Sistemas Distribuidos

Control Estaciones Meteorológicas Sockets y RMI

Práctica no guiada

Pedro Giménez Aldeguer **15419933C**

4-11-2018

pga44@alu.ua.es Turno lunes: 09:00-11:00

Contenido:

1. Introducción:

El objetivo de esta práctica es que los estudiantes extiendan y afiancen sus conocimientos sobre el uso de tecnologías de comunicación básicas como **sockets y RMI**, estudiadas durante las sesiones de teoría.

Los **sockets** de Internet constituyen el mecanismo para la entrega de paquetes de datos provenientes de la tarjeta de red a los procesos o hilos apropiados. Un *socket* queda definido por un par de direcciones IP local y remota, un protocolo de transporte y un par de números de puerto local y remoto.

Son la base de las comunicaciones entre computadores y suponen el punto de enlace básico entre nuestra aplicación e Internet, utilizando muy pocos recursos de red. Como contrapartida, las aplicaciones que utilizan sockets como mecanismo de comunicación deben interpretar los mensajes que intercambian entre sí, es decir, deben establecer un protocolo o acuerdo de comunicación entre ellos. Esta necesidad implica un fuerte acoplamiento entre las aplicaciones distribuidas que utilizan sockets como mecanismo de comunicación, ya que todas las partes deben conocer de antemano la estructura de los mensajes que serán intercambiados y codificarlo en su programación.

RMI (*Java Remote Method Invocation*) es un mecanismo ofrecido por Java para invocar un método de manera remota. Forma parte del entorno estándar de ejecución de Java y proporciona un mecanismo simple para la comunicación de servidores en aplicaciones distribuidas basadas exclusivamente en Java.

Es similar a los sockets, aunque supone un nivel de abstracción superior al trabajar a nivel de objeto Java, con las ventajas e inconvenientes que eso conlleva, ya que se transfieren objetos remotos al cliente, aunque los métodos se ejecutan en el servidor. Esto hace que la comunicación sea más costosa, pero aumenta la productividad en el desarrollo de aplicaciones distribuidas, ya que los clientes de estos objetos solo deben invocar los métodos y esperar un resultado de tipo conocido.

Esta práctica establece el marco inicial de un desarrollo que se ampliará durante el cuatrimestre y que permitirá al estudiante crear una aplicación similar a las que se desarrollan hoy en día en los entornos empresariales, poniendo el foco en el uso e integración de distintos paradigmas de comunicación susceptibles de ser utilizados.

2. Componentes:

Indicaré el nombre de los componentes software desarrollados y una descripción de cada uno de ellos:

SERVIDOR HTTP:

- **MyHttpServer.java**: Se encarga de leer lo que pongas en el navegador y se lo sirve al HiloServidor. Mediante un socket recibe la información del navegador.

- **HiloServidor.java:** Cuando recibe la información del MyHttpServer se encarga de leer y escribir al cliente respondiendo adecuadamente con un recurso. Este recurso solicitado puede ser de dos tipos:

    - **Estático:** el servidor comprobará si posee este recurso y lo servirá o devolverá un mensaje de error del tipo “recurso no encontrado”.

    - **Dinámico:** acceder a unas de los componentes de las estaciones meteorológicas para obtener o establecer un valor. Deberá conectarse mediante socket al controlador para conseguir la petición adecuada.

- **index.html:** Página HTML inicial, describiendo brevemente la práctica.

- **error.html:** Página HTML de error 404 cuando un recurso no sea accesible o no hay conexión con el controlador.

CONTROLADOR:

- **Controller.java:** Cuando el recurso solicitado que viene del HiloServidor es dinámico es necesario llamar al Controller. Invocará al componente distribuido Java RMI correcto, generará el contenido HTML necesario y lo enviará al HiloServidor.

