SD	**Sistemas Distribuidos**
18/19	Práctica no guiada
	Control Estaciones Meteorológicas Sockets y RMI

Preámbulo

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 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.

Por otro lado, la utilidad de RMI 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.

Especificación

El objetivo de la práctica a desarrollar es un sistema distribuido de gestión de estaciones meteorológicas. Vamos a suponer que somos una organización que se dedica a la monitorización de variables climáticas en estaciones repartidas por un amplio territorio, para ofrecer así el estado meteorológico del país.

Cuando una empresa ofrece un servicio, lo habitual es que ésta intente llegar al máximo de clientes posibles, y para ello no es extraño que implemente diferentes mecanismos de

comunicación haciendo uso de diversas tecnologías. En esta práctica se van a emplear dos tecnologías que serán revisadas en profundidad en clases de teoría:

- Sockets: los sockets son un sistema de comunicación entre procesos de diferentes máquinas y ofrecen un punto de comunicación por el cual un proceso puede emitir y recibir información.
- RMI: es un mecanismo ofrecido por Java exclusivamente para invocar un método de manera remota, proporcionando un mecanismo simple para la comunicación entre aplicaciones.

El objetivo es el desarrollo de un sistema que permita distribuir la información de las estaciones meteorológicas. Dado que para este ejercicio no tenemos físicamente una estación meteorológica, vamos a simularla vía software, es decir, construiremos un programa que simule que tiene los elementos que a continuación se detallarán. Una estación meteorológica posee sensores (elementos que capturan alguna información del medio) y actuadores (dispositivos con los que se puede interactuar), en nuestro caso definiremos cada estación meteorológica con los siguientes elementos:

Sensores	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
Actuadores	Pantalla LCD: pantalla que muestra mensajes, tamaño de 150 caracteres

Tabla 1. Sensores y actuadores de cada estación meteorológica

Aplicación WEB – Sockets y RMI

Se propone al estudiante implementar un sistema distribuido compuesto de los siguientes componentes software:

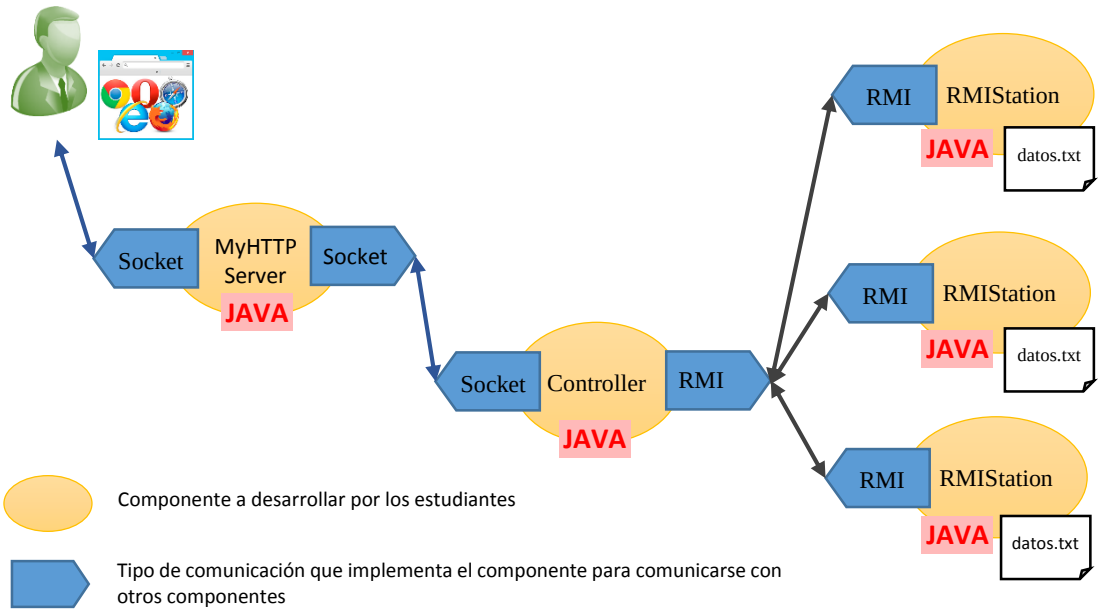

Imagen 1. Esquema conceptual del sistema software, interconexiones entre los componentes y tipos de interfaces.

Los componentes software que el estudiante ha de desarrollar son los siguientes:

- MyHTTPServer: el acceso al sistema de información meteorológica se realizará a través de una aplicación web que permitirá a cualquier usuario, utilizando un simple navegador web, acceder al sistema y poder ver la información de cualquier estación meteorológica. Este componente implementará una versión reducida del protocolo HTTP estándar.
- Controller: este componente se encarga de comunicarse con todas los componentes RMIStation desplegados y acceder a sus métodos bajo demanda del componente MyHTTPServer.
- RMIStation: componente que implementa la estación meteorológica. Utilizará un archivo ".txt" para almacenar los valores.

A continuación se especifica más detalladamente cada componente.

MyHTTPServer

El acceso al sistema se realiza mediante un navegador web haciendo uso de una aplicación que implemente una versión reducida del protocolo HTTP. Esta aplicación será **MyHTTPServer**. La aplicación utilizará el API de Sockets de Java y debe ser capaz de atender peticiones según las siguientes especificaciones.

El servidor MyHTTPServer debe ser capaz de atender peticiones de forma concurrente, es decir, que puede atender peticiones de distintos usuarios que se conectan a través de un navegador de forma simultánea. El número de conexiones simultáneas debe ser parametrizable.

El servidor MyHTTPServer debe poder escuchar en un puerto configurable. Normalmente los servidores web utilizan el puerto 80 para escuchar peticiones HTTP, pero en nuestro caso, y dado que nuestra aplicación podría convivir con otras, es deseable que se pueda indicar en que puerto atiende el servidor las solicitudes HTTP. Eso quiere decir que cuando arranquemos el servidor MyHTTPServer, uno de los argumentos que recibirá es el puerto de escucha. El servidor abrirá un socket en ese puerto y esperará mensajes que provendrán de un navegador web.

El servidor aceptará peticiones HTTP y devolverá respuestas HTTP. En ningún caso se ha de implementar el protocolo HTTP completo, solo los siguientes comandos. En caso de que reciba una petición con algún comando no soportado, debe devolver una respuesta del tipo "comando no soportado".

Peticiones de entrada HTTP

Cada petición MiniHTTP estará compuesta por tres partes como se muestra en la siguiente tabla siguiendo el estándar HTTP pero acotando las opciones:

Línea inicial	Define el método HTTP, la url de acceso al recurso y la versión del protocolo HTTP. Un salto de línea indica su fin y dónde comienza la sección *cabecera* del mensaje. **Sintaxis:** MétodoHTTP+EspacioEnBlanco+URIAccesoRecurso+EspacioEnBlanco+HTTPVersion+SaltoDeLínea **Ejemplo:** GET /prueba.html HTTP/1.1	Métodos a implementar: GET

Cabecera	Directivas que indican características adicionales de la petición y que pueden influir en el tratamiento de dicha petición por parte del servidor. Cada directiva de cabecera estará separada por un salto de línea. Se debe incluir una línea en blanco antes del cuerpo del mensaje para separarlo.	Tipo de URI: Abs_path
Cuerpo	Información necesaria en algunos casos para realizar la operación definida en la línea inicial.	

Tabla 2. Resumen del detalle de PETICION del protocolo HTTP reducido a utilizar

Línea inicial

Sólo se deben aceptar peticiones realizadas mediante el método GET. Aunque existen 4 tipos de URI de acceso a recursos, en el protocolo se implementará la más habitual (absolute path). En ella se indica la ruta absoluta para localizar el recurso. La dirección del servidor se indicará en la directiva de cabecera Host. Esta directiva es usada normalmente cuando el servidor soporta hosts virtuales o permite referenciar al servidor por varios nombres. En este caso, las rutas a los recursos pueden repetirse y deben ser diferenciadas por el nombre del servidor.

En nuestro caso, no será necesario el uso de la directiva Host.

Cabecera

Para simplificar la práctica y la implementación del servidor HTTP no se van a tener en cuenta las cabeceras de las peticiones.

Cuerpo del mensaje

Al implementar únicamente el método GET, este elemento deberá ignorarse.

