SISTEMAS DISTRIBUIDOS

Práctica final

JUAN MIGUEL PULGAR ESQUIVEL 100451036
100451036@alumnos.uc3m.es

ALBA MARCO UGARTE 100451139
100451139@alumnos.uc3m.es

Mayo de 2023

GRADO EN INGENIERÍA INFORMÁTICA

UC3M

Índice

1. Introducción.

En esta práctica, se ha pedido desarrollar una aplicación de mensajería que permita a los usuarios enviarse mensajes entre ellos. Para ello, contaremos con un servidor concurrente desarrollado en C y un cliente concurrente desarrollado en Python.

2. Diseño realizado.

A continuación se explicará detalladamente el diseño realizado, explicando el funcionamiento y los protocolos realizados para cada una de las funcionalidades:

- **Registro en el sistema**

  Para registrarse en el sistema, el cliente realizará los siguientes pasos:

  a. El cliente pulsa el botón de registrarse en la interfaz y escribe sus datos: Nombre completo, alias y fecha de nacimiento.
  b. El cliente pulsa el botón de registrarse.
  c. Se conecta al servidor de acuerdo al puerto e IP pasados por parámetros.
  d. Se envía la cadena "REGISTER" al servidor, que indica la operación.
  e. Se envía una cadena de caracteres con el nombre completo del usuario. En nuestro caso, hemos supuesto que el nombre de usuario podrá ser como máximo de 256 caracteres.
  f. Se envía una cadena de caracteres con el alias del usuario. Al igual que el nombre, hemos supuesto que podrá ser como máximo de 256 caracteres.
  g. Se envía por último la fecha de nacimiento del usuario como una cadena de caracteres.
  h. Se recibe del servidor un byte que indica el resultado de la operación.

Por su lado, el servidor realiza los siguientes pasos:

a. Se conecta al cliente de acuerdo al puerto pasado por parámetro.
b. Recibe la cadena "REGISTER", que indica la operación.
c. Crea un hilo que llama a la función de registro.
d. En esa función, recibe el nombre completo del cliente.
e. Recibe el alias del cliente.
f. Recibe la fecha de nacimiento.
g. Verifica que el usuario no exista, en cuyo caso enviará un 1 al cliente.
h. Si no existe, almacena el usuario que se quiere registrar y envía un 0 al cliente.
i. Si la operación ha sido un éxito se imprime por pantalla "REGISTER <userName> OK".
j. Si ha fallado imprime "REGISTER <userName> FAIL".

- **Darse de baja del sistema.**

Para darse de baja del sistema, el cliente realizará los siguientes pasos:

a. El cliente pulsa el botón de darse de baja en la interfaz.
b. Si no está registrado, saltará una ventana "pop-up" en la que se indica que el usuario debe estar registrado.
c. Si está registrado, se conecta al servidor, de acuerdo al puerto e IP pasados por parámetro.
d. Se envía la cadena "UNREGISTER" al servidor.
e. Se envía el alias del usuario como cadena de caracteres, de 256 caracteres como máximo.
f. Se recibe del servidor un byte que indica el resultado de la operación.

Por su lado, el servidor realiza los siguientes pasos:

a. Se conecta al cliente de acuerdo al puerto pasado por parámetro.
b. Recibe la cadena "UNREGISTER", que indica la operación.
c. Crea un hilo que llama a la función de registro.
d. En la función, recibe el alias del usuario que se quiere dar de baja.
e. Comprueba que el usuario está registrado.

f. Si lo está, borra sus datos y envía un 0 al cliente.
g. Si no existe, envía un 1 al cliente indicando que la operación ha fallado.
h. Si la operación ha sido un éxito se imprime por pantalla "UNREGISTER <userName> OK".
i. Si ha fallado imprime "UNREGISTER <userName> FAIL".

- **Conexión al sistema.**

Para conectarse al sistema, el cliente realizará los siguientes pasos:

a. El cliente pulsa el botón de conectarse en la interfaz.
b. Si no está registrado, saltará una ventana "pop-up" en la que se indica que el usuario debe estar registrado.
c. Si está registrado, se conecta al servidor, de acuerdo al puerto e IP pasados por parámetro.
d. Se envía la cadena "CONNECT" al servidor.
e. Se envía el alias del usuario como cadena de caracteres, de 256 caracteres como máximo.
f. Se busca un puerto libre y se crea un nuevo socket con este puerto, que se utilizará para recibir y enviar los mensajes desde el servidor.
g. Se crea un hilo que realiza la función de recibir mensajes por este nuevo socket mientras el usuario está conectado.
h. Se envía el puerto utilizado para este socket como cadena de caracteres.
i. Se recibe del servidor un byte que indica el resultado de la operación.

