SIMULADOR DE UNA RED DE ESTACIONES METEORÓLOGICAS

Pablo Andrés Rodríguez
pablorodriguez.bb@gmail.com

RESUMEN: *El presente documento presenta todo lo que concierne al diseño y desarrollo del proyecto final de materia de Principios de Computadoras II.*
El proyecto es una aplicación que simula una red de estaciones meteorológicas. Esta aplicación simula un sistema compuesto de tres componentes básicos: estaciones de almacenamiento de datos, estaciones meteorológicas y sensores. Los componentes son creados conjuntamente con el sistema dependiendo de las necesidades del usuario. Dicho software esta completamente implementado en java debido a ciertas restricciones que impone la cátedra.

PALABRAS CLAVE: simulador, java, redes

1 INTRODUCCIÓN

Un simulador es una herramienta muy utilizada actualmente en muchas áreas de trabajo. Esto se debe a que ayuda en el diseño de sistemas, prueba de modelos, recreación y evaluación de eventos pasados, y otra gran cantidad de soluciones sin la necesidad de interacción física con el medio.

La simulación de una red de estaciones meteorológicas es interesante no solo desde el punto de vista de como se comporta la red, la recopilación de datos desde distintos medios y el posterior tratado de estos datos; sino también en la recreación de sistemas reales a base de datos almacenados.

Este proyecto no abarca todo el espectro de posibilidades, pero cumple con algunos requisitos mínimos.

2 CARACTERÍSTICAS GENERALES

El proyecto de software simula una red de estaciones meteorológicas, la cual se conforma de tres componentes básico:

- Estación Base: estación encargada de manejar la red.
- Estación Meteorológica: estación que posee sensores. Es la encargada de manejar a estos.
- Sensor: elemento base del sistema del cual se obtienen las mediciones.

La estación base cuenta con 4 puertos de red para establecer comunicación con estaciones Meteorológicas

Cada estación meteorológica es configurable a la necesidades del usuario, es decir, cada estación presenta 4 interfaces estándar para conectar sensores, 3 puertos de red para establecer comunicación con otras estaciones y un puerto de red para la estación padre.

Los sensores se distribuyen en 4 tipos:

- Sensor de temperatura: indica la temperatura en ºC.
- Sensor de humedad: indica la humedad relativa porcentual.
- Sensor de viento: indica tanto la velocidad del viento en km/h, como la dirección de éste.
- Pluviómetro: indica la lluvia caída en los últimos 5 minutos, medida en mm.

En el sistema existe una única estación base, la cual es la raíz de toda la red. Ésta existe siempre y es desde ella de donde se maneja toda la red, ya sea agregar o eliminar estaciones, agregar o eliminar sensores o evaluación de estadísticas

Periódicamente la estación base interroga a cada estación y registra los datos recibidos en una base de información en RAM.

Al ser interrogada, una estación meteorológica, no solo obtiene las mediciones de sus sensores, sino también crea registros locales con valores medios, máximos y mínimos

3 DESCRIPCIÓN DEL SISTEMA

La aplicación esta conformada de pocas clases que intentan copiar lo mas fielmente la realidad.

La estructura del sistema es una red estrella conformada de una estación base, estaciones meteorológicas y sensores. El nodo principal de la red es la estación base, de la cual puede existir una y solo una.

La estación base no posee sensores, pero es la encargada de manejar todo el sistema. Una funcionalidad básica de la estación es la de crear o eliminar nodos, ya sean estaciones meteorológicas o sensores. Además, es la encargada de hacer los pedidos de datos, es decir, ningún nodo envía información de manera espontánea sino que cuando se le dice a la estación base que hay que actualizar el sistema, esta es la encargada de recorrer la red actualizando los nodos y recopilando la información.

La información recopilada es guardada en ram para la posterior presentación de los datos. Esta es enviada desde las estaciones meteorológicas a la estación base a través de un paquete de datos, el cual contiene la

información de la estación, sus sensores, las mediciones y la hora en que obtuvieron éstas.

Cuando se interroga a cada estación meteorológica esta recopila la información de todos sus sensores y envía esta información a través de un paquete de datos. Además, la estación recopila internamente resúmenes estadísticos de las mediciones: valores medios, máximos y mínimos. Esta información es guardada en disco en un archivo XML, el cual es accedido y modificado en cada actualización

El manejo de resúmenes se hace a través de archivos XML gracias a una biblioteca de Apache, Apache Commons Configuration. Esta biblioteca esta pensada para trabajar con configuraciones en distintos formatos, uno de ellos XML. Por su facilidad de uso se eligió trabajar con esta biblioteca, pero no para configuraciones sino para guardar los resúmenes con las estadísticas de las estaciones, como se dijo mas arriba.

La estación base también tiene la funcionalidad de recuperar estos resúmenes y guardarlos en su red. Como la aplicación simplemente simula una red, las estaciones meteorológicas guardan sus resúmenes en el disco local. Debido a esto es totalmente innecesario hacer la búsqueda y copia de cada resumen a una carpeta local. En caso de tener las estaciones meteorológica sin acceso compartido al disco local, como seria en la realidad, la estación base debería hacer una copia de de los resúmenes, lo que no implicaría un trabajo engorroso ya que lo que se hace actualmente es el pedido de la dirección del resumen y faltaría solo hacer la copia.

Además de las estaciones, se crearon varias clases de sensores, pero que siguen una interfaz común. Debido a la gran variedad de sensores posibles, no es fácil crear una interfaz genérica para que todos sigan. Esto se debe principalmente al tipo de medición La gran mayoría de los sensores tiene una parte del sensado en valores numéricos, pero que pasa con los que no. Para solucionar este problema se eligió que los datos de todos los sensores sea de tipo cadena de caracteres(String). El beneficio de esto es que cualquier sensor actual y cualquier otro que se quiera agregar, puede tener el mismo formato. Aun así, se supone que todo sensor tiene una parte numérica, debiendo ser esta la primera parte de la medición Para hacer uso del valor numérico, es necesario parcear este string.

Todo sensor tiene 2 métodos a destacar: getMedicion() y setMedicion(). Este último no tiene mucho sentido en un sistema real, debido a que a un sensor no se le puede dar el valor de la medición. Como la aplicación es solo una simulación sin interacción física con el medio, las mediciones que tiene cada sensor son impuestas de cierta manera. Actualmente el valor de la medición se da de manera pseudo-aleatoria cuando se pide el valor sensado. Esto no tiene porque ser así, pudiendo (en un futuro) cargar valores de un archivo o leerlos de algún puerto físico.

En el sistema existen solo 4 tipos de sensores distintos. Tres de ellos, como la mayoría de los sensores, manejan variables numéricas y no es de mayor problema. La situación se podia haber llegado a complicar un poco con el sensor de viento, debido a que este posee dos valores: la velocidad y la dirección del viento. Por trabajar con las mediciones en formato de cadena de caracteres, el procesamiento de estos datos no es mas complicado que el resto. El formato de este tipo de sensor se divide en dos partes, separadas por un caracter en blanco: “<velocidad> <dirección>”. El parceo de esta variable se hace a través del espacio en blanco que contiene para separar ambas mediciones Solo la primer parte se utiliza para evaluar estadísticas, aún así el valor completo de la medición es lo que se guarda en los registros. Esto se hace por que de otra manera se estarían perdiendo datos importantes y las estadísticas serian completamente obsoletas.

A pesar de ser la parte más importante de la aplicación, estas clases no hacen nada por si solas, y se necesita de un sector del programa que se dedique al arranque y configuración de la red. Esta función la cumple la clase *Main*, junto con *SimNoGui* o *MainWindow*, según la interfaz de usuario deseada. La clase principal (*Main*) solo se encarga de borrar todos los archivos de sesiones pasadas y redirigir la salida de error a un archivo de log. En este archivo no solo queda notificación de los errores, sino también todo tipo de información enviada por los loggers. Esta información se puede usar para debbugear el sistema si llegara a ser necesario.

Como se dijo anteriormente, existen dos interfaces de usuario, una desarrollada completamente desde la terminal y otra desde una ventana gráfica. Ambos sistemas son completamente funcionales. La principal diferencia, ademas del aspecto, es como se maneja la interacción con el usuario.

La aplicación desde la terminal es completamente guiada a través de menúes y preguntas. El usuario ingresa por teclado las posibles acciones a tomar y se trata de hacer las preguntas necesarias como para llegar a un resultado satisfactorio para este. Sobre esta interfaz se puede:

- Crear una nueva red
- Cargar un ejemplo preconfigurado
- Modificar la red actualmente en uso
- Borrar los resúmenes

Un problema que existe en esta interfaz es que no existe modo de parar la simulación una vez comenzada. Es decir, se le pregunta al usuario cuantas veces desea actualizar el sistema y luego se inicia la simulación

La aplicación desde el entorno gráfico es mocho mas amigable y cómoda. Ésta también es completamente guiada a través de preguntas, pero al poder tomar como eventos toda interacción con el usuario, el problema de parar la simulación acá no existe.

La interfaz es simple y limpia a la vista. En la *Figura 1* se puede apreciar una impresión de pantalla. Esta consta de:

- Panel con un visor del estado de la red
- Panel que muestra la red, en forma de árbol de directorios
- Menú con todas las acciones posibles a tomar
- Botones de acceso rápido a las acciones mas usadas del menú
- Botón de arranque y parada de la simulación
- Texto de información al fondo de la ventana

Sobre la izquierda de la ventana se encuentra el panel que muestra la estructura de la red. Junto a este, el panel donde se vuelca la información del sistema.

Desde el menú Resúmenes se puede acceder a los resúmenes y visualizarlos en un visor interno del programa. Al seleccionar la función *Ver resumen* se abrirá un cuadro de dialogo para seleccionar el resumen a ver y luego se mostrara en un visor como el que se presenta en la *Figura 2*.

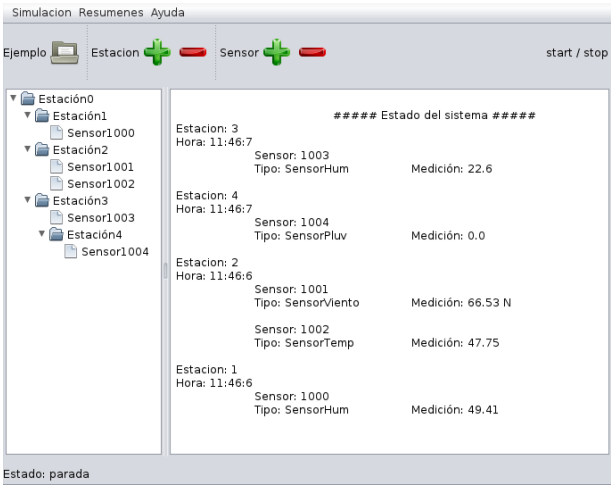

Figura 1. Impresión de pantalla principal

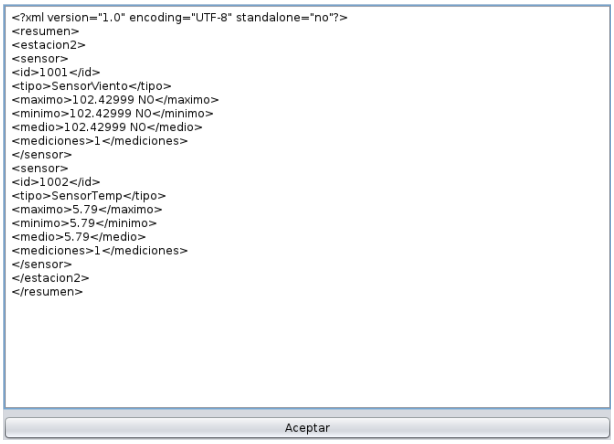

Figura 2. Impresión de pantalla del Visor de Resúmenes

4 POSIBLES MEJORAS

De seguro existen varias mejoras que se le pueden hacer a la aplicación debido a que esta versión contiene una estructura mínima para que el sistema funcione correctamente. Esto no quiere decir que no se pueda minimizar el código, sino que cumple con la funcionalidad mínima

Una posible mejora que encontraría realmente útil, es la de recreación de sistemas. Esto se refiere a la posibilidad de crear una red y que los valores de los sensores en el tiempo sean cargados de un archivo.

En la actualidad existen estaciones meteorológicas que llevan registro de acceso publico de todas sus estaciones. Estos registros podrían ser cargados en el simulador y eventualmente poder recrear cierto estado.

El tiempo entre actualizaciones del sistema esta fijado en una constante *REFRESH_TIME* dentro de la clase que inicia la interfaz, es decir, en MainWindow y en SimNoGui. Este valor podría seterse de manera externa en la interfaz gráfica

Otra mejora que empece en un principio es cambiar el modo en que se actualiza el sistema. Actualmente los cambios en las mediciones se hacen cuando un timer avisa que paso el tiempo *REFRESH_TIME*. El cambio radicaría en sacar este timer y reemplazarlo por un reloj independiente, el cual levante el evento del timer, pero con la posibilidad de cambiar la velocidad de la simulación

Desde el punto de vista del entorno gráfico se podría agregar la opción de modificación de la red a través del panel que muestra la estructura de ésta y de esta manera facilitar el uso de la herramienta.

5 CONCLUSIÓNES

Este proyecto cumplió con el propósito inicial de presentar al alumno frente a un proyecto de pequeña envergadura así como también de interiorizar los conocimiento adquiridos de la materia.

Debido a complicaciones personales el tiempo de desarrollo fue corto, estando obligado a realizar al menos el proyecto mínimo en tiempo y forma. Por este motivo el proyecto no cumplió mis expectativas iniciales, pero aún así sirvió de referencia para futuros proyectos, así como también de experiencia en desarrollo de software.

6 AGRADECIMIENTOS

Un especial agradecimiento a Manuel Argüelles quien me ayudo a solucionar unos problemas a ultima hora.

7 REFERENCIAS

- http://commons.apache.org/configuration/
- Mark Allen Weiss, *"Estructura de datos en java"*, Addison-Wesley, 2000..