

Introducción

Este trabajo practico esta dividido en dos grandes secciones. La sección del cliente, encargada de enviar los comandos, esperar la respuesta y mostrarlos por salida estándar, y la sección del servidor, que mantiene las conexiones con cada cliente, recibe sus respectivos comandos y devuelve una respuesta.

Para una correcta abstraccion del cliente con el servidor e viceversa, se utilizo el patrón de diseño *Proxy*. Por ese motivo, existe un servidor proxy del lado del cliente y un cliente proxy del lado del servidor, cada uno abstrayendo al cliente y servidor respectivamente de la lógica relacionada al envío y recepción de comandos y respuestas.

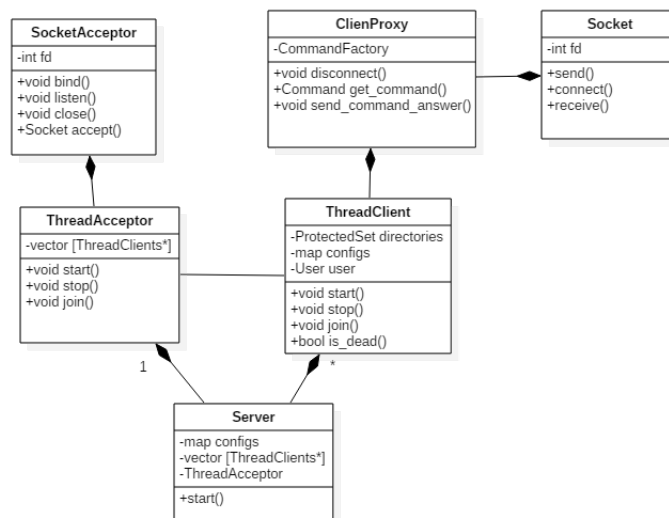


Figure 1: Composición general del servidor

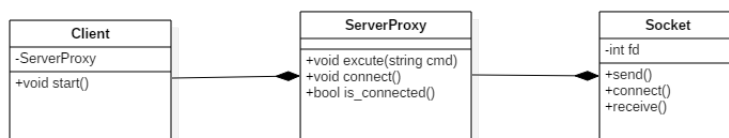


Figure 2: Composición general del cliente

Hilos

En el programa es posible identificar tres tipos distintos de hilos. En primer lugar tenemos el hilo principal, que inicializa una instancia server e inicia un hilo aceptador. Una vez finalizado esta secuencia, el hilo principal procede a esperar indefinidamente que se ingrese por entrada estándar el comando de salida. El hilo aceptador mientras tanto acepta los nuevos clientes y lanza un nuevo hilo para mantener la comunicación en paralelo mientras espera por la conexión de nuevo clientes.

Hilo Aceptador

Este hilo, como mencionado previamente, se dedica a aceptar los nuevos clientes e instanciar un nuevo hilo para mantener la conexión en paralelo al proceso de aceptar nuevas conexiones. Al aceptar un nuevo cliente, recorre el vector con todos los clientes aceptados previamente verificando si la conexión sigue establecida o no, procediendo a cerrar e liberar los recursos si la conexión fue cortada.

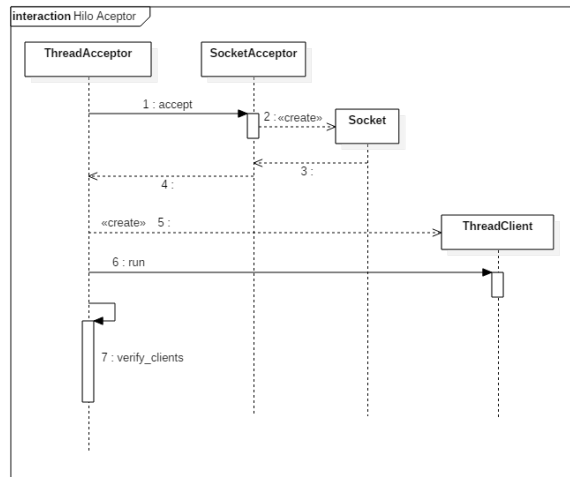


Figure 3: Secuencia hilo aceptador

Hilo Cliente

Este hilo mantiene una conexión con uno de los clientes conectados. Mediante un cliente proxy (para mantener la abstracción) recibe los comandos del cliente, los ejecuta y devuelve la respuesta al comando. Internamente posee una referencia a una estructura del tipo *Map* con las configuraciones del servidor además de una referencia al set *thread safe* con los directorios "creados" por el cliente.

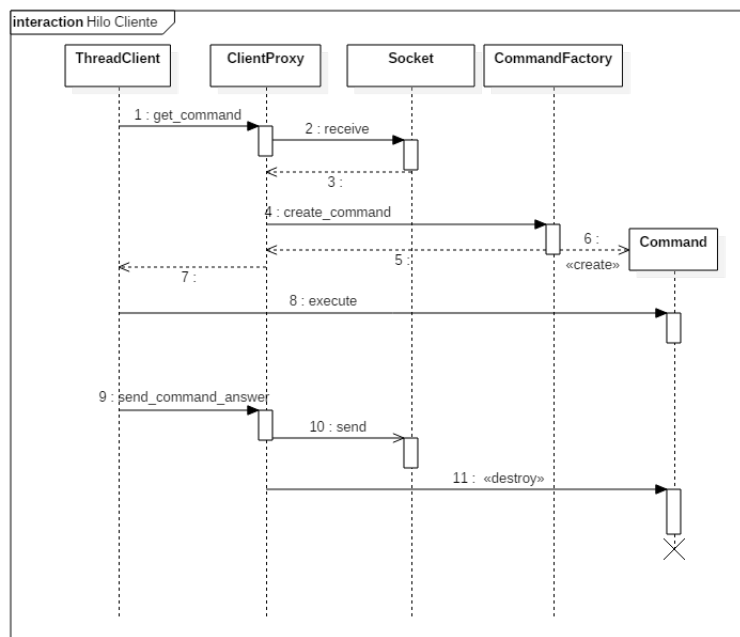


Figure 4: Secuencia hilo cliente

Comandos

Los comandos se ejecutan de manera polimórfica. Existe una clase abstracta **Commnad** con los métodos **execute** y el metodo **send_answer**. El metodo **execute** recibe una referencia al *Map* con las configuraciones, una referencia al set de directorios y una referencia a un objeto del tipo **User**, usado para modelar el estado de la conexion (*Logged*, *NotLogged*, *LoggedOut*) del usuario.

El metodo **send_answer** recibe el socket que establece la conexion con el cliente y este se encarga de enviar la respuesta apropiada.

Estado de la conexion

Uno de las restricciones de servidor es que el cliente no pueda realizar ninguna operación a menos que este estuviera iniciado sesión correctamente. Con el objetivo de encapsular el guardado los datos de usuario y contraseña ingresados por el cliente y la determinación de su validez, se creo la clase **User**.

En un objeto de la clase **User** es posible almacenar la contraseña y usuario ingresado por el cliente, verificar si es coinciden y obtener el estado de conexion del usuario en ese momento.