Por su lado, el servidor realiza los siguientes pasos:

a. Se conecta al cliente de acuerdo al puerto pasado por parámetro.
b. Recibe la cadena "CONNECT", que indica la operación.
c. Crea un hilo que llama a la función de conexión.
d. En la función, recibe el alias del usuario que se quiere conectar al servidor.
e. Comprueba que el usuario está registrado.

f. Si no lo está, envía un 1 al cliente para notificar que el usuario no está registrado.
g. Si lo está, recibe el puerto del usuario como una cadena de caracteres.
h. Rellena los campos IP y puerto del usuario.
i. Se cambia su estado a "conectado".
j. Si ya estaba conectado, envía un 2 al cliente.
k. Si no lo estaba, envía al cliente un 0, que indica que la conexión se ha realizado con éxito

- **Desconexión del sistema.**

Para desconectarse del sistema, el cliente realizará los siguientes pasos:

a. El cliente pulsa el botón de desconectarse en la interfaz.
b. Si no está registrado, saltará una ventana "pop-up" en la que se indica que el usuario debe estar registrado.
c. Si está registrado, se conecta al servidor, de acuerdo al puerto e IP pasados por parámetro.
d. Se envía la cadena "DISCONNECT" al servidor.
e. Se envía el alias del usuario como cadena de caracteres, de 256 caracteres como máximo.
f. Se destruye el hilo que se mantiene en un bucle infinito recibiendo mensajes por el socket creado con el comando connect.
g. Se recibe del servidor un byte que indica el resultado de la operación.

Por su lado, el servidor realiza los siguientes pasos:

a. Se conecta al cliente de acuerdo al puerto pasado por parámetro.
b. Recibe la cadena "DISCONNECT", que indica la operación.
c. Crea un hilo que llama a la función de desconexión.
d. En la función, recibe el alias del usuario que se quiere desconectar del servidor.
e. Comprueba que el usuario está registrado.
f. Si no lo está, envía un 1 al cliente para notificar que el usuario no está registrado.

g. Si lo está, recibe el puerto del usuario como una cadena de caracteres.
h. Elimina los campos IP y puerto del cliente en cuestión.
i. Se cambia su estado a "desconectado".
j. Si ya estaba desconectado, envía al cliente un 2.
k. Si no lo estaba, envía al cliente un 0, que indica que se ha realizado la desconexión con éxito.

- **Envío de mensajes.**

Para enviar mensajes, el cliente realizará los siguientes pasos:

a. El cliente escribe el mensaje y el destinatario en la interfaz.
b. El cliente pulsa el botón de enviar mensaje en la interfaz.
c. Se conecta al servidor de acuerdo al puerto e IP pasados por parámetros.
d. Se envía la cadena "SEND" al servidor.
e. Se envía el alias del usuario emisor del mensaje como cadena de caracteres, de 256 caracteres como máximo.
f. Se envía el alias del destinatario del mensaje como cadena de caracteres, de 256 caracteres como máximo.
g. Se envía el mensaje, codificado como cadena de caracteres (de tamaño máximo 256 caracteres).
h. Se recibe del servidor un byte que indica el resultado de la operación.
i. Si ha sido un éxito, recibe el identificador del mensaje.

- **Recepción de mensajes.**

Para recibir mensajes, el cliente realizará los siguientes pasos:

a. Una vez conectado, se crea un hilo que espera en un bucle infinito a recibir mensajes.
b. Por su lado, el servidor envía la cadena "SEND_MESSAGE" al cliente.
c. El cliente recibe una cadena de texto que indica el usuario que envía el mensaje.
d. El cliente recibe el identificador del mensaje.

e. El cliente recibe el mensaje, de máximo 256 caracteres.
f. El servidor cierra la conexión con el cliente.

