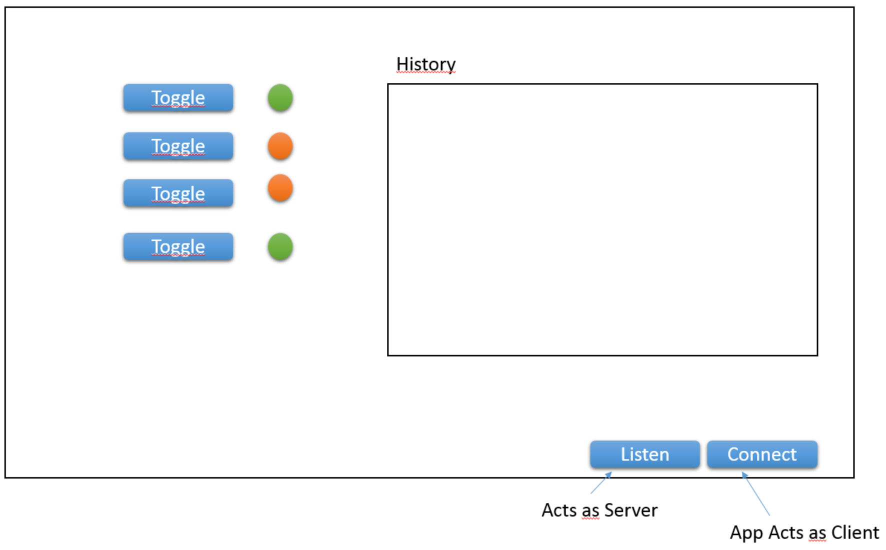
# EXAMPLE 1

Create One Application which can act as Client or as Server as shown in the screenshot.



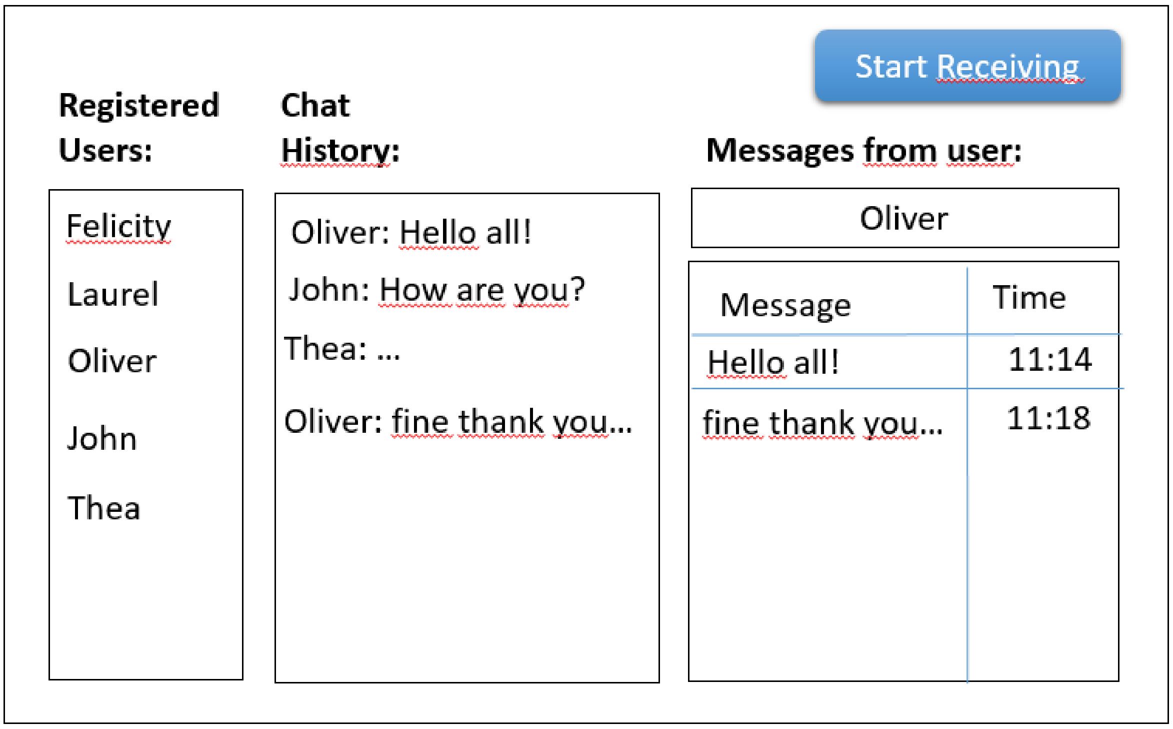
If the user triggers a change via the Toggle buttons the data is transmitted to the client and mirrors the set of the graphical representation of the green or red lights. Listen starts the application in Server mode and Connect starts the Application in Client mode (which connects to the server). The setting supports several clients for one server. After pressing one of the toggle buttons the data should be sent to the other clients which should change the lights corresponding to the sender. The states are always sent from server to clients (not vica versa) but the history is shown on all clients and on the server. The clients do not displays the Toogle buttons. After the user has clicked the Listen/Connect button they should be disabled.