ESTACION Y RMI:

- **Station.java**: Se encarga de crear los objetos Station, una implementación de Interfaz. Contiene los métodos Set y Get necesarios para la práctica.
  Definiremos cada estación meteorológica con los siguientes elementos:

    - **Temperatura:** obtiene un valor de temperatura de entre -30ºC y 50 ºC.

    - **Humedad:** obtiene un valor de entre 0 y 100.

- **Luminosidad:** obtiene un valor de entre 0 y 800.

- **Pantalla LCD:** pantalla que muestra mensajes, tamaño de 150 caracteres.

- **InterfazRemoto.java**: conjunto de métodos que pueden ser invocados remotamente por el Controller. Los métodos se implementan en Station.

- **Registro.java**: Se encarga de registrar la estación en el registro RMI para que el Controller pueda acceder a ella.

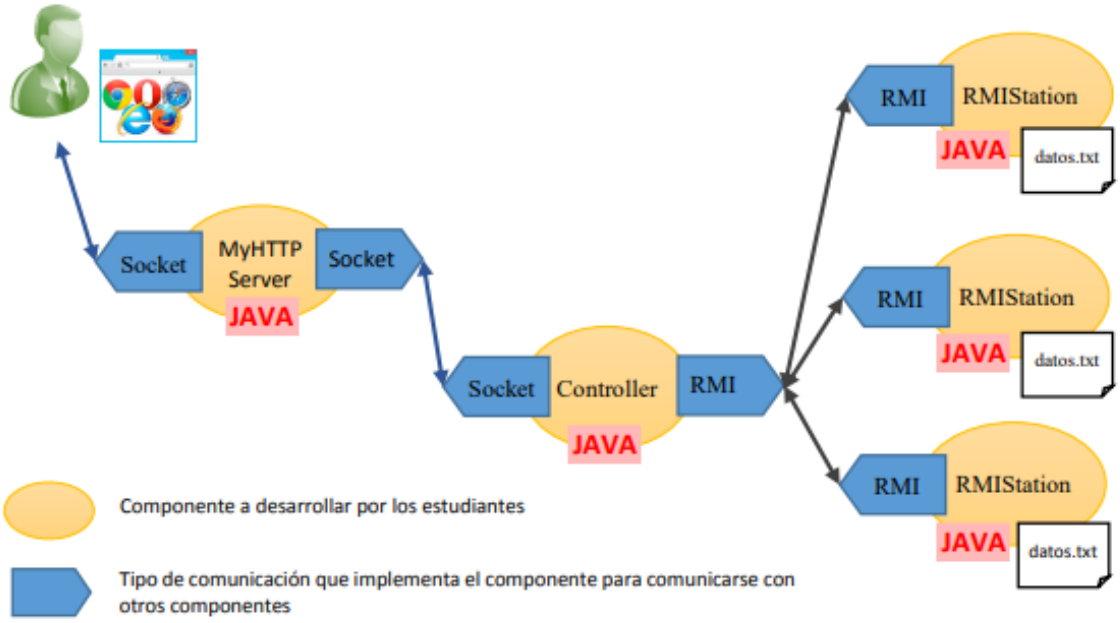
- **registrar.policy:** Política de seguridad necesaria debido al modelo de seguridad de Java 2.

En todas las respuestas HTML que hagamos al navegador aparecerán siempre las siguientes opciones:

- **Estaciones:** Aparecerá una lista de las estaciones registradas en el RMIRegistry.

- **Inicio:** Podremos volver siempre al index.html.

Distribución:



3. Guía de despliegue:

A continuación, haré una guía de despliegue de la aplicación:

Primer PC:

- **Compilar** MyHTTPServer e HiloServidor:

  Javac MyHttpServer.java && Javac HiloServidor.java

- **Ejecutar** myHttpServer.java:

java myHttpServer <DirecciónController> <PuertoNavegador> <PuertoController>

Segundo PC:

- **Compilar** Controller:

  Javac Controller.java

- **Ejecutar** Controller:

  java -Djava.security.policy=registrar.policy Station.Controller
  <DirecciónRMI><PuertoRMI><Puerto HTTP>

Tercer PC:

- **Compilar** Station, Registro e InterfazRemoto, e **iniciar** RMIregistry:

  Javac Station.java && Javac Registro.java && Javac InterfazRemoto.java

  rmiregistry -J-Djava.security.policy=registrar.policy.

