Maak een chat applicatie

<https://goo.gl/LjtVmt>

[Client/Server chatten](#_30j0zll)

[[Beroepsproduct] Multi-Client uitbreiding](#_1fob9te)

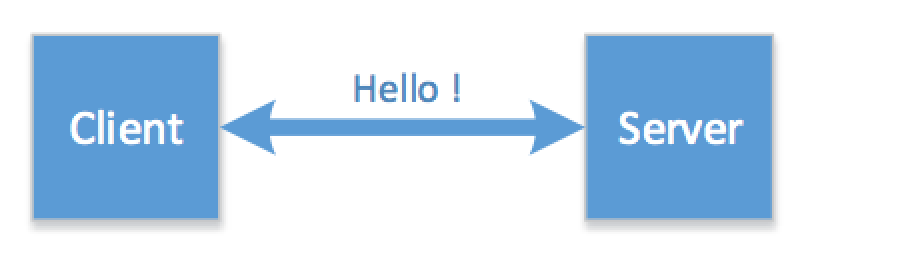
Markdown template

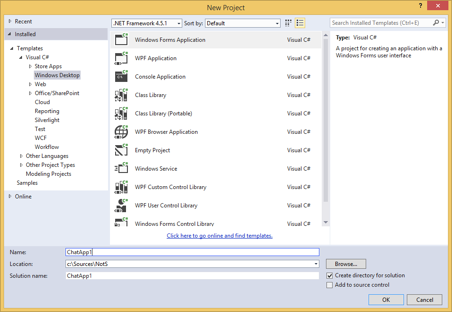
# Client/Server chatten

In deze opdracht wordt je geleid tot het maken van een client/server chatapplicatie. Deze applicatie wordt gebruikt als basis en steeds verder uitgebreid tot uiteindelijk een volledig werkende load balancer.

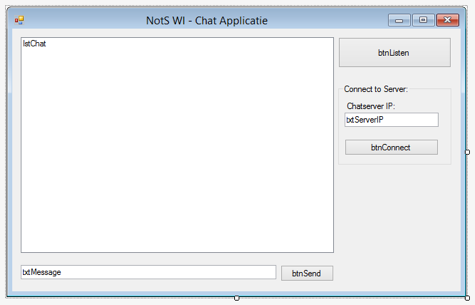
Hieronder weergegeven de netwerkcomponenten die we nodig hebben:

* Een “server” (definitie: een programma dat diensten levert aan andere computers). Dit is het netwerkcomponent dat staat te wachten op een client die met hem wil verbinden.
* Een client; Deze wil graag met een server verbinden om ermee te communiceren.



1. Maak in Visual Studio een nieuw project aan van het type C# à Windows Desktop à Windows Forms Application

2. Design het ***form*** zodanig dat het er zo ongeveer uitziet



3. Declareer een **TcpClient**, een **NetworkStream** en een **Thread** zodanig dat ze toegankelijk zijn vanuit de event handlers van de knoppen.

4. Declareer een **delegate** met een input-parameter van het type string en geen returnwaarde.

5. Maak een private method met een input-parameter van het type string.

In dit event kijk je d.m.v. this.lstChat.InvokeRequired of gebruik van de delegate noodzakelijk is. (dit is het geval wanneer er *niet* vanuit de UI-Thread een ui-component wordt aangesproken)

* Indien ja, maak dan een instantie van de in 4 gedeclareerde delegate aan, en roep ‘m aan met this.Invoke. Invoke heeft als parameters nodig: de zojuist gemaakte delegate, een new object[] en de tekst die moet worden doorgegeven.
* Indien nee, dan zitten we in de juiste thread en kan de gui-update worden uitgevoerd.