The History is a datagrid which logs each button click with (button id [Button1…Button4 and state [green|red] and a timestamp).

# Example 2

Create a Server Application which can receive Chat messages from several Telnet Applications. The server displays the messages in several ways. The first info shows the registered users. The first message which is sent to the server is used as Username. The chat history shows the received messages. All messages are redirected to all other registered users. If the user of the server application selects a user entry of the registered users element all messages are listed in the “messages from user” element with a timestamp. The selected user information is also listed in an label above. The server starts receiving messages after the button Start Receive was clicked.

**GUI**



Felicity
Laurel
Oliver
John
Thea

Oliver: Hello all!
John: How are you?
Thea: ...
Oliver: fine thank you…

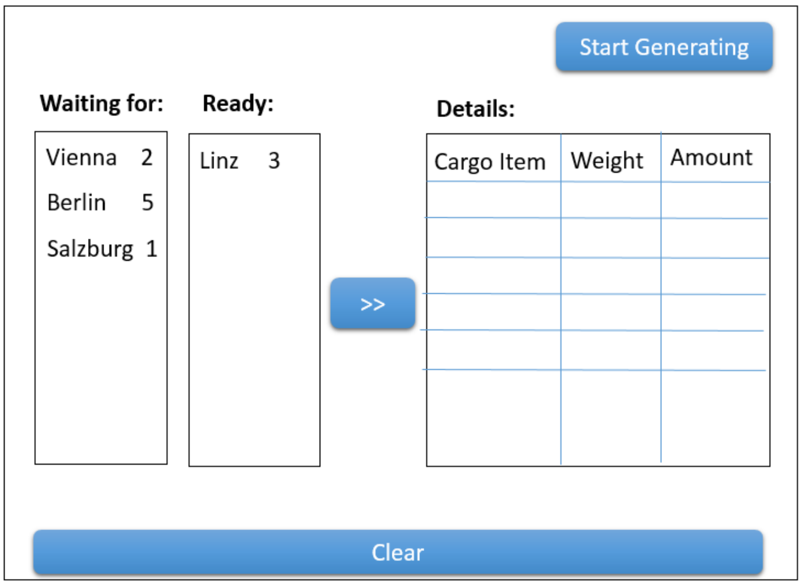
Oliver

| Message | Time |
|---|---|
| Hello all! | 11:14 |
| fine thank you… | 11:18 |

# Example 3

Create ONE Application which generates every 5 Seconds an Object of type „Cargo“ this Cargo can contain several Cargo Items like (Window, Bricks, Timber) with amount (int) and weight (float) information for each entry. Each created cargo has also a description (like Vienna, berlin, ,….) and a delivery time which is between 1 and 5 seconds. The value for the delivery time is calculated randomly. Create an array of Cargo objects with 5 entries and use these entries to generate them randomly every 5 seconds. After a Cargo is randomly selected the GUI displays it in the “Waiting for” GUI Element with the description and the delivery time. After the delivery time is elapsed the information is put to the “Ready” List. If the user clicks on the “>>” Button the details are displayed in the Details datagrid. The generation of the objects start after the user clicks on the “Start generating” button. If the user clicks the Clear button all entries are removed.

**GUI**



**Hints:**

That example does not contain tcp communication parts. Use threads to simulate the different delivery times.