Respuesta de salida MiniHTTP

La respuesta HTTP que se debe implementar está compuesta por tres partes:

Línea de estado	Indica el estado del servidor como respuesta a la petición realizada. **Sintaxis:** HTTPVerión+EspacioEnBlanco+CódigoDeEstado+EspacioEnBlanco+DescripciónEstado+SaltoDeLínea **Ejemplo:** HTTP1.1 200 OK Los códigos de estados pueden ser consultados en: http://www.w3.org/Protocols/rfc2616/rfc2616-sec6.html#sec6
Cabecera	Directivas que indican características adicionales de la respuesta y que pueden influir en el tratamiento de dicha respuesta por parte del navegador. Cada directiva de cabecera estará separada por un salto de línea. Se debe incluir una línea en blanco antes del cuerpo del mensaje para separarlo.
Cuerpo	Contenido HTML que será mostrado al cliente a través del navegador.

Tabla 3. Resumen del detalle de RESPUESTA del protocolo HTTP reducido a utilizar

Línea de estado

Utilizaremos la versión HTTP1.1. Cuando el servidor reciba durante una petición algún método no soportado deberá devolver un código de estado “405” “Method Not Allowed”.

Cabecera

La sintaxis es:

```
Directiva: valor
Directiva: valor
Directiva: valor
...
```

A continuación se describen las cabeceras que se deben incluir en la respuesta HTTP.

Connection	Indica el tipo de conexión, si el servidor soporta conexiones persitentes o no. En este caso no se soportarán conexiones persistentes por lo que en todas las respuestas se enviará el valor close.
Content-Lenght	Longitud del contenido del cuerpo del mensaje. El tamaño es indicado en número de bytes contenidos.
Content-Type	Indica el tipo de información enviada en el cuerpo del mensaje y la codificación del mensaje (charset).
Server	Incluye información sobre el servidor.

Tabla 4. Resumen del detalle de CABECERAS DE RESPUESTA del protocolo HTTP reducido a utilizar

Cuerpo del mensaje

Contendrá el código HTML que será mostrado por el navegador mostrando la información de los sensores o la descripción de la operación realizada por el actuador. Destacar aquí que MyHTTPServer sólo devuelve el contenido de los recursos solicitados, es decir que si se solicita un .txt se devolverá su contenido, si es un .html igual, y así para todo. El servidor no formatea, adapta ni transforma ningún contenido, solo sirve los recursos solicitados o en caso de que no pueda acceder a ellos simplemente devolverá una página de error.

Ante cualquier duda se deberá seguir la especificación estándar de HTTP que puede encontrarse en http://www.w3.org/Protocols/rfc2616/rfc2616.html

Funcionamiento general de MyHTTPServer

El funcionamiento general de la aplicación MyHTTPServer será entonces muy similar a la de un Servidor Web convencional (como Apache, IIS, nginx…). El servidor recibirá una petición solicitando un recurso (por ejemplo “index.html”). Este recurso solicitado puede ser de dos tipos:

- O bien un recurso estático, con lo que el servidor comprobará si posee este recurso y lo servirá o devolverá un mensaje de error del tipo “recurso no encontrado”.
- Un recurso dinámico, es decir, acceder a unas de los componentes de las estaciones meteorológicas para obtener o establecer un valor, con lo que MyHTTPServer deberá conectarse con Controller y hacer la petición adecuada, recibir respuesta y esta respuesta transmitirla al usuario.

Es deseable que el estudiante cree al menos una página index.html que en caso de no especificar ningún recurso (es decir, el usuario introduce http://IP:puerto) se utilice como página por defecto, permitiendo así una navegación cómoda.

Errores

Los errores que debe detectar y manejar adecuadamente (es decir, que debe devolver un mensaje que indique cada uno de ellos) el servidor son los siguientes:

- Recurso no accesible: se solicita un recurso estático que no existe.
- No hay conexión con controlador: si no se ha podido establecer una conexión con el controlador.

Controller

La aplicación MyHTTPServer mediante el protocolo HTTP estándar únicamente permite el acceso a recursos estáticos a través de la red. Para acceder a recursos dinámicos, que pueden ser aplicaciones ejecutables, en el lado del servidor es necesaria la introducción de algún elemento que extienda y haga posible la invocación de aplicaciones ejecutables. En nuestro caso lo haremos a través de un elemento controlador denominado **controller**.