6. **[Server gedeelte]** In de event handler van het OnClick event van btnListen doe je het volgende:

1. Declareer en initialiseer een **TcpListener** die luistert op poort 9000 op alle IP-adressen van de pc. Luisteren op alle IP-adressen doe je door **IPAddress.Any** op te geven als adres-parameter.
2. Start de **TcpListener**.
3. Voeg aan lstChat een item toe: “Listening for a client.”.
4. Zodra een client verbinding wil maken, voeg je nog een item toe aan **lstChat**: “Connecting...”.
5. Initialiseer de eerder gedeclareerde TcpClient. Deze TcpClient krijg je van de **TcpListener** m.b.v. de methode **AcceptTcpClient**().
6. Initialiseer de eerder gedeclareerde **Thread**. De aan te roepen methode voor de **ThreadStart** parameter heet **ReceiveData**(); die we zo gaan maken.
7. Start de Thread.

7. Maak een nieuwe methode **ReceiveData**(). In de methode doe je het volgende:

a. Declareer een int- en een string-variabele.

b. Declareer en initialiseer een Byte-array van 1024 bytes.

c. Initialiseer de eerder gedeclareerde **NetworkStream** met het resultaat van de methode **GetStream**() van de **TcpClient**

d. Voeg, **via de delegate**, aan lstChat een item toe: “Connected!”. (WAAROM?)

e. Maak een while-loop met als conditie true. In de loop gebeurt het volgende:

i. Lees met de methode Read() van de NetworkStream binnenkomende gegevens in de byte-array.

ii. Zet met behulp van de methode Encoding.ASCII.GetString de gegevens van de Byte-array in de string-variabele.

iii. Als de string-variabele gelijk is aan “bye” breek dan uit de loop.

iv. Voeg de string-variabele **via de delegate** als een *item* toe aan lstChat.

f. Vul de Byte-array m.b.v. Encoding.ASCII.GetBytes() met de tekst “bye”

g. Schrijf de Byte-array m.b.v de methode Write() naar de NetworkStream. (WAAROM?)

h. h. Sluit de NetworkStream en de TcpClient.

i. j. Voeg, via de delegate, aan lstChat een item toe: “Connection closed.”

8. **[Client gedeelte]** In de event handler van het OnClick event van btnConnect doe je het volgende:

a. Voeg een item toe aan lstChat: “Connecting...”.

b. Initialiseer de eerder gedeclareerde TcpClient. Maak verbinding met het IP-adres dat is opgegeven in txtServerIP op poort 9000.

c. Initialiseer de eerder gedeclareerde Thread. De aan te roepen methode voor de ThreadStart is ook nu ReceiveData().

d. Start de Thread.

9. In de event handler van het OnClick event van btnSend doe je het volgende:

a. Declareer en initialiseer een Byte-array van 1024 bytes.

b. Vul de Byte-array m.b.v. Encoding.ASCII.GetBytes() met de tekst die in txtMessage is ingetikt.

c. Schrijf de Byte-array m.b.v de methode Write() naar de NetworkStream.

d. Voeg de tekst van txtMessage ook als een item toe aan lstChat.

e. Maak txtMessage leeg en zet de focus terug naar txtMessage.

10. Compileer de applicatie en kopieer deze naar een andere pc. M.b.v. ipconfig.exe kun je het IP-adres van je pc opzoeken. Laat één van de applicaties luisteren en laat de andere verbinding maken. Je kunt beide applicaties ook op één pc draaien. Vul dan het IP-adres 127.0.0.1 in IPv4 verwijst naar localhost. (<https://nl.wikipedia.org/wiki/Localhost>)

**Let op: je kan er maar 1 tegelijk debuggen.**

11. Verander de grootte van de byte-array in de ReceiveData methode eens in 2 i.p.v. 1024. Kijk naar het resultaat.

a. Welk probleem of welke problemen doen zich voor?

b. Hoe komt dat?

c. Bedenk een oplossing.

