar os comandos para o sítio correcto e efectuar a sua função respectiva. Também efectua a função de guardar o novo nome de utilizador se o mesmo for alterado. Trata do *disconnect* do cliente se o mesmo acontecer.

|  |
| --- |
| #Region "Client/Chat handling"  Public Class handleClnt  Dim cSocket As TcpClient ' Client Socket  Dim clName As String  Dim clList As Hashtable  Dim cnum As Integer  Dim cip As String  Dim nickcount As Integer  Public Sub startClient(ByVal inClntSocket As TcpClient, ByVal clntName As String, ByVal cList As Hashtable)  Me.cSocket = inClntSocket ' Client Socket  Me.clName = clntName ' Client Name  Me.clList = cList ' Client List  Me.cip = inClntSocket.Client.RemoteEndPoint.ToString  cip = cip.Substring(0, cip.IndexOf(":"))  Me.cnum = count ' Client Number  userlist.Add(Me.clName)  iplist.Add(Me.cip)  Dim ctThread As Threading.Thread = New Threading.Thread(AddressOf initiateChat) ' Client Threading  ctThread.Start()  End Sub  Private Sub initiateChat() ' Message from Client  Dim bytesFrom(10024) As Byte  Dim dataClnt As String ' Data from Client to Server  While (True)  Try  Dim Stream\_network As NetworkStream = cSocket.GetStream() ' Network Stream  Stream\_network.Read(bytesFrom, 0, CInt(cSocket.ReceiveBufferSize))  dataClnt = System.Text.Encoding.ASCII.GetString(bytesFrom)  If dataClnt.IndexOf("$") < 0 Then  clntDisc = True  cdisc()  Exit Sub  Else  dataClnt = dataClnt.Substring(0, dataClnt.IndexOf("$"))  message("Client - " + DateTime.Now + " " + Me.clName + " " + Me.cip + " " + "(" + Convert.ToString(cnum) + ")" + " : " + dataClnt)  Dim namelength As Integer  namelength = Me.clName.Length  If dataClnt.Contains("is sending a file") = True Then  fileflag = True  bcast(dataClnt, clName, False)  Else  fileflag = False  'msg2 = dataClnt  cmds()  If dataClnt = "/getusers" Then  msg2 = dataClnt  cmds()  bcast("Command typed: " + dataClnt, Nothing, False)  ElseIf dataClnt.Contains("/whois") Then  msg2 = dataClnt  cmds()  bcast("Command typed: " + dataClnt, Nothing, False)  ElseIf dataClnt.Contains("/myinfo") Then  msg2 = dataClnt  cmds()  bcast("Command typed: " + dataClnt, Nothing, False)  Else  bcast(dataClnt, clName, True)  End If  End If  If dataClnt.Contains("has changed his name to ") = True Then  Dim length As Integer  Dim copystring As String  Dim stringv2 As String = " has changed his name to"  copystring = dataClnt.Remove(0, namelength)  length = stringv2.Length  copystring = copystring.Remove(0, length - 1)  copystring = copystring.Replace(" ", "")  For i = 0 To userlist.Count - 1  If userlist.Item(i).ToString.Contains(Me.clName) Then  userlist.RemoveAt(i)  End If  Next  For i = 0 To Me.clList.Count - 1 'tofix  If clList.Contains(Me.clName) Then  clList.Remove(Me.clName)  clList.Add(copystring, cSocket)  End If  Next  Dim ctThread As Threading.Thread = New Threading.Thread(AddressOf initiateChat)  Me.clName = copystring  userlist.Add(copystring)  dataClnt = copystring  startClient(cSocket, copystring, clList)  ctThread.Start()  End If  End If  Catch ex2 As Exception  If cSocket.Connected = False Then  clntDisc = True  cSocket.Close()  cdisc()  Exit Sub  End If  End Try  End While  End Sub  Sub cdisc()  If clntDisc = True Then  message(clName + " " + "(" + Convert.ToString(cnum) + ")" + " " + "has disconnected. " + DateTime.Now)  'bcast(clName + " " + "(" + Convert.ToString(cnum) + ")" + " " + "has disconnected " + DateAndTime.Now, Nothing, False)  bcast(clName + " has disconnected. $", Nothing, False)  If Me.cnum = cnum Then  clList.Remove(Me.clName)  clList.Remove(Me.cip)  clList.Remove(Me.cnum)  clList.Remove(Me.cSocket)  userlist.Remove(Me.clName)  iplist.Remove(Me.cip)  End If  cSocket.Close()  clntDisc = False  Exit Sub  End If  End Sub  Sub cmds()  Dim whoisstr As String  If msg2 Is Nothing Then  Exit Sub  Else  Select Case msg2  Case "/getusers"  If msg2 = "/getusers" Then  Dim countclient As Integer = count - 1  Dim i As Integer = 0  While i <> countclient  Try  If i <> countclient Then  bcast("$Username: " & userlist.Item(i).ToString, Nothing, False)  Thread.Sleep(500) 'temporary fix?  i = i + 1  End If  Catch ex As ArgumentOutOfRangeException  Exit While  End Try  End While  End If  Case "/myinfo"  Dim getip As String  If msg2 = "/myinfo" Then  msg2 = ""  getip = Me.cip  bcast("IP is: " + getip + " ", Me.clName, False)  End If  End Select  If msg2.Contains("/whois") Then  whoisstr = "/whoisstr"  End If  Select Case whoisstr  Case "/whoisstr"  Dim copy As String  copy = msg2  msg2 = ""  copy = copy.Remove(0, 7)  Dim i As Integer  Dim test As Integer = count - 1  While i <> test  Try  If i <> test Then  If userlist.Item(i).Equals(copy) Then  Dim ip As String  ip = iplist.Item(i).ToString  bcast("Whois" + Environment.NewLine + "Username: " + copy + Environment.NewLine + "IP: " + ip, Nothing, False)  End If  i = i + 1  End If  Catch ex As ArgumentOutOfRangeException  End Try  End While  End Select  End If  End Sub  End Class  #End Region |