# Example 4

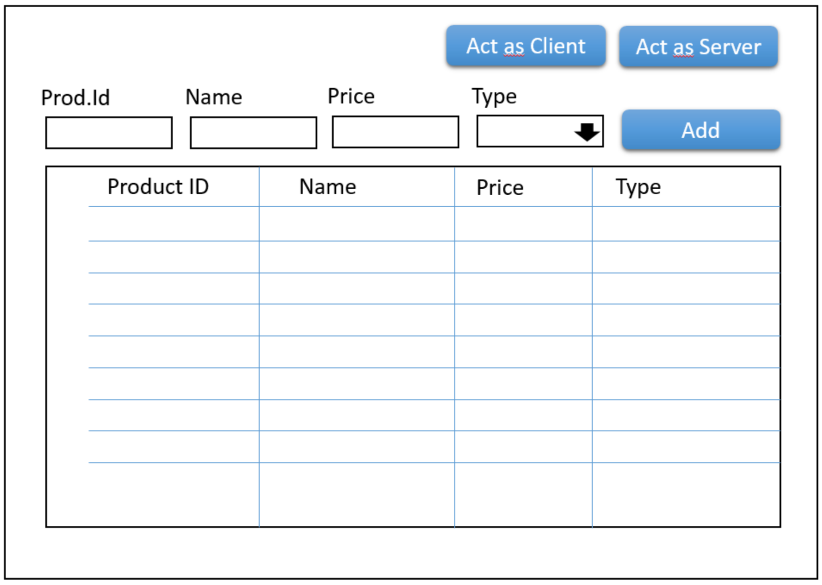
Create a System which displays data based on a model (Product with fields: Name, ProductId (like A-453-DT5), Price, Type [Engine, gears, body] in a data grid.

The Application could act as server OR client. Both can add new entries to the grid by adding all the necessary data and clicking Add. The Type information is provided by a Combobox. The add button is only visible if all fields are populated and a connection established. The added entries are also sent to the other connected applications so that all display the same data.

Act as Client or Act as Server is disabled after one of the buttons was clicked.

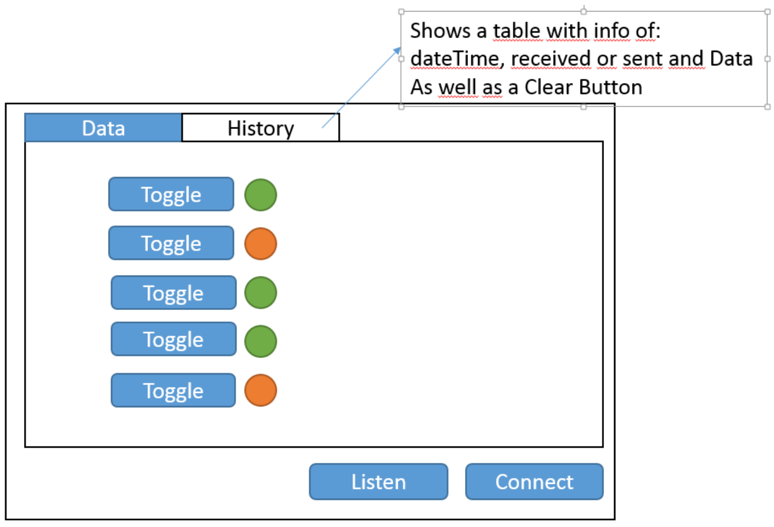
One server can handle more clients.

GUI:



# Example 5

Create One Application which can act as Client or as Server as shown in the screenshot.


Shows a table with info of:
dateTime, received or sent and Data
As well as a Clear Button
Data
History
Toggle
Toggle
Toggle
Toggle
Toggle
Listen
Connect

If the user triggers a change via the Toggle buttons the data is transmitted to the client/or Server and mirrors the set of the graphical representation of the green or red lights. Listen starts the application in Server mode and Connect starts the Application in Client mode (which connects to the server). The setting supports only one client for one server. After pressing one of the toggle buttons the data should be sent to the other application (client or server) which should change the lights corresponding to the sender.  The amount of buttons and circles is set via a Collection (so you need to use datatemplates for presentation of the content).

The History Tab contains a Datagrid which logs each receiving or sending action. The Datagrid should contain a DateTime (Use DateTime.Now function), a flag to visualize if it was sent or received and the Data.

A clear Button should be available to clear the Datagrid entries of the history.

