# Name:

Carl Orge Retzlaff, PCOLGGYRWT (just me)

# Functionality:

The requirements from the exercises have been fulfilled, the message board itself is fully functional, and the distributed servers as well. Private messages can be sent to users and public messages posted to all boards, the login and logout works as well.  
I tested it with three servers, but theoretically there could be many more servers, which keep the message database actualized by publishing changes to the public messages all other web browsers.  
The bug of message duplication is something that is still pending and could be done in the future.

# Errors:

Sometimes messages get duplicated when switching between themes, which is most likely due to a mismatch between the req/res and pub/sub.

# Improvements made:

The dmserver and dmforum are now called a bit differently to allow invocation with the command line.

Dmserver is called as follows: *node dmserver.js 5555 5556 5557 5558 5559 5560*Every first value is a address for Req/Res, every second an address for Pub/Sub. This command therefore starts three servers, listening to requests on 5555,5557 and 5559 and publish on 5556, 5558, 5560  
Forum is called as follows: *node forum.js 5555 5556 10000*  
The first value is the port for Req/res, the second for Pub/Sub and the third indicated the Web Browser Port.

Also, a lot of documentation was added where due, and DMserver as well as forum log now the different steps of the process to enable better troubleshooting.

# Further Improvements:

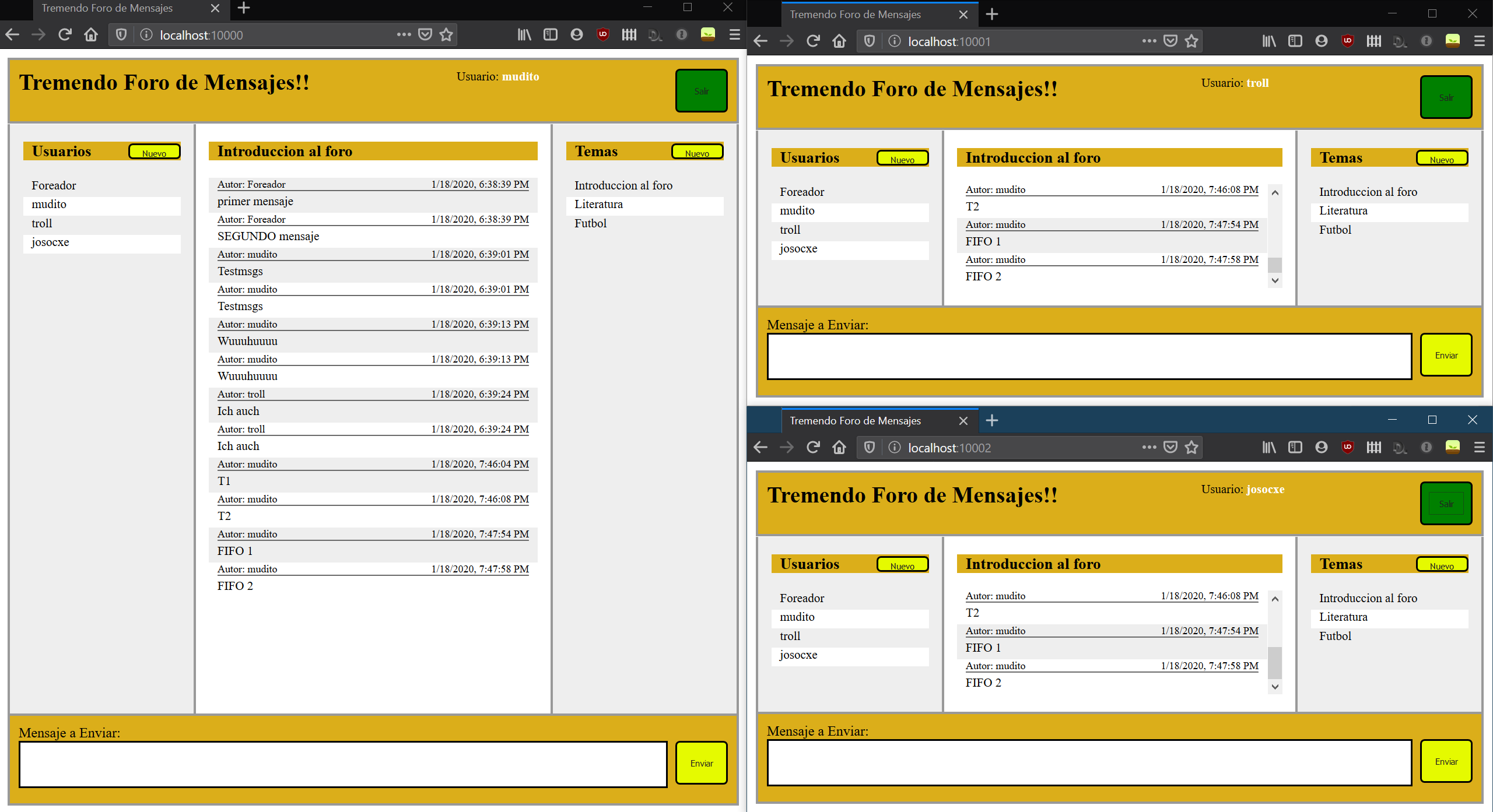
Ultimately, all messages etc should be propagated via pub/sub and not the req/res and the dm\_remote, since the callback queue is prone to problems due to the distribution. If all messages would be published the system would be more robust and probably no message duplication would take place.

# System tests:

Of course I just checked whether all messages arrive at all web servers, arrive in the correct order and arrive unchanged. These tests were successful. Also, the mozilla firefox console was used to troubleshoot messages not arriving in the other servers (due to the dreaded “message lost due to asynchrony” error).

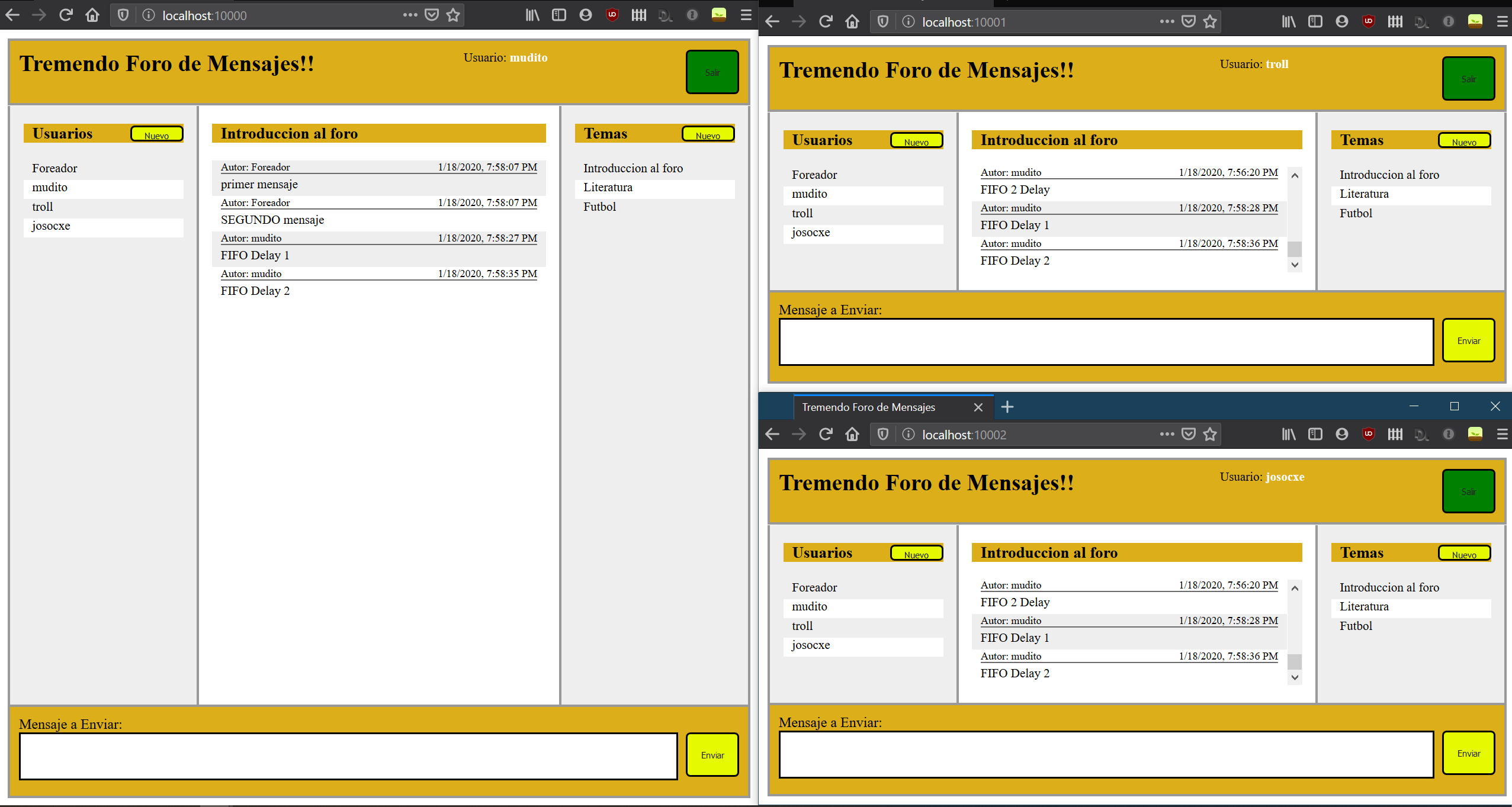
# Consistency tests:

FIFO Test 1:



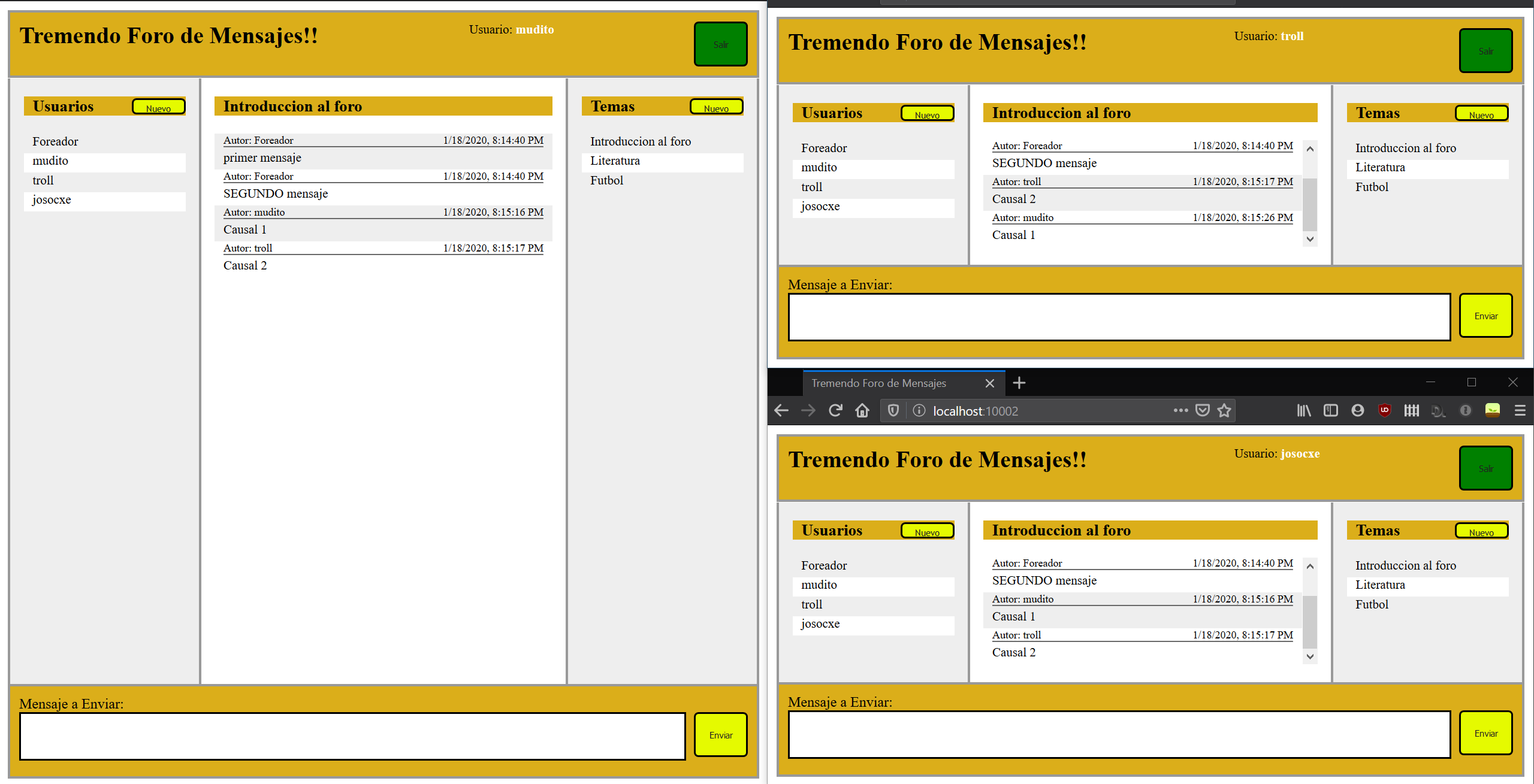
The messages work just as expected, first arrives FIFO 1 then FIFO 2

When adding a delay of one second, the messages FIFO Delay 1 and FIFO Delay 2 also arrive as expected



Causal Consistency 2:

Causal 1 was sent from 10000, Causal 2 from 10001. The



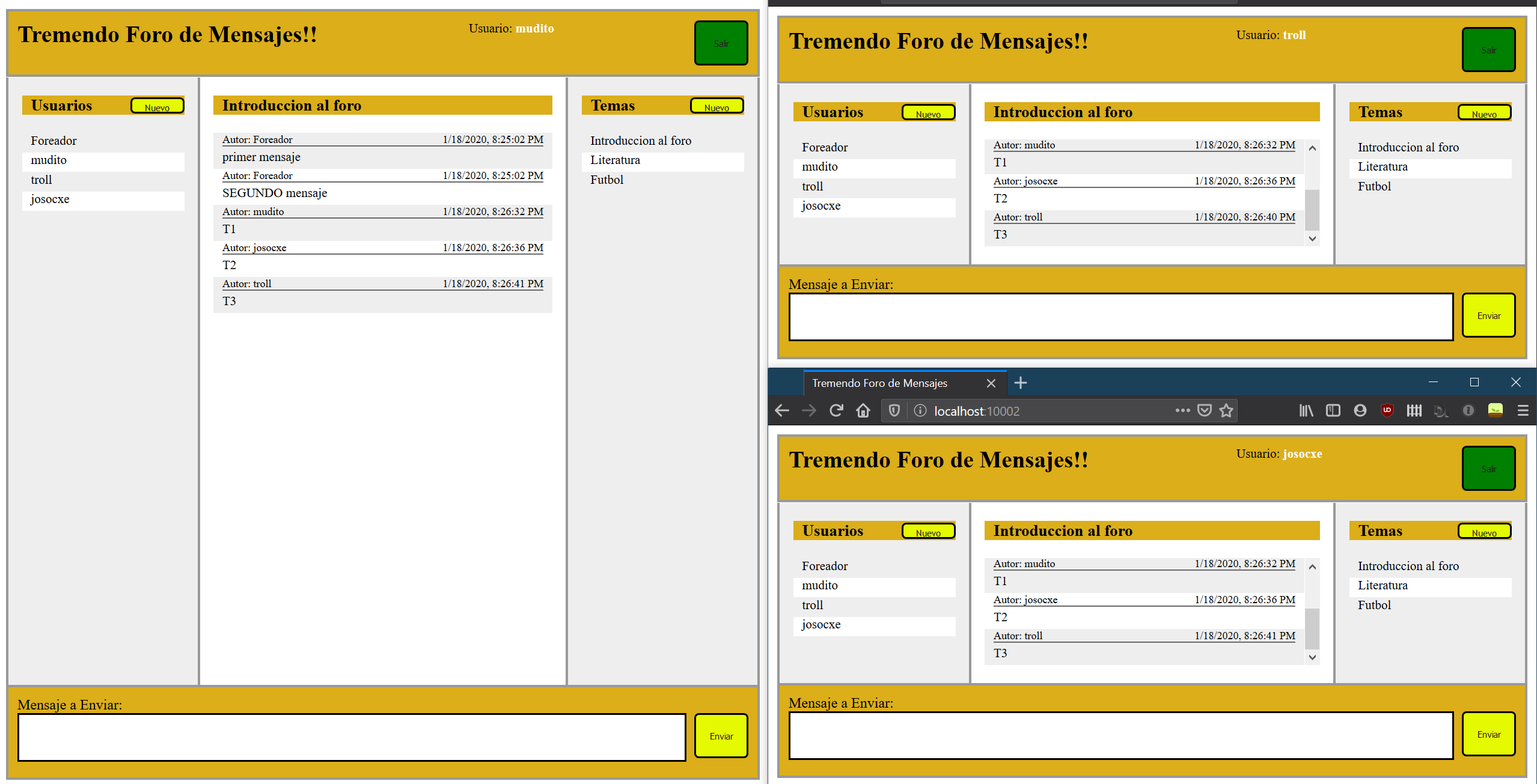
The causal test with the delays shows that S2 has the message order “Causal 2, Causal 1”, whereas the 10000 and 10002 both have the other causal message order “Causal 1, Causal 2”. The delay is in place for messages from 10000 to 10001

Sequential Consistency:

Sequential consistency is defined as:

In sequential consistency [15], the result of an execution is the same as if the operations of all processes will be executed in some sequential order, and the operations of each process would appear in that sequence in the order dictated by their respective program.

Since all messages are directly replicated in the other servers, sequential consistency is given. The delay as given in the causal test disrupts this consistency (since sequential consistency implicates causal consistency), but without this limitation all messages appear in the correct order of appearance for all processes.



Our system also implements strict consistency since at least from my side only one action can be executed at a time, which are all also directly completed, which therefore does not really allow for any consistency problems. Since all actions are also directly distributed to the other servers, no inconsistencies take place. Further tests which automatically fire multiple messages at the same time or added delays as shown above can of course disturb this consistency and lead to a less strict consistency model such as FIFO or cache.