## Cliente

O cliente efectua a ligação ao servidor e permite o mesmo conversar com outros ligados ao Servidor (se existirem, é claro.), permite também o envio e recepção de ficheiros.

**Função conectar/desconectar** – Esta função trata de ligar ao servidor/porta escolhida nas preferências.

|  |
| --- |
| #Region "Connect/Dc button"  Private Sub cntbutton\_Click(sender As System.Object, e As System.EventArgs) Handles cntbutton.Click  If connB = True Then  Try  If info.Nickname = "" Or info.Nickname Is Nothing Or \_  info.IP = "" Or info.IP Is Nothing Or Not IsNumeric(info.IP) Then  MsgBox("Please review needed fields on preferences.")  Exit Sub  Else  cSocket = New System.Net.Sockets.TcpClient  msgthread()  cSocket.Connect(info.IP, info.Port) ' Connects to IP and port determined by svip and port.  servStream = cSocket.GetStream() ' Client to Server handling  Dim Stream\_out As Byte() = System.Text.Encoding.ASCII.GetBytes(info.Nickname + "$")  servStream.Write(Stream\_out, 0, Stream\_out.Length)  servStream.Flush()  Dim ctThread As Threading.Thread = New Threading.Thread(AddressOf getMsg) ' Client Threading  ctThread.Start()  cntbutton.Text = "Disconnect"  sndbtn.Enabled = True  connB = False  End If  Catch ex As SocketException  chattxt.Text = chattxt.Text + Environment.NewLine + " >> " + "Connection timed out! Check IP/port"  cntbutton.Enabled = True  sndbtn.Enabled = False  End Try  ElseIf connB = False Then  CheckForIllegalCrossThreadCalls = False  sndbtn.Enabled = False  cntbutton.Text = "Connect"  servStream.Flush()  cSocket.Close()  ListBox1.Items.Clear()  connB = True  CheckForIllegalCrossThreadCalls = True  End If  End Sub  #End Region |