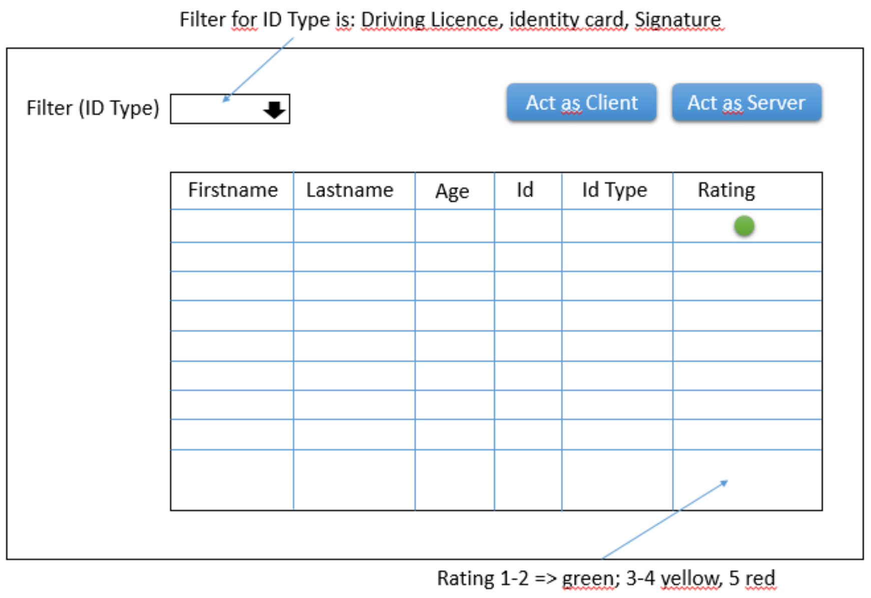
# Example 6

Create a System which displays data based on a model (Person with fields: Firstname, Lastname, Age, ID Nummer, ID Type [Driving License, identity card, Signature], and Rating [1-5]) in a grid.

The Rating is displayed as a green, yellow and red ellipse. if the Rating is 1 or 2 => green; 3 => yellow; otherwise red.

The Application could act as server OR client. The server generates the data (generate some demo data) and the clients (more!) just displays the data. The user can change the fields: Firstname, Lastname, Age, ID number and ID type via the datagrid. If data changes it is sent to all clients which updates their GUI with the new data. A Dropbox enables the user to filter the results shown in the datagrid. These changes do NOT have to be submitted to the clients. The changes should be updated immediately after the changes are done (no button for submit)

GUI:


Filter for ID Type is: Driving Licence, identity card, Signature
Filter (ID Type)
Act as Client
Act as Server
Firstname
Lastname
Age
Id
Id Type
Rating
Rating 1-2 => green; 3-4 yellow, 5 red

# Example 7

The application is able to receive production orders from several command line applications which send these orders every 5 seconds. The received orders are listed in a listbox. The application prints the detail info of the order after the user selects an entry of the listbox.

Format of a production order:   NAME{Component1;Component2;…}Amount;Completiontime

CompletionTime format: hh:mm

Blueprints are available for:

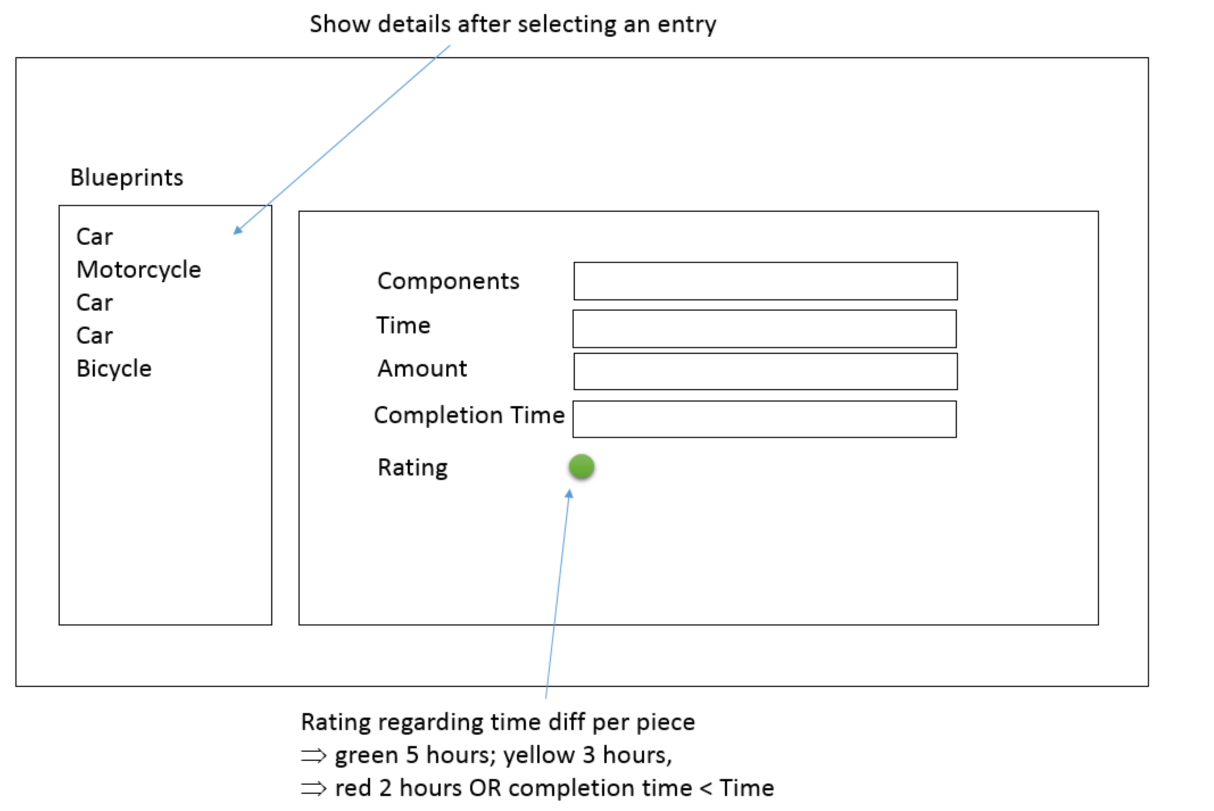
- Car{Engine;Tires;Center;Gear}
- Motorcycle{Engine;Tires;Chassis;Tank}
- Bicycle{rack;Tires;Pedals}

