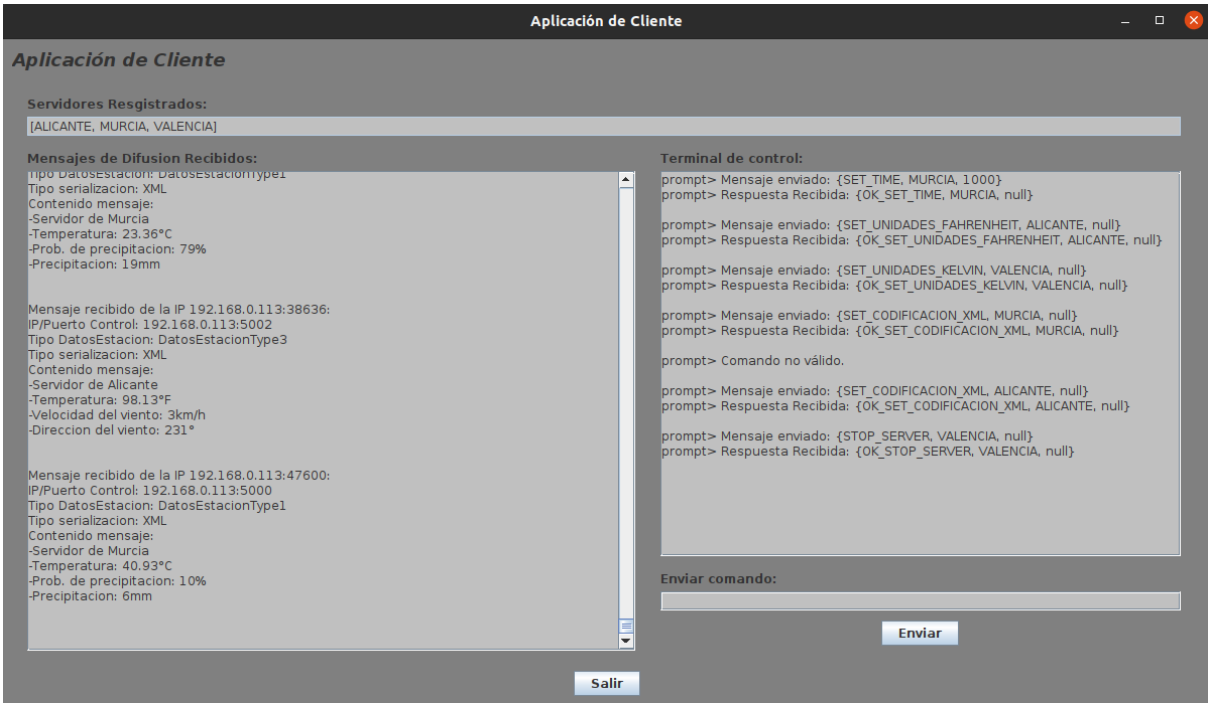
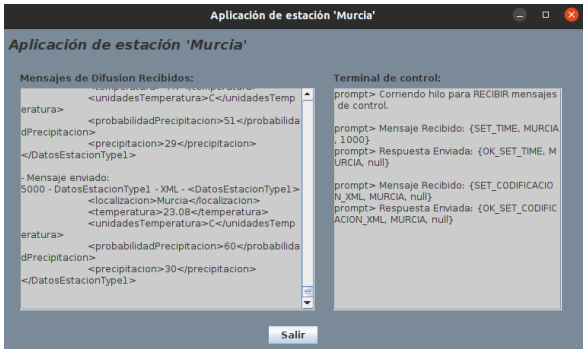
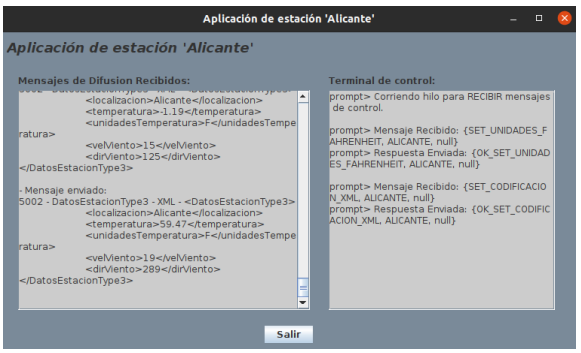
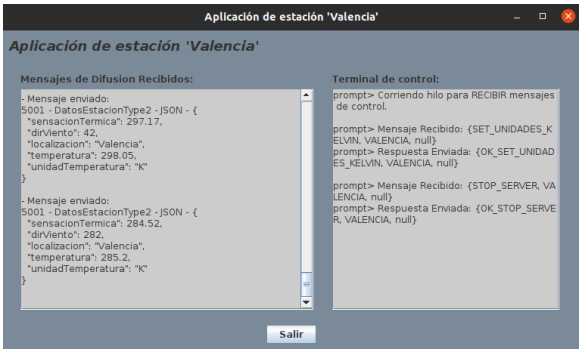
Programación para las Comunicaciones
Práctica 2 - Gestión de datos meteorológicos