- **Petición de usuarios conectados.**

Para obtener los usuarios conectados, el cliente realizará los siguientes pasos:

a. El cliente pulsa el botón para obtener los usuarios conectados en la interfaz.
b. Si no está registrado, saltará una ventana "pop-up" en la que se indica que el usuario debe estar registrado.
c. Si está registrado, se conecta al servidor, de acuerdo al puerto e IP pasados por parámetro.
d. Se envía la cadena "CONNECTEDUSERS" al servidor.
e. Se envía el alias del usuario como cadena de caracteres, de 256 caracteres como máximo.
f. Se recibe del servidor un byte que indica el resultado de la operación.
g. Si la operación es un éxito, se recibe del servidor una cadena de caracteres que indica el número de usuarios conectados.
h. Se recibe del servidor el nombre de los usuarios conectados.

Por su lado, el servidor realiza los siguientes pasos:

a. Se conecta al cliente de acuerdo al puerto pasado por parámetro.
b. Recibe la cadena "CONNECTEDUSERS", que indica la operación.
c. Crea un hilo que llama a la función que halla los usuarios conectados.
d. En la función, recibe el alias del usuario que quiere obtener los usuarios conectados.
e. Busca al usuario, si no lo encuentra envía al cliente un 2, que indica que no está registrado.
f. Si lo encuentra pero no está conectado, envía un 1 que indica que no está conectado.
g. Si está conectado, envía un 0.
h. Envía al cliente el número de usuarios conectados.

i. Envía cada cliente conectado como una cadena de caracteres al cliente.

- **Servidor web**

  Por último, hemos creado un servidor web en Python que sirve para formatear los mensajes enviados por el cliente. Así, para cualquier mensaje enviado, antes de enviarlo al servidor principal, se envía al servidor web, que transforma el mensaje dejando únicamente un espacio entre las palabras. Para ello realiza los siguientes pasos:

  a. Se conecta con el cliente en el puerto 5000.
  b. Recibe el mensaje sin formatear.
  c. Transforma el mensaje, dejando solo un espacio entre cada palabra.
  d. Envía de vuelta el mensaje al cliente.
  e. Cierra la conexión.

3. Consideraciones para la corrección.

En este apartado, se muestran consideraciones que hemos considerado necesarias para la corrección de la práctica, es decir, decisiones de diseño que hemos tomado.

- En primer lugar, al enviar mensajes, hemos hecho que si el usuario se conecta y tiene algún mensaje en cola, le envíe todos los mensajes y luego vacíe el array de mensajes pendientes, es decir, no envía un mensaje y lo borra del array, sino que libera y borra el contenido del array después de enviar todos los mensajes. Pensamos que es más sencillo de implementar y más rápido, por lo que hemos decidido realizarlo de este modo ya que no afecta al funcionamiento de la aplicación.

- Por otro lado, hemos supuesto que si el usuario que envía el mensaje no está conectado cuando el otro usuario recibe el mensaje, no se le envía el mensaje de confirmación que indica que el otro usuario ha recibido el mensaje. En el enunciado no se explica este caso, por lo que lo hemos implementado de esta manera.

- El puerto del servidor web siempre es el 5000. No se indica nada en el enunciado, por lo que hemos considerado que lo más sencillo y rápido es asignarle un puerto cualquiera para que funcione correctamente.

- El usuario con el que se está registrado se borra del cliente si se intenta registrar con otro usuario cuyo nombre ya existe. Esto se debe a que cuando falla el registro se pone la variable client._alias a None para que otras funciones no tengan problemas y puedan saltar correctamente las ventanas pop-up.

- Si un usuario se vuelve a registrar, el usuario con el que estaba registrado se queda guardado en el servidor. Esto se debe a que no existe un botón de iniciar sesión, por lo que no podemos comprobar

4. Compilación del proyecto.

Para compilar y utilizar la aplicación, se deben realizar los siguientes pasos:

1. Extraer los ficheros del proyecto en un directorio.
2. Una vez extraídos, utilizar en la consola el comando cd para situarse en el directorio donde se ha extraído.
3. Ejecutar el comando make para generar el ejecutable del servidor.
4. Ejecutar el servidor en el puerto deseado con el comando ./servidor <puerto>
5. Ejecutar el servidor web de python con el comando python3 ./servidor_web.py
6. Ejecutar tantos clientes como se desee con el comando python3 ./client.py -s <ip> -p <puerto>
7. Se abrirá una interfaz en la que se podrá probar el programa.