**d.** **Implementeer deze oplossing.**

12. Verbeter het programma op de volgende punten:

a. Foutafhandeling (try..catch..finally), zeker rondom het maken van de verbinding

b. Schakel knoppen uit die niet ingedrukt mogen worden. (Denk eraan dat je vanuit de andere thread niet zomaar knoppen mag in- en uitschakelen.)

c. Zorg dat je met zo min mogelijk clicks snel zaken kan testen, bijvoorbeeld door een zinvolle default-invulling te geven, bijvoorbeeld door alvast “127.0.0.1” in de textbox te plaatsen; de berichten te kunnen verzenden door op “enter” te drukken ipv eerst weer je muis aan te moeten raken etc.

d. Maak duidelijker welke berichten binnenkomen en welke uitgaan door bijvoorbeeld “>>“ voor binnenkomende tekst te zetten en “<<” voor uitgaande.

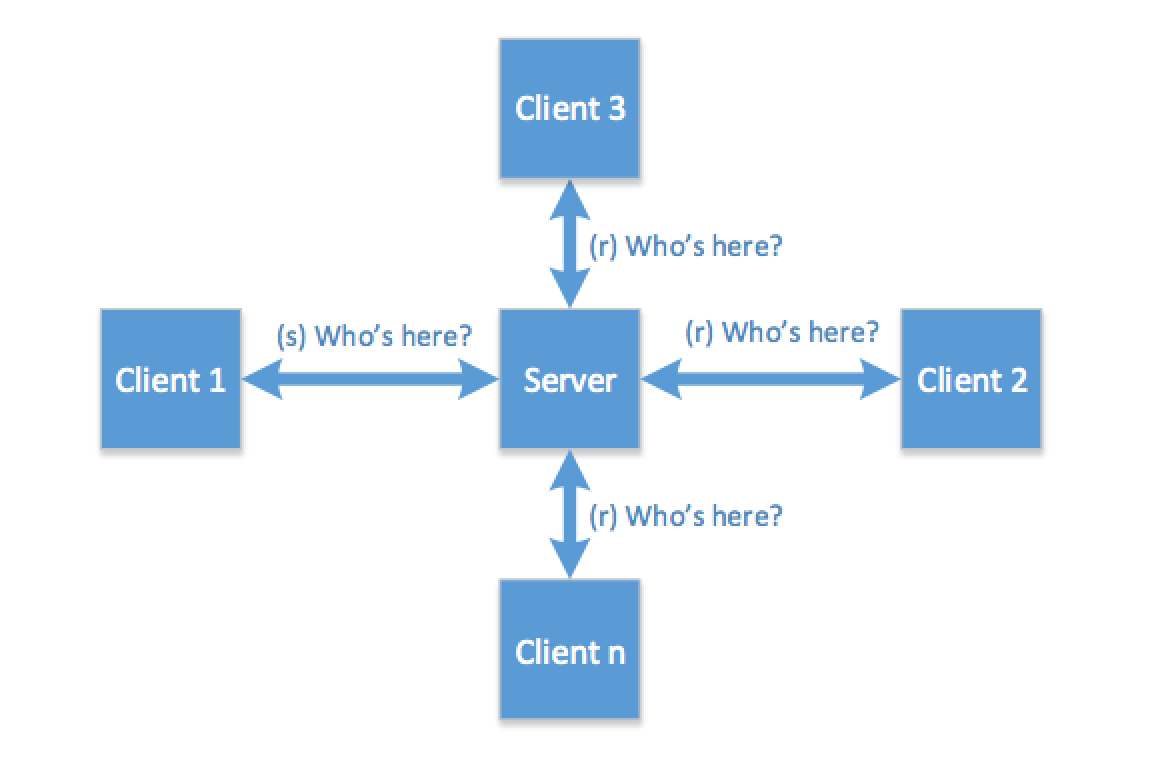
e. Wat gebeurt er trouwens als de listbox vol loopt? Is dit op te lossen?