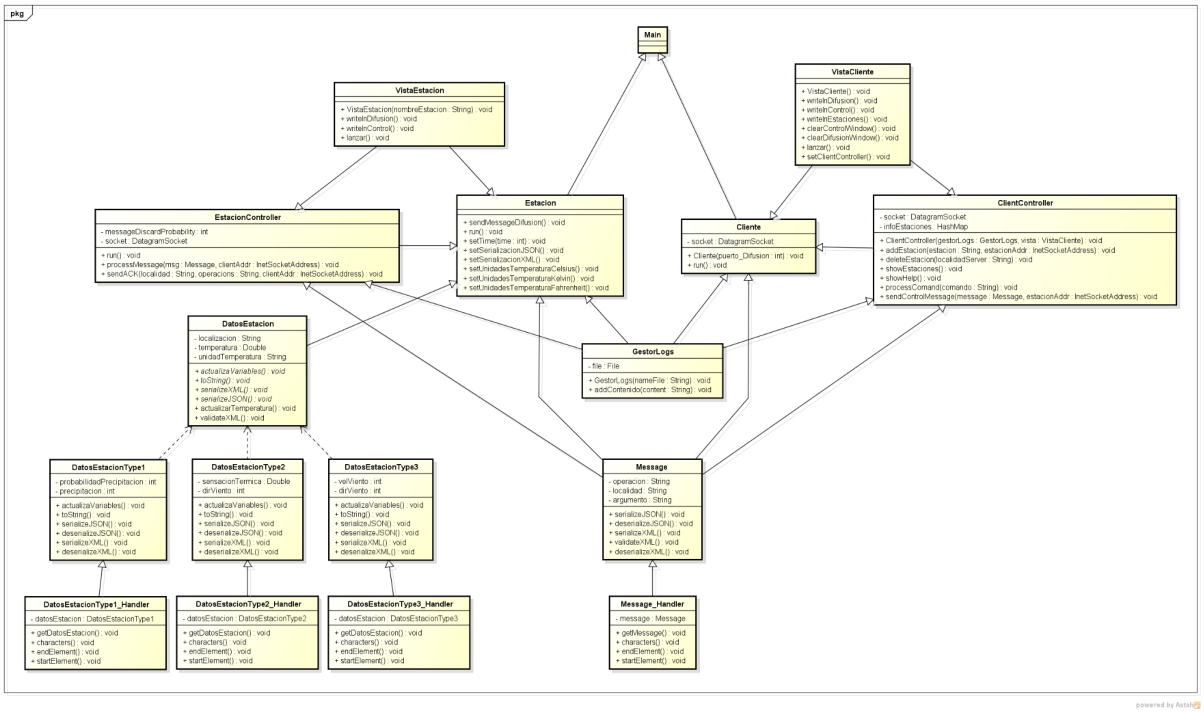




Pedro Nicolas Gomariz
pedro.nicolasg@um.es
48753907W

1.- Descripción sobre el diseño e implementación y listado de clases incluyendo los diagramas de clases más relevantes.