- **Ejecutar** Registro:

  java -Djava.security.policy=registrar.policy Station.Registro
  <DirecciónRMI><IDEstacion>

Además, añadir en **CLASSPATH:**

export CLASSPATH=$CLASSPATH:ruta_interfaz

export PATH=$PATH:ruta_j2sdk/bin

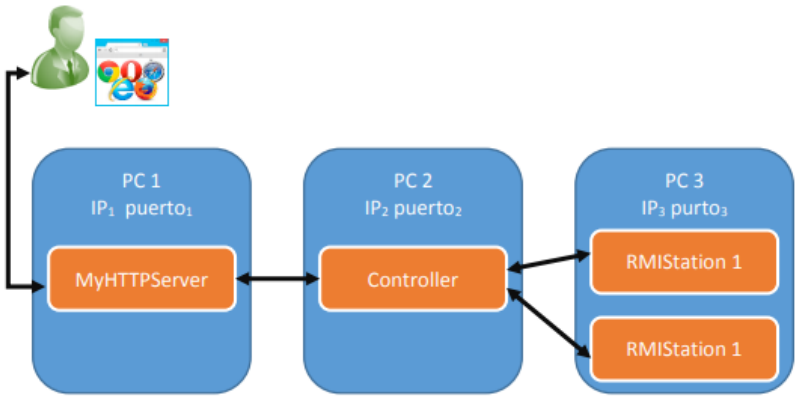
export CLASSPATH=$CLASSPATH:/home/guest/cliente.jar

Y generar los **stubs y skeletons**:

rmic Station.java

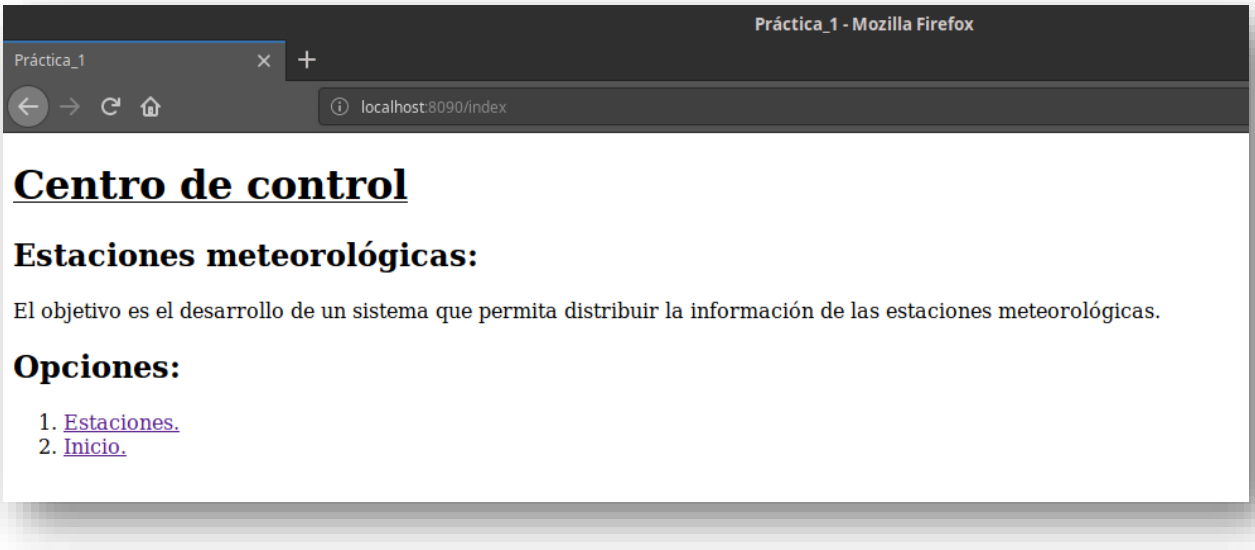
jar cvf cliente.jar InterfazRemoto.class Station_Stub.class