# 

# 

# **[Beroepsproduct]** Multi-Client uitbreiding

Het is nu slechts mogelijk om maar met 1 client tegelijkertijd te verbinden, maar het is leuker om met meerdere clients in een groepsgesprek te komen. Hieronder wordt deze gewenste architectuur, welke o.a. ook door [IRC [Internet Relay Chat]](https://tools.ietf.org/html/rfc1459) wordt toegepast, schematisch weergegeven:



**Functionele requirements:**

· De gebruiker is in staat om berichten te verzenden naar alle andere gebruikers

· De gebruiker is in staat om berichten te ontvangen van alle andere gebruikers

· De gebruiker kan zich aanmelden om deel te nemen aan de berichtenstroom

· De gebruiker is te allen tijde in staat om zich af te melden van de berichtenstroom

**Technische requirements:**

· Het systeem wordt geschreven in C# op het .Net platform

· Netwerkverkeer gebeurt op basis van het TCP protocol

· Er wordt gebruik gemaakt van multi-threading en/of asynchrone afhandeling.

· Het systeem moet om kunnen gaan met een variabele buffergrootte. Deze moet instelbaar zijn vanaf de user-interface

· Het systeem maakt inzichtelijk wat er zich afspeelt op netwerkniveau.

**Niet-functionele requirements:**

· Het systeem moet robuust zijn. Dit houdt in dat er goed moet worden omgegaan met protocol fouten, verkeerde user-input etc.

· Het systeem moet minstens 20 gelijktijdige gebruikers ondersteunen

· De user-interface is te allen tijde responsief.

**Deployment requirements:**

· Annotaties van methodes, input & output parameters en types

· Annotaties van alle control structures (if...then, loop, while etc)

· Documentatie van gebruikte concepten, waaronder

o Generics

o Boxing & Unboxing

o Delegates & Invoke

o Threading & Async

· In de vorm van:

o Beschrijving van concept in eigen woorden

o Code voorbeeld van je eigen code

o Authentieke en gezaghebbende bronnen

o Alternatieven

# Test cases

1. Twee servers op een zelfde IP en Port
   1. Start chatApp 1 op local host, open een tweede app kies je ook voor host.
2. Connecten wanneer er geen server is
   1. Start chatApp 1 probeer te connecten terwijl er geen server actief is.
3. Connecten zonder username
   1. Start chatApp 1 probeer te connecten zonder dat je een username hebt ingevuld.
4. Server disconnecten wanneer er clients zijn verbonden
   1. Start een chatserver en 1 of 2 clients. Stuur berichten naar elkaar en druk vervolgens op de server op disconnect
5. Resize het window
   1. Resize het window naar alle mogelijk formaten, alles zal op z'n plek blijven :)
6. Bericht verzenden wanneer niet verbonden
   1. Probeer een bericht te verzenden wanneer je niet bent verbonden
7. Buffersize aanpassen terwijl chat actief is
   1. Probeer de buffer size via de settings-tab aan te passen terwijl je bent verbonden met een server

# Markdown template

Bijgaand de template ten behoeve van de documentatie van je beroepsproduct

>>>>>

Studentnaam:  
  
Studentnummer:   
  
---  
#Algemene beschrijving applicatie  
  
##Generics  
###Beschrijving van concept in eigen woorden  
###Code voorbeeld van je eigen code  
###Alternatieven & adviezen  
###Authentieke en gezaghebbende bronnen  
  
##Boxing & Unboxing  
###Beschrijving van concept in eigen woorden  
###Code voorbeeld van je eigen code  
###Alternatieven & adviezen  
###Authentieke en gezaghebbende bronnen  
  
##Delegates & Invoke  
###Beschrijving van concept in eigen woorden  
###Code voorbeeld van je eigen code  
###Alternatieven & adviezen  
###Authentieke en gezaghebbende bronnen  
  
##Threading & Async  
###Beschrijving van concept in eigen woorden  
###Code voorbeeld van je eigen code  
###Alternatieven & adviezen  
###Authentieke en gezaghebbende bronnen  
  
<<<<<