(hacer zoom para ver)



Las clases más importantes son:

-Estacion → Implementa la parte de la estación relacionada con el envío de los mensajes de difusión y lleva el control de los datos meteorológicos que envía en dichos mensajes.

-EstacionController → Implementa la parte de la estación relacionada con la recepción de mensajes de control y envío de sus correspondientes ACKs.

-Cliente → Implementa la parte del cliente relacionada con la recepción de los mensajes de difusión enviados por las estaciones.

-ClientController → Implementa la parte del cliente relacionada con el procesamiento de comandos por parte del cliente y en función de estos, crear y enviar los mensajes de control correspondientes a las estaciones indicadas.

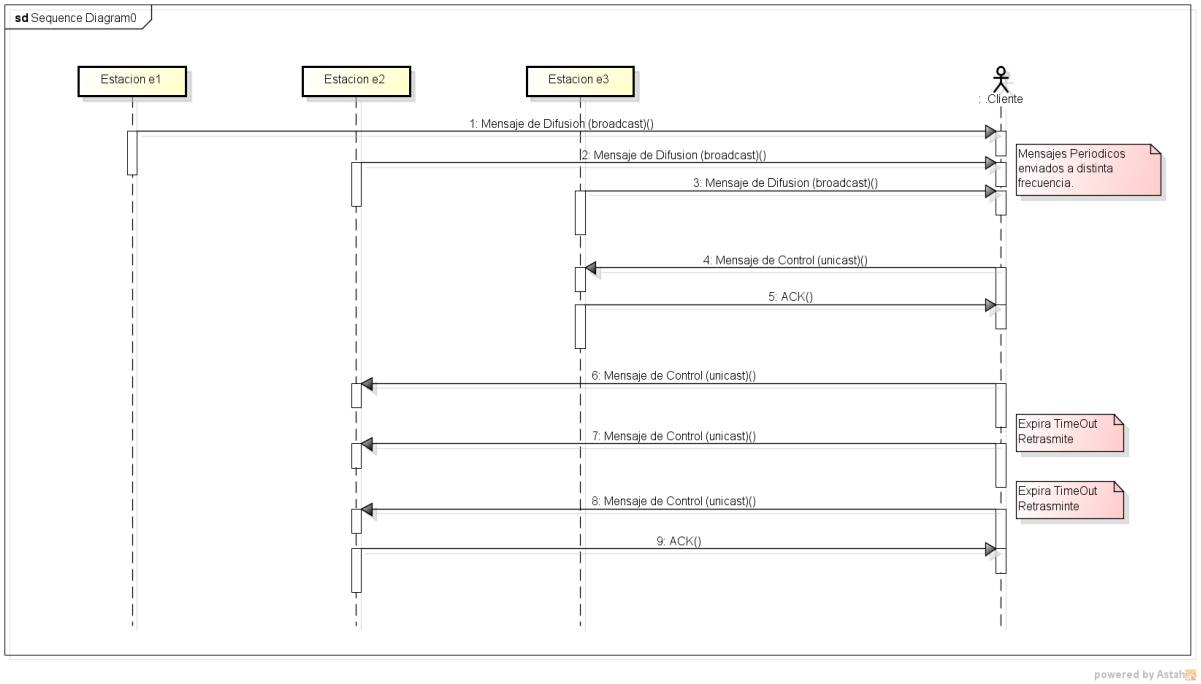
-DatosEstacion → Pueden ser de distintos tipos (1, 2 o 3) y se utilizan para almacenar y gestionar las variables meteorológicas que envía una estación.

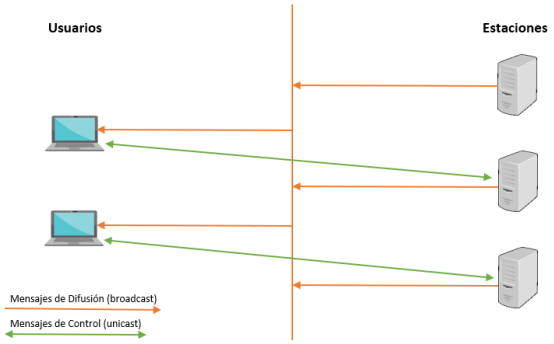
-DatosEstacionType*_Handler → Implementan el manejador para realizar la deserialización de los mensajes XML.