La función del controlador es dar acceso a los diferentes módulos funcionales que nos ofrecen las estaciones meteorológicas (sensores y actuadores) y que son encapsulados a través de componentes de negocio distribuidos haciendo uso de RMI. Para ello, el controlador será invocado cada vez que la URL del recurso invocado cumpla un patrón previamente definido en el servidor MyHTTPServer. Por ejemplo, en el servidor se puede establecer que todas las url´s que sean del tipo

http://IPServer:Port/**controladorSD/**...

deban invocar al controlador. Es decir, url´s como:

http://IPServer:Port/index.html
http://IPServer:Port/page1.html
http://IPServer:Port/error.html

devolverán el recurso estático solicitado (index.html, page1.html, error.html). Mientras que cualquier url que contenga tras la IP del servidor y el puerto de escucha la palabra "controladorSD" deberá invocar a **controller**.

Una vez que el servidor Web detecte este patrón, encapsulará los datos de la petición y se los pasará al controlador. Un ejemplo de encapsulación de datos podrías consistir en la siguiente tabla, ante una petición con la siguiente url:

http://192.168.1.1:3000/controladorSD/temperatura?Station=1

Objeto Request	Descripción	Ejemplo
ResourcePath	Ruta al recurso a continuación de controladorSD	temperatura
Parameters []	Colección de parámetros incluidos en la petición GET	station=1
Headers []	Colección de directivas de cabecera que soporta el servidor	

Tabla 5. Ejemplo de encapsulación de datos para que desde MyHTTPServer de invoque a Controller

Esto quiere decir, que el servidor detecta que variables y la estación a la que quiere acceder y realiza una petición a controlador. El protocolo de comunicación entre servidor y controlador no tiene por qué cumplir con ningún estándar, pudiendo determinarlo el estudiante.

A partir de la petición recibida, el controlador debe ser capaz de invocar al componente distribuido Java RMI correcto. Para ello deberá realizar una búsqueda en el registro con la información recibida, obtener el *stub* correspondiente e invocar al componente.

Posteriormente, el controlador o un componente especializado, debe generar el contenido HTML con la información obtenida que se le pasará al servidor Web para que se lo presente al usuario a través del navegador. A continuación se muestra un ejemplo con la secuencia de acciones que ocurren ante varias peticiones:

Ejemplo 1. Solicitud de un recurso estático.

- El servidor recibe la petición “get /index.html HTTP 1.1”
- El servidor detecta que el patrón de recurso dinámico (la subcadena “controladorSD”) no está en la petición, por tanto es un recurso estático.
- El servidor mira si en su sistema está el recurso solicitado (si posee el archivo “index.html”):
    - En caso de que si existe, accede a su contenido y lo devuelve en una respuesta HTTP al solicitante.
    - En caso de no existir devuelve una respuesta de error

Ejemplo 2. Solicitud de un recurso dinámico que SI existe.

- El servidor recibe la petición “get /controladorSD/temperatura?station=1 HTTP 1.1”
- El servidor detecta el patrón de recurso dinámico (la subcadena “controladorSD”), por tanto es un recurso dinámico.
- El servidor encapsula en un mensaje una petición para controller indicando que desea obtener la temperatura de la estación meteorológica registrada como 1. Se conecta a controlador y le envía la petición, quedando a la espera de respuesta.
- El controlador recibe la petición de servidor, detecta los parámetros de la petición e invoca al componente distribuido adecuado RMIStation para obtener respuesta (la temperatura d la estación 1). El controlador recibe la respuesta de RMIStation (por ejemplo 35), con este valor compone una página web HTML válida y lo devuelve en un mensaje de respuesta al servidor.
- El servidor recibe la respuesta y la envía a su vez al usuario que originalmente realizó la petición.

Ejemplo 3. Solicitud de un recurso dinámico que NO existe.

- El servidor recibe la petición “get /controladorSD/radiacionUVA?station=1 HTTP 1.1”
- El servidor detecta el patrón de recurso dinámico (la subcadena “controladorSD”), por tanto es un recurso dinámico.
- El servidor encapsula en un mensaje una petición para controller indicando que desea obtener la radiacionUVA de la estación meteorológica registrada como 1. Se conecta a controlador y le envía la petición, quedando a la espera de respuesta.
- El controlador recibe la petición de servidor, detecta los parámetros de la petición, detecta que no es uno de los parámetros correctos y devuelve un error del tipo “variable no valida”. Este error es una página web HTML válida.
- El servidor recibe la respuesta y la envía a su vez al usuario que originalmente realizó la petición.