5. Pruebas realizadas.

Las pruebas se han realizado una vez realizado el servidor web, por lo que algunas pruebas de la llamada “send” contienen mensajes que se deben formatear.

Pruebas para registro:

- **Prueba de registro de un usuario:** Se observa que se muestra por pantalla el siguiente mensaje indicando que todo ha salido correctamente: "c> REGISTER Tex s> REGISTER OK".

- **Prueba de intento de registro duplicado:** Se observa que se muestra por pantalla el siguiente mensaje indicando que el usuario ya está registrado: "c> REGISTER Tex s> USERNAME IN USE".

- **Prueba de registro simultáneo con el mismo alias desde diferentes clientes:** Se observa por pantalla del segundo cliente indicando que se han registrado dos usuarios iguales: "c> REGISTER Tex s> USERNAME IN USE".

- **Prueba de registro de varios usuarios desde diferentes clientes:** Se observa en todas las pantallas de todos los clientes el siguiente mensaje indicando que se han registrado correctamente: "c> REGISTER Nombre s> REGISTER OK".

Pruebas para desregistrar:

- **Prueba de registro y desregistro de un usuario**: Se observa que se muestra el mensaje del registro correctamente y el mensaje de desregistrar de esta manera: “c> UNREGISTER Alba , c> UNREGISTER Alba”.

- **Prueba de registro, desregistro y segundo desregistro de un usuario:** Aparece una pantalla de usuario no registrado y no se muestra ningún mensaje.

- **Prueba de registro de un usuario, registro de otro usuario desde otro cliente y desregistro de ambos usuarios**: Se observan los mensajes de registro correctamente y aparecen los dos mensajes de desregistro en los dos clientes.

- **Prueba de registro de un primer cliente, registro del segundo cliente y desregistro de este segundo:** Se observan los mensajes de registro correctamente y aparece un mensaje de desregistrado del segundo cliente de está manera: “c> REGISTER Tex c> REGISTER Alba c> UNREGISTER Alba c> REGISTER Tex c> REGISTER OK c> REGISTER OK c> UNREGISTER OK”

Pruebas para connect y disconnect: Las funciones de connect y disconnect se prueban sobre todo en las otras funciones ya que dependen de ellas.

- **Prueba de desconexión simultánea de varios usuarios:** Se muestran los siguientes mensajes por pantalla indicando que la conexión y la desconexión ha salido correctamente en los dos clientes: s> "CONNECT OK s> DISCONNECT OK c> CONNECT Tex c> DISCONNECT Tex s> CONNECT OK s> DISCONNECT OK c> CONNECT Tex c> DISCONNECT Tex".

- **Prueba de conexión y doble desconexión:** Se muestran los siguientes mensajes por pantalla indicando que no se puede desconectar a un usuario dos veces: s> DISCONNECT OK s> DISCONNECT FAIL / USER NOT CONNECTED c> DISCONNECT Tex c> DISCONNECT Tex.

- **Prueba de conexión y desconexión de un usuario desregistrado:** Aparece una pantalla de usuario no registrado y no te permite conectarlo (Lo mismo pasa con la desconexión).

- **Prueba de conexión , desconexión y envío de mensaje al usuario desconectado para comprobar que realmente se ha desconectado:** La conexión y desconexión funcionan correctamente como en anteriores pruebas. El emisor envía el mensaje , pero el receptor desconectado no recibe nada, además el emisor no recibe el ack porque el receptor no lo ha recibido.

Pruebas para send:

- **Prueba de envío de mensaje a un usuario conectado:** se muestra por pantalla el envío y la recepción correcta del mensaje además al emisor le llega una confirmación de que su mensaje se ha recibido: c> SEND Te Hola Te! s> SEND OK - MESSAGE 2 s> SEND MESSAGE 2 OK s> MESSAGE 2 FROM Alba Hola Te! END

- **Prueba de envío de mensajes a un usuario conectado**: se muestra por pantalla los mensajes recibidos y enviados: c> SEND Te Hola Te! s> SEND OK - MESSAGE 2 s> SEND MESSAGE 2 OK c> SEND Te ¿Que tal? s> SEND OK - MESSAGE 3 s> SEND MESSAGE 3 OK s> MESSAGE 2 FROM Alba Hola Te! END s> MESSAGE 3 FROM Alba ¿Que tal? END

- **Prueba de envío de mensaje a un usuario conectado desde varios clientes:** Se muestra por pantalla como llegan mensajes de distintos clientes: c> SEND a holaaa que tal estas s> SEND OK - MESSAGE 1 s> SEND MESSAGE 1 OK c> SEND a hola a yo tambien estoy s> SEND OK - MESSAGE 2 s> SEND MESSAGE 2 OK s> MESSAGE 1 FROM b holaaa que tal estas END s> MESSAGE 2 FROM c hola a yo tambien estoy END

- **Prueba de envío de mensaje a un usuario no conectado**: se muestra por pantalla que si el usuario no está conectado no lo recibe el emisor, por lo que no llega una confirmación de recibido : c> SEND a holaaa que tal estas s> SEND OK - MESSAGE 3

- **Prueba de envío de mensaje a un usuario no conectado y luego conectar a dicho usuario**: Con la prueba anterior, al conectar a le llegan los mensajes y el emisor recibe el ack: "a" s> MESSAGE 3 FROM b holaaa que tal estas END y desde "b" s> SEND MESSAGE 3 OK

- **Prueba de envío de varios mensajes a un usuario no conectado y luego conectar a dicho usuario:** Se muestra por pantalla que primero se envían los mensajes pero los emisores no reciben el ACK que indica que el mensaje se ha recibido. A continuación, se ve como al usuario le llegan todos los mensajes en orden y a cada usuario le llega su confirmación: c> SEND a como te va la vida a? s> SEND OK - MESSAGE 0 c> SEND a a mi me va todo bien s> SEND OK - MESSAGE 1 c> SEND a ¿quedamos luego? s> SEND OK - MESSAGE 2. Una vez conectado "a" - s> MESSAGE 0 FROM b como te va la vida a? END s> SEND MESSAGE 0 OK s> MESSAGE 1 FROM b a mi me va todo bien END s> SEND MESSAGE 1 OK s> MESSAGE 2 FROM b ¿quedamos luego? END s> SEND MESSAGE 2 OK

- **Prueba de envío de mensajes a un usuario no conectado desde varios clientes y luego conectar a dicho usuario**: Se muestra por pantalla como al conectarse le llegan todos los mensajes y a los emisores los ack correspondientes : c> SEND a holaaa que tal estas s> SEND OK - MESSAGE 1 c> SEND a hola a yo tambien estoy s> SEND OK - MESSAGE 2 s> MESSAGE 1 FROM b holaaa que tal estas END s> SEND MESSAGE 1 OK s> MESSAGE 2 FROM c hola a yo tambien estoy END s> SEND MESSAGE 2 OK

- **Prueba de envío de mensaje desde un usuario desconectado:** Se muestra por pantalla que el envío falla si el emisor no está conectado : c> SEND Te Hola Te! s> SEND FAIL.

<u>Pruebas para usuarios conectados:</u>

- **Prueba de creación de usuario sin conexión y consulta de usuarios conectados**: Se muestra por pantalla un mensaje de usuario no conectado indicando que antes de ver los usuarios conectados se debe estar conectado: "c> CONNECTEDUSERS s> CONNECTED USERS FAIL / USER IS NOT CONNECTED".
- **Prueba de creación y conexión de múltiples usuarios desde distintos clientes, y verificación de la función connectedUsers:** Se muestra por pantalla el mensaje: CONNECTED USERS (2 users connected) OK - Tex, a.
- **Prueba de creación y conexión de múltiples usuarios desde clientes distintos, uso de la función connectedUsers, desconexión de un usuario y comprobación de connectedUsers tras la desconexión:** Se muestran por pantalla los mensajes:
  s> CONNECTED USERS (3 users connected) OK - a, b, c
  c> DISCONNECT b
  s> CONNECTED USERS (2 users connected) OK - a, c

6. Conclusión.

En esta práctica hemos aprendido a conectar un cliente y servidor en distintos lenguajes de programación utilizando sockets. Para ello, hemos aprendido a utilizar un estándar para poder conectar los mensajes entre los distintos lenguajes de programación. Consideramos que la práctica ha servido de mucho para aprender a manejar programas concurrentes y el envío de mensajes utilizando sockets.