**Função que recebe as mensagens** – Esta função recebe as mensagens de outros clientes que são enviadas para o Servidor, ou mensagens do Servidor em si.

|  |
| --- |
| #Region "Get Message handle"  Private Sub getMsg()  Try  Do While True  If cSocket.Connected = False Then  CheckForIllegalCrossThreadCalls = False  chattxt.Text = chattxt.Text + Environment.NewLine + " >> " + "Disconnected from server!"  ListBox1.Items.Clear()  CheckForIllegalCrossThreadCalls = True  Exit Sub  Else  servStream = cSocket.GetStream() ' Server to Client Handling  Dim buffSize As Integer ' Buffer size  Dim Stream\_in(10024) As Byte ' Buffer size for BuffSize (also max bytes allowed by server.)  buffSize = cSocket.ReceiveBufferSize  servStream.Read(Stream\_in, 0, buffSize)  Dim returnd As String = System.Text.Encoding.ASCII.GetString(Stream\_in)  read = "" + returnd ' Data received from server.  msgthread()  End If  Loop  Catch ex As IOException  CheckForIllegalCrossThreadCalls = False  chattxt.Text = chattxt.Text + Environment.NewLine + " >> " + "Connection terminated!"  cntbutton.Enabled = True  cntbutton.Text = "Connect"  ListBox1.Items.Clear()  sndbtn.Enabled = False  CheckForIllegalCrossThreadCalls = True  Catch ex2 As ObjectDisposedException  CheckForIllegalCrossThreadCalls = False  'chattxt.Text = chattxt.Text + Environment.NewLine + " >> " + "Object not disposed properly!"  cntbutton.Enabled = True  sndbtn.Enabled = False  CheckForIllegalCrossThreadCalls = True  Catch ex3 As InvalidOperationException  CheckForIllegalCrossThreadCalls = False  'chattxt.Text = chattxt.Text + Environment.NewLine + " >> " + "Invalid Operation exception!"  cntbutton.Enabled = True  sndbtn.Enabled = False  CheckForIllegalCrossThreadCalls = True  Catch ex4 As Win32Exception  End  End Try  End Sub  #End Region |

**Função de Multithreading e Gestão de mensagens** – Esta função trata de tratar o que é recebido nas mensagens e se for algum comando redireccionar para o sítio certo, se não é tratado como uma mensagem comum.

|  |
| --- |
| #Region "Multithreading + Message Handling"  Private Sub msgthread()  If Me.InvokeRequired Then ' Thank you André for helping me out on this part.  Me.Invoke(New MethodInvoker(AddressOf msgthread)) ' Multithread  Else  Try  Dim Copystr As String  Dim copystrlength As Integer  'If read.Contains(info.Nickname) AndAlso read.Contains("joined") Then  ' ListBox1.Items.Add(info.Nickname)  'End If  If read.Contains("$Username: ") Then  read = read.Substring(11)  Copystr = read  copystrlength = Copystr.LastIndexOf("/getusers")  If Copystr.Contains("/getusers") Then  Copystr = Copystr.Substring(0, copystrlength)  End If  If ListBox1.Items.Contains(Copystr) Then  read = ""  Copystr = ""  Exit Sub  Else  ListBox1.Items.Add(Copystr)  read = ""  Copystr = ""  Exit Sub  End If  End If  If read.Contains("Server Operator has terminated your connection. $") Then  read = read.Substring(0, 47)  cntbutton.PerformClick()  ListBox1.Items.Clear()  End If  If read.Contains(" has disconnected. $") Then  Dim copyread As String = read  Dim copyreadlength As Integer = copyread.LastIndexOf("$")  Dim readlength As Integer = read.LastIndexOf(" has disconnected. $")  copyread = copyread.Substring(0, copyreadlength)  read = read.Substring(0, readlength)  Dim readcopy As String = read  For i = 0 To ListBox1.Items.Count - 1  If ListBox1.Items(i).ToString.Contains(readcopy) Then  ListBox1.Items.RemoveAt(i)  ListBox1.Refresh()  End If  Next  read = ""  chattxt.Text = chattxt.Text + Environment.NewLine + " >> " + copyread  Exit Sub  End If  If read.Contains("has changed his name to ") = True Then  ListBox1.Items.Clear()  msgtxt.Text = "/getusers"  sendBtn.PerformClick()  msgtxt.Text = ""  End If  If read.Contains("joined ") Then  ListBox1.Items.Clear()  msgtxt.Text = "/getusers"  sendBtn.PerformClick()  msgtxt.Text = ""  End If  Catch ex As NullReferenceException  End Try  chattxt.Text = chattxt.Text + Environment.NewLine + " >> " + read  End If  End Sub  #End Region |

## *Stresstester*

Este programa foi desenvolvido utilizando partes do código do Cliente e é utilizado meramente para testes.

**Função de Stresstest** – Esta função realiza os testes quando os valores que são pedidos forem inseridos.

|  |
| --- |
| #Region "Stresstest Button"  Private Sub Button1\_Click(sender As System.Object, e As System.EventArgs) Handles Button1.Click  Dim usercount As Integer  If Not IsNumeric(TextBox1.Text) Then  MsgBox("Please insert a numeric value into the textbox.")  Else  Try  Do  Try  Dim nickname As String = "yo" & usercount  cSocket = New System.Net.Sockets.TcpClient  msgthread()  If String.IsNullOrEmpty(TextBox2.Text) Then  MsgBox("Please insert a value into the Port Textbox.")  Exit Sub  End If  cSocket.Connect("127.0.0.1", TextBox2.Text) ' Connects to IP and port  servStream = cSocket.GetStream() ' Client to Server handling  Dim Stream\_out As Byte() = System.Text.Encoding.ASCII.GetBytes(nickname + "$")  servStream.Write(Stream\_out, 0, Stream\_out.Length)  servStream.Flush()  usercount = usercount + 1  count = count + 1  Dim ctThread As Threading.Thread = New Threading.Thread(AddressOf getMsg) ' Client Threading  ctThread.Start()  Catch ex As SocketException  If MsgBox("Error performing Stresstest.") = DialogResult.OK Then  My.Computer.FileSystem.WriteAllText("StressTest " + FormatDateTime(DateAndTime.Today, DateFormat.LongDate) + ".log", "Stress test FAILED at: " + DateTime.UtcNow + Environment.NewLine, True)  Exit Sub  End If  End Try  Loop Until count = TextBox1.Text  MsgBox("Stress test completed." + Environment.NewLine + "Amount of users tested: " & usercount)  Catch ex As IOException  End Try  End If  End Sub  #End Region |

# Historial do Programa

0.1a:

* Organização de ideias, revisão dos objectivos e início do projecto;
* Desenho da Interface;
* Início do desenvolvimento do Servidor.

0.1b:

* Servidor encontra-se a 50% de completo;
* Início do desenvolvimento do Cliente.

0.1c:

* Servidor aceita agora mais que um cliente;
* Algumas excepções arranjadas;
* Cliente permite a ligação para além do *localhost;*
* Início do desenvolvimento do *Multithread.*

0.1d:

* Início do desenvolvimento das transferências de ficheiros;
* Algum desenvolvimento do *Multithread;*

0.1e:

* Transferências de ficheiros terminado;
* Protótipo de notificação para transferências de ficheiros.

0.2:

* Implementação de uma estrutura para organizar a informação de cada utilizador;
* Implementação de uma janela de preferências;
* Início da criação de comandos para utilizadores;
* Lista de clientes optimizada;
* Início da criação do sistema *Whois.*

0.3:

* Implementada uma lista de servidores manipulada pelo utilizador através de um ficheiro XML;
* Concluído o sistema *Whois;*
* Planeamento para o sistema PM (*Private Message*) entre utilizadores;

0.4:

* Sistema PM adiado devido a requerer um desenvolvimento totalmente novo do projecto;
* Adicionado um leitor de *feed* RSS;
* Optimização no servidor;
* Optimização na lista de clientes.

0.5:

* Lista de clientes agora remove clientes desconectados;
* Adicionado um *stresstester* (em desenvolvimento.);
* Adicionados comandos via Servidor para o Administrador.

0.5a:

* Forçou-se o *input* da Console no arranque do Servidor;
* Alterou-se todos os procedimentos “*if*” nos comandos de consola para um procedimento “*select*”.

0.6:

* Adicionada uma função “*say*” no Servidor;
* Parcialmente implementado e desactivado de momento, verificador de *nickname* (se existente, altera automaticamente).

0.7:

* Início do desenvolvimento da actualização da lista de clientes automática por via de um temporizador (*Timer*).

0.8/0.9:

* Lista de clientes optimizada;
* Verificador de *nicknames* concluído e implementado;
* Actualizador da lista de clientes implementado;
* Função para ligar e desligar o temporizador no Servidor implementada;
* Implementada uma função para alterar o Estado de cada utilizador (Online, Ocupado, Ausente);
* Modificação na estrutura do Cliente que permite relembrar as configurações previamente utilizadas.
* Implementada a opção de o utilizador definir uma porta (já tinha sido implementado anteriormente, mas foi removido devido a testes.)
* Actualizado o *Stresstester* com a opção de definir a porta
* Implementada uma função nova no Servidor (locate)

Pre-1.0/a:

* Adicionado um splash e about box no Client

Pre-1.0b:

* Adicionado material visual no Servidor, about e splash

Pre-1.0c:

* Implementada a função de verificação de placas de redes existentes

Pre-1.0d:

* Melhoramentos nas excepções

Pre-1.0e:

* Função de verificação de nicknames melhorada.

Pre-1.0f:

* Adicionado material visual
* Stresstester actualizado
* Adicionado as funções de info e about no Servidor.

Pre-1.0g:

* Implementado a função de banir e desbanir utilizadores no Servidor.

# Tecnologias usadas

Este projecto foi desenvolvido recorrendo ao *framework VB.NET (Visual Basic.NET)* presente no Microsoft Visual Studio.

Usou-se como recurso para testes o sistema de máquinas virtuais que foi permitido utilizando na escola o Oracle VirtualBox e em casa utilizando o VMWare.

Também foi utilizado um *Stresstester* desenvolvido especificamente para testar a carga máxima da CPU, utilização de RAM máxima e quantos clientes conseguia suportar o Servidor.

# Conclusão

Este projecto foi desenvolvido para avaliação como Projecto Tecnológico.

Na nossa opinião, foi um projecto um pouco difícil em algumas partes mas que foram ultrapassas com rapidez e uma rápida implementação de novas revisões com novas funcionalidades e as que não estavam a 100% arranjadas.

Foi-nos introduzido o sistema de *Threading* e *Sockets* via TCP/IP, matéria que não foi dada em sala de aula e que foi abordada e estudada por nós próprios, o que nos deu motivação para continuar a elaborar este projecto.

Foi um projecto que no final de tudo, apos tantas horas a rondar, testar o código e arranjar *bugs* nos deu uma base sólida para continuarmos com tanta motivação no futuro.

# Agradecimentos

No fundo, gostaríamos de agradecer ao Prof. Manuel Tróia e à Profª. Sandra Rodrigues por toda a ajuda e tempo dispensado na realização deste projecto.

Gostaríamos também de agradecer à Profª. Sandra Cid pela ajuda que ofereceu na verificação deste relatório.

E também aos colegas de turma, que acompanharam o desenvolvimento do mesmo e deram algumas ideias para serem implementadas.