Example for a message from the client to the server:

Car{Engine;Tires;Center;Gear}5;10:20

The server processes a (Risk)Rating by calculating the time difference between receiving Time (Time set by server after receiving the message) and Completion Time and displays that state via a red, yellow or green light on the GUI.

GUI:


Show details after selecting an entry
Blueprints
Car
Motorcycle
Car
Car
Bicycle
Components
Time
Amount
Completion Time
Rating
Rating regarding time diff per piece
⇒ green 5 hours; yellow 3 hours,
⇒ red 2 hours OR completion time < Time

**Adapt the Solution and use DataTemplates to display the Blueprints differently)**

# Example 8

Create two Applications (in one solution – solution has to be named like [YourLastname]_210214) . One Application generates as console application every 5 seconds. (see hint) ships with a payload and sends that data to a server WPF based dashboard. The client outputs the sent data on the console

The Submission string contains:

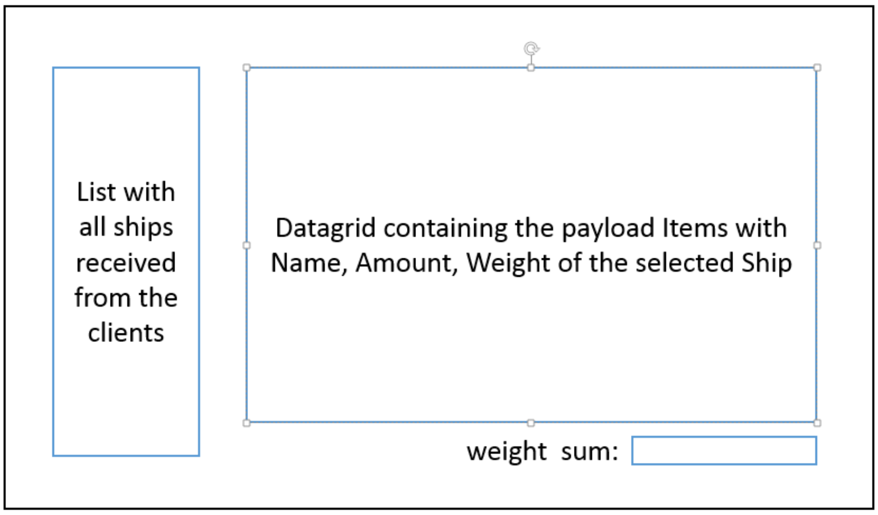
- Ship ID (each client contains one randomly generated ID)
- Cargo Information
  - Name
  - Amount
  - Weight (Sum)

The string with cargo could look like:

ShipId@recorder,20000,25000|DVDPlayer,10000,20000|PCs,50000,200000

So ID and cargo a separated via "@" the cargo ame and amount, weight with a "," and different cargo with "|"

The server dashboard should look like this:


List with all ships received from the clients
Datagrid containing the payload Items with Name, Amount, Weight of the selected Ship
weight sum:

**Display the payload items via datatemplates**