Consideraciones sobre Controller

Como puede verse, el controlador es un intermediario entre el servidor que atiende las peticiones HTTP y los componentes distribuidos RMIStation. Se encarga de recibir una petición del servidor, comprobar que el recurso y acción solicitados son correctos, invocar el método adecuado para obtener un resultado, crear una página web HTML de respuesta y devolverla al servidor.

El servidor por su parte NO crea páginas web HTML, solo accede a recursos estáticos o realiza peticiones al controlador. Eso quiere decir que igual que en un servidor web convencional las páginas de error son páginas HTML estáticas que ya existen y que son devueltas por defecto cuando se produce un error del tipo “recurso no encontrado”.

Por otro lado, RMIStation por tanto, tras acceder a un objeto y obtener el valor de uno de los sensores por ejemplo, se devuelve dicho valor (nunca una página web HTML u otro documento), siendo el controlador el que al recibir dicha respuesta, compone una página HTML de respuesta que envía al servidor.

Se deja a criterio del estudiante proporcionar la utilidad de tener una URL del servidor que permita obtener un listado de todas las estaciones meteorológicas junto con el acceso a todas las variables, por ejemplo si el usuario introduce la url: http://IP:puerto/controladorSD/index , que esta url produzca como resultado una página html con la lista de todas las estaciones y el acceso a cada variable. De esta forma, el uso de la aplicación es mucho más intuitivo.

Errores

Los errores que debe detectar y manejar adecuadamente (es decir, que debe devolver un mensaje que indique cada uno de ellos) el controlador son los siguientes:

- Variable no válida: se solicita información de una variable de la estación que no existe, por ejemplo “radiacionUVA”.
- La estación no existe: se solicita información de una estación que no existe, por ejemplo “station=123” si esta no ha sido registrada.

RMIStation

Cada componente distribuido encapsulara la funcionalidad de los sensores y actuadores. El valor de los sensores debe poder ser obtenido en cualquier momento bajo petición, y el valor de los actuadores (la pantalla de display) debe poder ser además establecido. Como no tenemos un dispositivo físico real al que acceder, el funcionamiento de la estación meteorológica será simulado mediante un archivo. Es decir, cada estación RMIStation que sea puesta en marcha va a tener un archivo de texto llamado “estacionX.txt” (donde la X indicará el número de la estación) que tendrá el siguiente contendió:

```
Temperatura=30
Humedad=90
Luminosidad=450
Pantalla=Hola, esta es la practica de SD
```

Los valores de temperatura, humedad y luminosidad serán inicialmente establecidos y podrán cambiarse manualmente en cualquier momento. Estos son los valores que deberá devolver RMIStation cuando sea invocado uno de sus métodos. El valor de Pantalla además de ser devuelto bajo consulta también debe poder ser cambiado bajo petición (es decir, que existirá además de un método get, un método set).

Esto quiere decir que si por ejemplo en el sistema se registran 3 estaciones meteorológicas, al registrar a cada una se le ha de asignar un número, por ejemplo “1”, “2” y “3”, y cada una de ellas deberá poseer un archivo “estacion1.txt”, “estacion2.txt” y “estacion3.txt”. Estos archivos pueden haber sido creados manualmente antes de lanzar el sistema o pueden ser creados por defecto al arrancar la aplicación.

Nota: estos archivos solo tienen el objetivo de simular que existe un recurso físico que sería consultado o actuado desde RMIStation.

Guía mínima de despliegue

Para la correcta evaluación de la práctica es necesario comprobar que la aplicación distribuida solicitada es desplegada (puesta en marcha) en un entorno verdaderamente distribuido. Es por ello que para su prueba es necesario al menos 3 PCs distintos en los que se desplegarán los componentes solicitados. Al menos se ha de desplegar junto con el servidor y el controlador, 2 componentes de estación meteorológica, proporcionando el siguiente escenario:

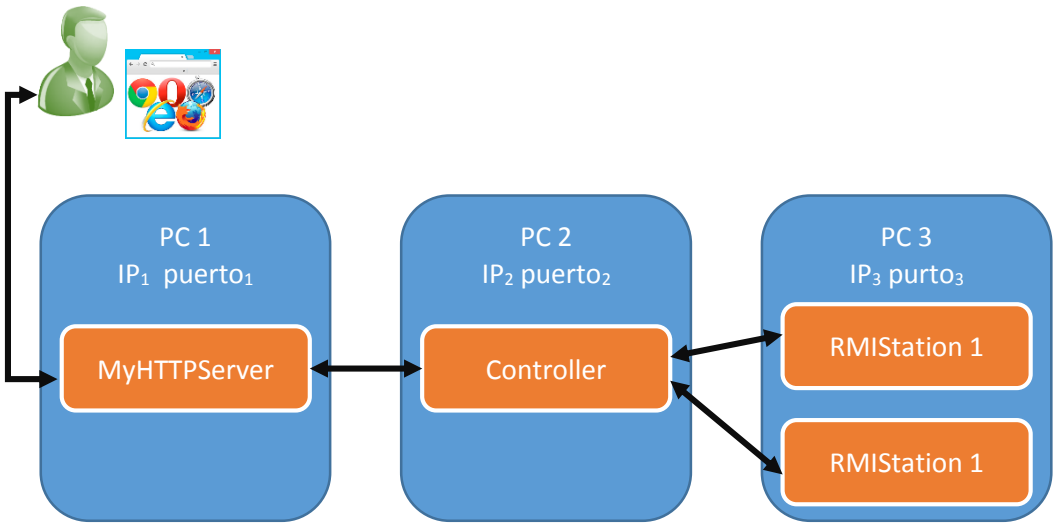

Imagen 2. Escenario físico mínimo para el despliegue de la práctica.

Por supuesto las estaciones RMIStation podrían ser desplegadas a su vez, cada una en un PC distintos, pero al menos se ha de poder comprobar que se pueden utilizar de forma separada el servidor en un PC, el controlador en otro PC y las estaciones en otro.

Para facilitar al estudiante el despliegue se permite el uso de máquinas virtuales, por lo que el estudiante podría utilizar un único PC físico en el que arranque dos máquinas virtuales más, lo que le permitiría desplegar uno de los componentes sobre el PC real, y los otros sobre las dos máquinas virtuales.

Finalmente, para desplegar la aplicación, en general se deberá seguir la siguiente secuencia de arranque:

- Iniciar el registro RMI (consultar las prácticas y ejemplos no guiados de RMI)
- Registrar las estaciones RMIStation 1 y RMIStation 2
- Iniciar el controlador indicando puerto de escucha y conexión al registro RMI

- Iniciar servidor indicando puerto de escucha del servidor y puerto de conexión al controlador.

Entregables y evaluación

La evaluación de la práctica se realizará en los laboratorios. **El estudiante debe desplegar, en los ordenadores del laboratorio, el sistema completo que cumple con la especificación del enunciado, es decir, un servidor, un controlador y al menos dos estaciones, todos ellos interconectados entre sí** (requisitos indispensables para poder realizar la corrección). Además deben poderse evaluar positiva o negativamente todos los apartados que aparecen en la Guía de corrección (ver documento de guía de corrección de la práctica de sockets y RMI). Cada uno de los apartados puntúa de forma variable, por tanto cada apartado no implementado o que no pueda comprobarse su correcto funcionamiento no podrá ser tenido en cuenta y por tanto no puntuara. El estudiante deberá presentar para la evaluación el documento **"Guía de corrección"** cumplimentado para que el profesor pueda validar los apartados implementados.

El estudiante deberá entregar, además, por tutoría virtual a su profesor de prácticas una **memoria de prácticas**, un documento donde se detalle la siguiente información. El formato es libre pero debe ser un documento ordenado y debidamente formateado, cuidando la redacción y ortografía.

- Portada con el nombre, apellidos y DNI del estudiante, año académico y el título de la práctica.
- Una informe donde se indique el nombre de los componentes software desarrollados y una descripción de cada uno de ellos, indicando además del servidor, controlador y estaciones, otros códigos como registros o generador de html… Cualquier archivo .java de código que el estudiante haya creado debe indicarlo y describirlo.
- El detalle, paso a paso, de una guía de despliegue de la aplicación, que deberá ser la misma que utilice cuando haga la corrección de la práctica.
- Capturas de pantalla que muestren la página de inicio de la aplicación, la página de inicio que produce el controlador con todas las estaciones registradas, las páginas de resultados al consultar alguna variable de una estación y las páginas de error implementadas.

Cada profesor de prácticas podrá solicitar a los estudiantes cualquier otra evidencia que considere adecuada para poder formalizar la evaluación al igual que podrá hacer cualquier pregunta sobre el código que implementa la práctica.