-Message → Se utiliza para almacenar la información que se puede enviar en un mensaje de control.

2.- Descripción del protocolo mediante diagramas de secuencia.

(hacer zoom para ver)





El protocolo consiste en el envío periodico de mensajes de difusión broadcast por parte de las estaciones a los clientes, indicándose información acerca de datos meteorológicos. Puntualmente los clientes pueden enviar mensajes de control unicast a alguna de las estaciones para cambiar algunos de sus parámetros, como la serialización/deserialización, la frecuencia de envío de los mensajes, parar el envío de mensajes por parte de esta o para cambiar las unidades de la variable de temperatura.

3.- Formato de los mensajes.

Los campos de los mensajes en mi caso son separados por el separador ‘ - ’.

Mensajes de difusión:

PUERTO CONTROL	TIPO DATOS ESTACIÓN	TIPO CODIFICACIÓN	CONTENIDO DEL MENSAJE

Mensajes de control:

TIPO CODIFICACIÓN	CONTENIDO DEL MENSAJE

4.- Arquitectura de estaciones-clientes y modelo de gestión de E/S.

En mi caso la implementación de la estación se basa en una arquitectura multi hilo, por un lado tenemos un hilo el cual se encarga de crear y enviar los mensajes de difusión de esta, y por otro lado tenemos otro hilo encargado de la parte de control, es decir de recibir los mensajes de control, procesarlos, y enviar su respuesta correspondiente.

Por otro lado, la arquitectura utilizada para los clientes también será multihilo, uno se encargará de realizar todas las tareas relacionadas con la parte de difusión (recibir mensajes de difusión, deserializar, mostrar por pantalla, etc), y el otro hilo será el encargado de ejecutar la función *processComand*(), esta función se ejecuta cuando hacemos clic en el botón enviar o intro en el cuadro de texto de la interfaz gráfica y ejecuta un *ActionListener* que invoca dicha función. Esta función será la encargada de realizar todo lo oportuno para los mensajes de control (procesamiento del comando, creación mensajes de control, gestión de reenvíos y recepción de confirmaciones).

En mi caso la E/S de los socket es bloqueante ya que cuando ejecutamos la orden *socket.receive()* el hilo que la ejecuta se queda "suspendido" esperando a que haya datos en el socket para leer, es por esto por que para estar leyendo mensajes de control y de difusión al mismo tiempo sin que el programa quede "paralizado" he tenido que utilizar una arquitectura multi hilo.

5.- Mejoras.

- Gestión de las pérdidas en los mensajes de control, y su posible solución mediante reenvíos → Para gestionar esto se ha implementado un mecanismo en *EstacionController* que dada una probabilidad indicada descarta tramas de control que le llegan porque podrían estar corruptas, en ese caso no le responde al clientes y cuando en este expira un timeout establecido sin recibir una confirmación al mensaje, lo vuelve a enviar. Así hasta que finalmente recibe la confirmación. En el caso de que se reenvíe un número de veces preestablecido y no se haya recibido confirmación se dejara de intentar, en mi caso el número máximo de intentos es 5 y el timeout es de 1 segundo.

- Uso de distintas tramas de control → Se implementan tramas de control para controlar:

- ➔ Cambiar codificación XML/JSON en tiempo de ejecución.
- ➔ Cambiar frecuencia de transmisión de los mensajes de difusión.
- ➔ Cambiar las unidades de la variable temperatura de las estaciones.
- ➔ Pausar el envío de datos por parte de una estación.
- ➔ Reanudar el envío de datos por parte de una estación.

- Implementación de interfaz gráfica para mostrar la información en las estaciones y clientes → Se muestra un ejemplo de funcionamiento en la portada.