Escenario:

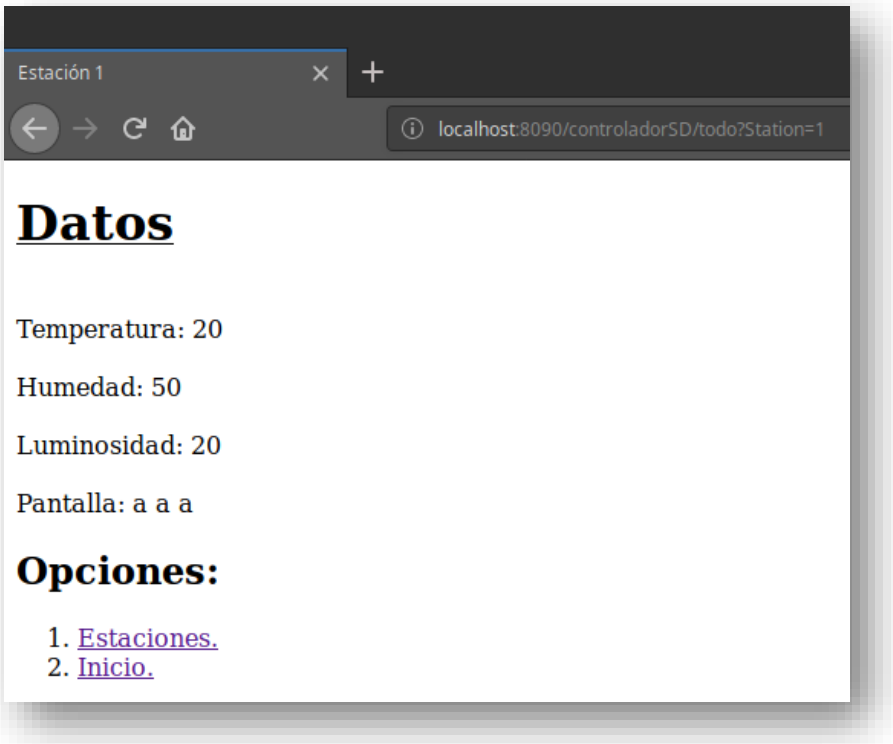


4. Capturas de pantalla:

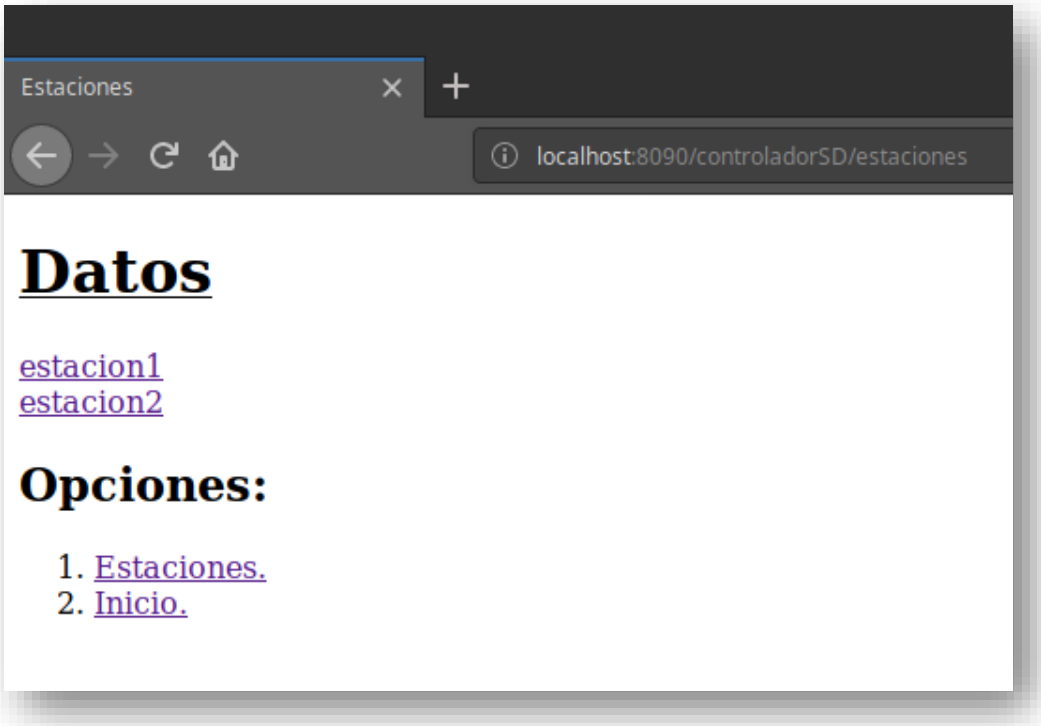
Índice:



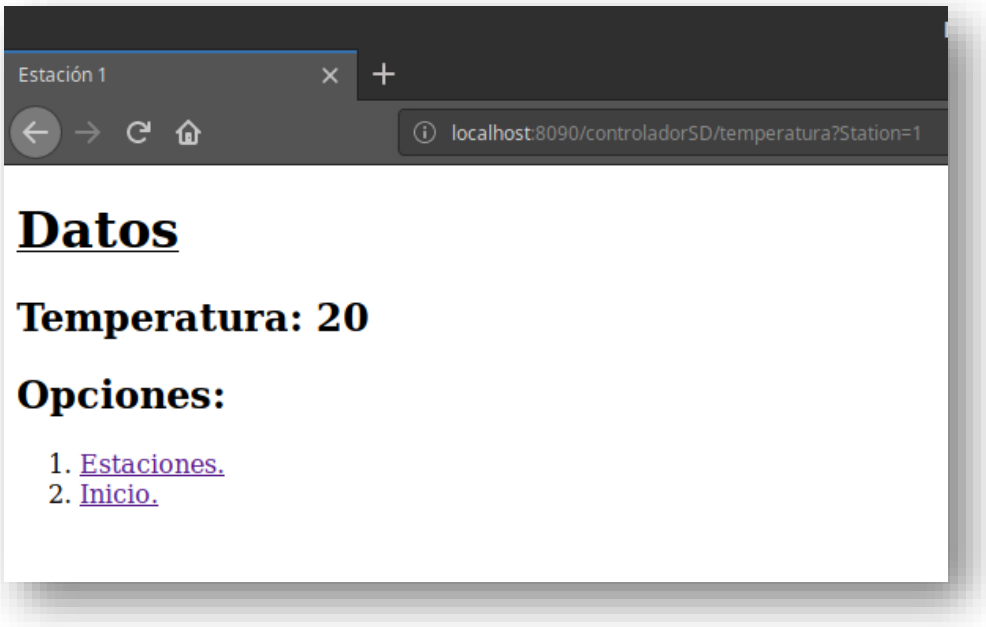
Lista de estaciones:



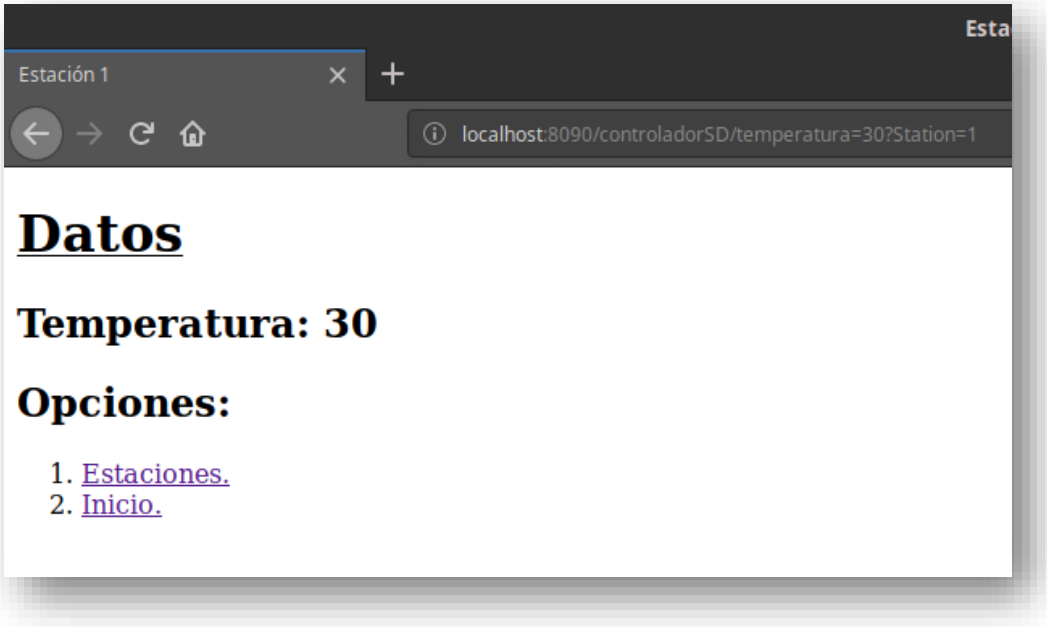
Todos los elementos de una estación:



Cambiar un elemento:

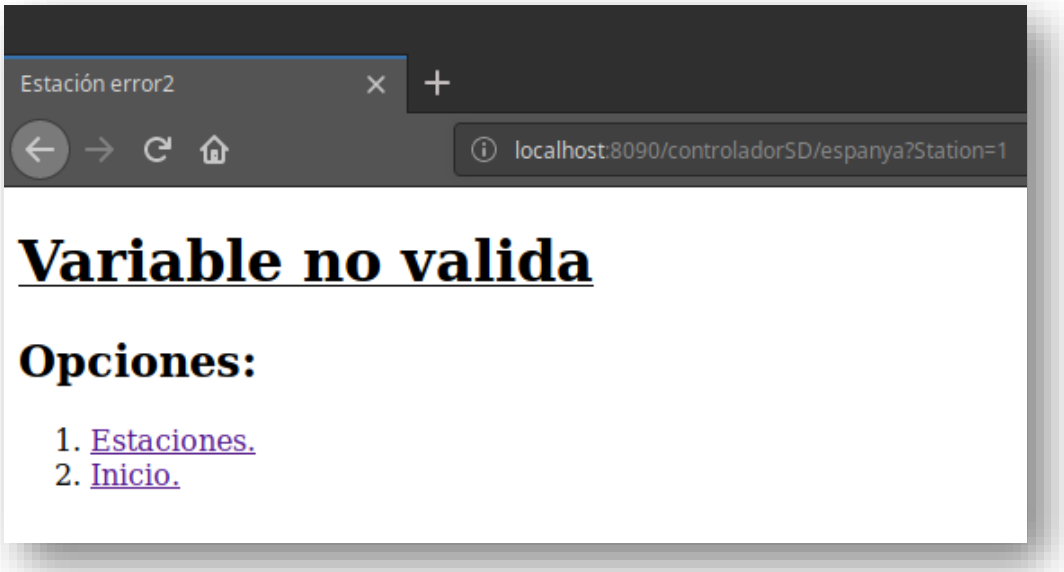


Imprimir un elemento:



Error 404:

Error de Variable:



Error de Estación: