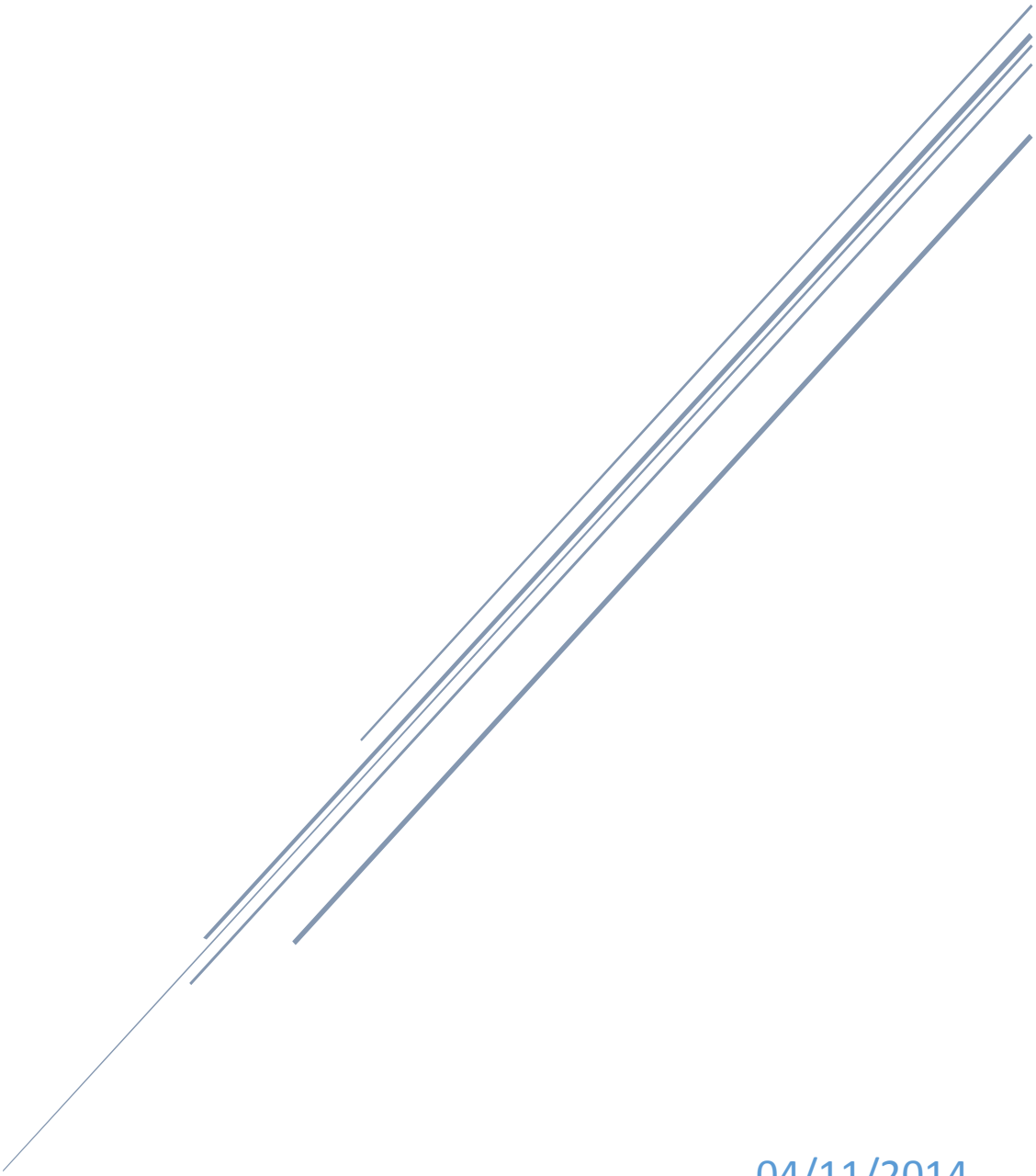
SISTEMAS DISTRIBUIDOS

Práctica 3



04/11/2014

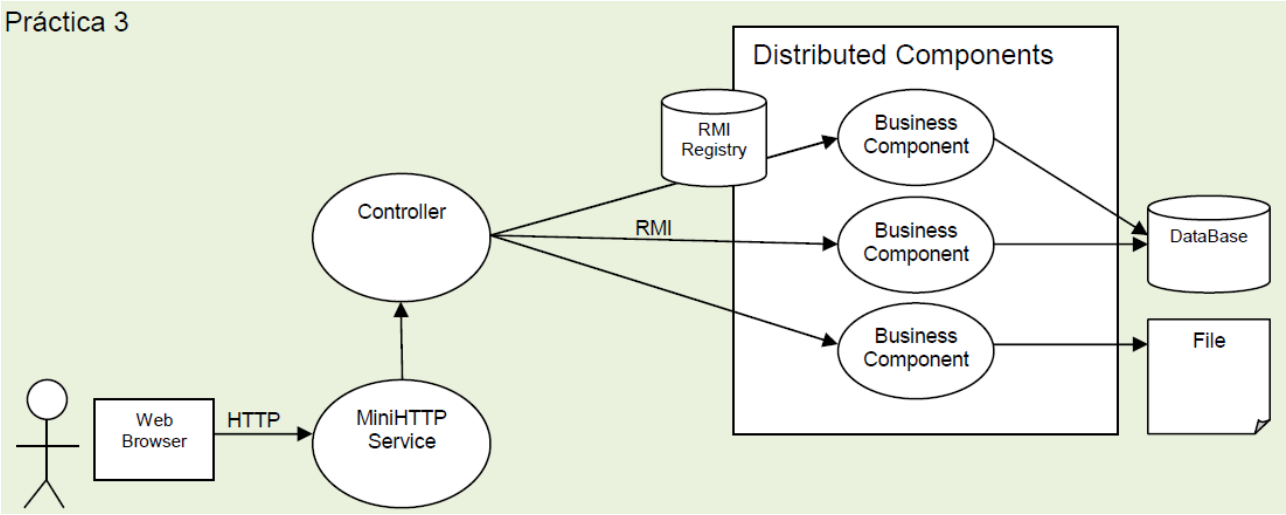
Encarna Amorós Beneite

Índice

1. Introducción 2
2. Archivos 3
3. Vistas 6
4. Scripts utilizados 9
5. Despliegue 12

1. Introducción

El objetivo de la práctica es el diseño de un sistema distribuido que gestione máquinas de vending permitiendo la monitorización y el control de las mismas a través de internet mediante los sensores y actuadores que contienen.



Los sensores de las máquinas serán:

- Stock
- Monedero
- Temperatura
- Estado
- Fecha y hora de la lectura

Los actuadores serán:

- Display
- Alimentación
- Refrigerador

2. Archivos

En este apartado explicaremos para que sirve cada archivo contenido en la práctica.

La carpeta **Servidor_RMI** contiene:

- **MaquinaVending.java** – Contiene la clase MaquinaVending la cual representa al objeto remoto. Dicho objeto es el que se encontrará en una máquina remota. En ella se sitúan los sensores de la máquina (estados) y los actuadores (métodos).
- **InterfazMaquinaVending.java** – En ella se encuentra una clase que es interfaz del objeto remoto anteriormente mencionado. Define los métodos de dicho objeto remoto que es la máquina vending y debe extender la interfaz java.rmi.Remote para que la comunicación RMI pueda funcionar.
- **Registro.java** – Contiene una clase que es la encargada de registrar el objeto remoto en el servicio de nombres para que pueda ser accedido desde el cliente que será nuestro controlador.
- **registrar.policy** – Es un archivo con la política de seguridad que debemos incluir debido al modelo de seguridad de Java 2.
- **run_registrar_maquinas.bat** – Es un script que utilizaremos para compilar y registrar una o más máquinas de vending. En el lugar donde se encuentre el objeto remoto deberá de contener también el archivo Registro.java para poder registrar dicha máquina en el servicio de nombres.
- **MaquinaVending_Stub.class** – Es la clase proxy que nos permite la comunicación con el objeto remoto MaquinaVending, implementa la interfaz. Este archivo junto con InterfazMaquinaVending.class serán necesarios en el equipo donde se utilicen métodos del objeto remoto.

La carpeta **Cliente_RMI** contiene:

- **Controlador.java** – Contiene la clase la cual será el cliente RMI. El controlador se encarga de comunicarse con el objeto remoto a través de RMI y de comunicarse con cada instancia de HiloServidor por sockets**.**
  - Se comunica por RMI con los objetos remotos para ver los sensores de cada máquina y generar un archivo *index.html* que contiene la información de cada máquina con sus sensores. Si se clica en una de esas máquinas, se accede a un html que previamente a generado el Controlador (cuando genera index), en el que se encuentran los actuadores

de cada máquina en un formulario. El nombre de este html específico para cada máquina es el nombre de la máquina seguido de su id con la extensión html.

- Una vez generados los archivos index.html y cada html para cada máquina, el controlador cada 2 segundos comprueba los sensores de la máquina y vuelve a generar el index y el html de cada máquina por si los valores han cambiado. Además, si hay valores incorrectos, pone la máquina fuera de servicio y muestra alertas (tanto en las páginas html como en la pantalla donde se han registrado las máquinas de vending).
- El controlador es también, aparte de cliente RMI, el servidor de cada instancia de la clase HiloServidor. Si HiloServidor le envía un string en el que se incluyen actuadores de alguna máquina, el Controlador se comunica por RMI con las máquinas para ejecutar dicho actuador y devuelve a HiloServidor una cadena que en html mostrará que se ha cambiado.

- **ServidorConcurrente.java** – Contiene una clase que se comunica por sockets (siendo el servidor) con un cliente, es el servidor HTTP. Este cliente será cualquier navegador web. Por cada petición que le llegue instanciará un hilo de la clase HiloServidor, de esta forma conseguimos tener un servidor que escuche y se comunique con más de una petición a la vez.
- **HiloServidor.java** – Es el hilo creado por ServidorConcurrente. Se encarga de escuchar peticiones del cliente navegador web. Recibe una petición HTTP en la que si el método HTTP no es GET se muestra una página html indicando dicho error junto con el código 405 en la respuesta HTTP. Si le llega una petición con una URL en la que se pide un URI, es decir, un recurso html busca en su directorio si encuentra dicho archivo, lo lee y lo manda por sockets al navegador web. Si no lo encuentra recibe una página html de error indicándolo, así como el código 404 en la respuesta HTTP. Finalmente, si detecta una URL de acceso a recursos en la que se pide un archivo html junto con parámetros, envía una cadena que le proporciona el controlador cuando le pasa dichos parámetros.
- **registrar.policy** – Es un archivo con la política de seguridad que debemos incluir debido al modelo de seguridad de Java 2.
- **index.css** – Es el estilo css que tendrá el fichero index.html.
- **maquina.css** – Es el estilo css que tendrá el fichero html de cada página y el de las paginas en formato string devueltas por el controlador cuando ejecuta actuadores del objeto remoto comunicándose con él por RMI.

- **run_servicio_nombres.bat** – Es un script que ejecuta una serie de instrucciones para levantar el servicio de nombres (lugar donde se registraran las maquinas vending remotas y donde se buscarán por el controlador).
- **run_controlador.bat** – Script con una serie de instrucciones para compilar y ejecutar el controlador.
- **run_servidor_http.bat** – Se trata de un script que contiene las instrucciones necesarias para compilar el ServidorConcurrente (que es el servidor HTTP).
- **index.html** – Archivo generado por el controlador en el que se muestran todas las máquinas registradas junto a sus sensores y alertas. Además se puede observar la hora y fecha en la que se ha generado dicho index. Si se pulsa en una de las máquinas se accederá a una página html con sus actuadores.
- **<nombre máquina><id máquina>.html** – Archivo generado por el controlador en el que se muestra para ese nombre e id de la máquina un form con sus actuadores, es decir, en dicho formulario se permite cambiar los valores de la máquina. Si se clica en volver, se mostrará la página index.html. Si se cambia un valor se mostrará una página html (devuelta y escrita por el controlador al ServidorConcurrente) en la que se indica que valores se han cambiado.
  Se muestra de la siguiente manera:

3. Vistas

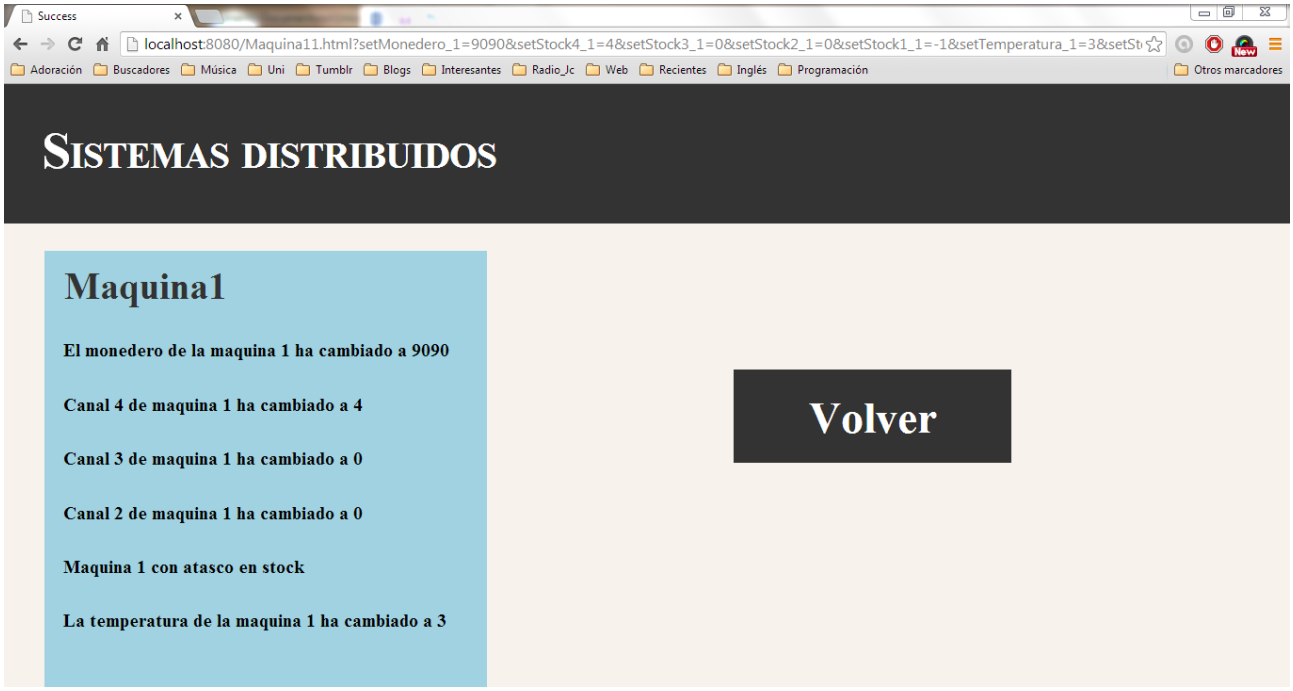
La página index.html se mostraría de la siguiente manera:

Se muestra cada máquina levantada con sus sensores. En este caso no hay ninguna alerta y la máquina funciona correctamente.

La página <nombre maquina><número id>.html es la siguiente:

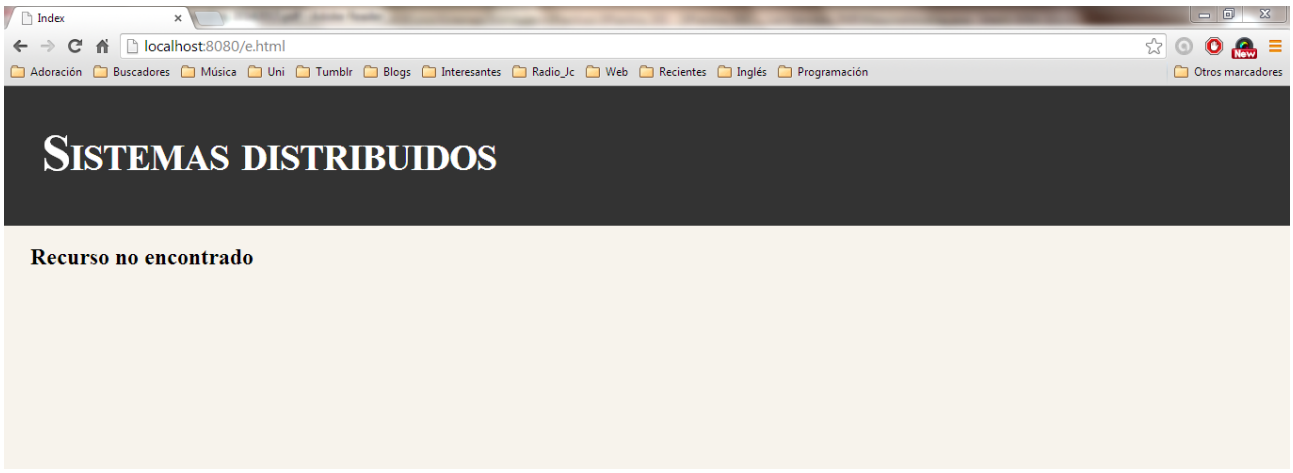
En el formulario se pueden cambiar los distintos valores. Si como argumento escribimos algo del tipo *IP:<puerto http>/<nombre maquina><id maquina>.html?set_Monedero_<id maquina>=valor&* también se cambia. Si clicamos en volver iríamos a la página index.html.

En esta vista vemos el resultado de enviar los datos del formulario. Se nos informa de que los valores efectivamente se han cambiado. Si clicamos en volver iríamos a la página index.html.



Si volvemos a la página index.html vemos los valores cambiados y las respectivas alertas.

Si accedemos a un recurso que el ServidorConcurrente (el mini servidor http) no encuentra, se muestra una página de error.


Index
localhost:8080/e.html
Adoración
Buscadores
Música
Uni
Tumblr
Blogs
Interesantes
Radio_Jc
Web
Recientes
Inglés
Programación
Otros marcadores
SISTEMAS DISTRIBUIDOS
Recurso no encontrado

4. Scripts utilizados

En este apartado explicaremos las instrucciones contenidas en cada script, para qué se utilizan y cómo deben utilizarse (parámetros…). Los scripts utilizados han sido los siguientes:

- run_servicio_nombres.bat este script se ejecuta en la carpeta Cliente_RMI y contiene:

set CLASSPATH=C:\Users\Encarna\Documents\Documentoos\Universidad\3Curso\Sistemas Distribuidos\Practicas\3Practica_SD\src\Servidor_RMI

Con esta instrucción especificamos donde se encuentra la interfaz del objeto remoto, es decir InterfazMaquinaVending. Es necesaria para que posteriormente en la clase Registro se puedan utilizar los métodos del objeto remoto MaquinaVending.

La instrucción genérica sería *set CLASHHPATH=<ruta de la InterfazMaquinaVending>.*

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

En esta instrucción levantamos el servicio de nombres para que las máquinas se puedan registrar.

- run_registrar_maquinas.bat este script se ejecuta en la carpeta Servidor_RMI contiene:

javac InterfazMaquinaVending.java

Compilamos la clase Interfaz del objeto remoto, InterfazMaquinaVending.

set CLASSPATH=C:\Users\Encarna\Documents\Documentoos\Universidad\3Curso\Sistemas Distribuidos\Practicas\3Practica_SD\src\Servidor_RMI

Especificamos la ruta donde se encuentra la interfaz InterfazMaquinaVending del objeto remoto.

La instrucción genérica sería *set CLASHHPATH=<ruta de la InterfazMaquinaVending>.*

javac MaquinaVending.java

Compilamos la clase del objeto remoto MaquinaVending.

rmic MaquinaVending

Gracias a esta instrucción se generan los stubs y skeletons para hacer posible la conexión RMI.

jar cvf cliente.jar InterfazMaquinaVending.class MaquinaVending_Stub.class

Los archivos .class de la interfaz del objeto remoto InterfazMaquinaVending y el stub del objeto remoto MaquinaVending se necesitan en el equipo donde se encuentre el cliente RMI, es decir, el controlador. Por lo que reunimos estos dos archivos en un .jar al que llamaremos cliente.jar.

javac Registro.java

Compilamos la clase Registro.

java -Djava.security.policy=registrar.policy Registro 127.0.0.1 1099 Maquina1 Maquina2 Maquina3 Maquina4 Maquina5 Maquina6 Maquina7 Maquina8

Con esta instrucción estamos registrando una serie de máquinas con sus nombres en una URL del tipo *rmi://IP_servidor_nombres:PUERTO_servidor_nombres/nombre*. Por lo que necesitamos la IP donde se encuentra el servicio de nombres (rmiregistry -J-Djava.security.policy=registrar.policy), el puerto por donde escucha pues con estos datos obtenemos el lugar donde registrar las máquinas y el nombre de la maquina pues cada registro debe diferenciarse, en este caso hemos elegido el nombre de cada máquina.

La instrucción genérica sería *java -Djava.security.policy=registrar.policy Registro <IP_servicio_nombres> <PUERTO_servicio_nombres> <Nombres maquinas>.*

- run_controlador.bat contiene:

set CLASSPATH=C:\Users\Encarna\Documents\Documentoos\Universidad\3Curso\Sistemas Distribuidos\Practicas\3Practica_SD\src\Servidor_RMI\cliente.jar;C:\Users\Encarna\Documents\Documentoos\Universidad\3Curso\Sistemas Distribuidos\Practicas\3Practica_SD\src\Cliente_RMI

Se especifica en el CLASSPATH donde se encuentra el archivo cliente.jar (que contiene el .class de la interfaz del objeto remoto y el stub del objeto remoto) y la propia ruta del controlador. La instrucción genérica sería *set CLASSPATH=ruta_jar;ruta_controlador*.

javac Controlador.java

Compilamos el controlador.

java -Djava.security.policy=registrar.policy Controlador 1234 127.0.0.1 1099 8 Maquina1 Maquina2 Maquina3 Maquina4 Maquina5 Maquina6 Maquina7 Maquina8

Ejecutamos el controlador para que se quede a la escucha de peticiones provenientes del ServidorConcurrente (que es el servidor http) y se conecte por RMI al objeto remoto para contestar y ejecutar diferentes actuadores de la máquina vending y para que genere las diferentes páginas html y las vaya actualizando cada 2 segundos.

Necesitamos la IP y puerto del servicio de nombres ya que el controlador, para poder utilizar sus métodos necesita conocer que objetos remotos son y obtener sus datos y esto se consigue buscando dichas máquinas en el servicio de nombres. Para buscar las máquinas en el servicio de nombres utilizamos la URL que utilizábamos para registrar las máquinas: rmi://IP_servidor_nombres:PUERTO_servidor_nombres/nombre.

La instrucción genérica sería *java –Djava.security.policy=registrar.policy Controlador <PUERTO_ServidorConcurrente_HTTP> <IP servicio nombres> <PUERTO servicio nombres> <cantidad maquinas vending> <nombre maquinas>.*

- run_servidor_http.bat contiene:

javac ServidorConcurrente.java

Compilamos el ServidorConcurrente lo que hace que también se compile el HiloServidor.

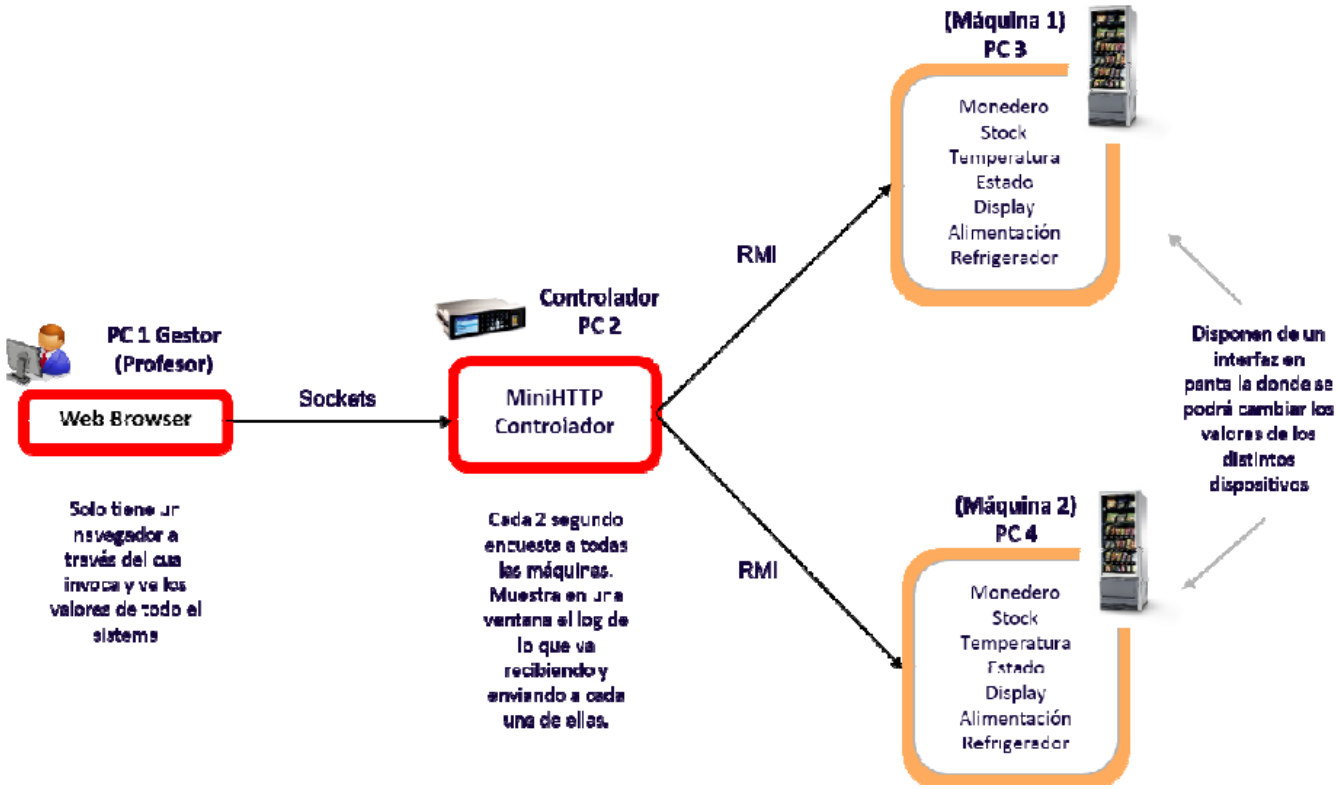
java ServidorConcurrente 127.0.0.1 1234 8080

Ejecutamos el ServidorConcurrente que es el servidor HTTP para que se quede a la escucha de peticiones HTTP del cliente navegador web.

La instrucción genérica sería *java ServidorConcurrente <IP servicio de nombres> <PUERTO servicio de nombres> <PUERTO http>*.

5. Despliegue

En este apartado pondremos un ejemplo de despliegue del sistema siguiendo el modelo de la siguiente figura.



Nota: El .jar debería estar inicialmente en el equipo donde está el cliente RMI (el controlador) y no sería necesarias las compilaciones, solo las ejecuciones.

1. En el **PC 2** dispondríamos del archivo .jar con los ficheros InterfazMaquinaVending.class y MaquinaVending_Stub.class. Entonces ejecutaríamos el script run_servicio_nombres.bat que lanzaría las siguientes instrucciones:

   set CLASSPATH=<ruta de la InterfazMaquinaVending>

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

   Se ha levantado el servicio de nombres para que las máquinas se puedan registrar.

2. En el **PC 3** con el script run_registrar_maquinas.bat se ejecutarían las instrucciones:

   javac InterfazMaquinaVending.java

   set CLASSPATH=<ruta de la InterfazMaquinaVending>

   Ya se ha compilado la interfaz del objeto remoto y especificado donde está.

   javac MaquinaVending.java

   Ya se ha compilado el objeto remoto.

   rmic MaquinaVending

Se han generado los stubs y skeletons para hacer posible la conexión RMI.

jar cvf cliente.jar InterfazMaquinaVending.class MaquinaVending_Stub.class

Reunidos en el .jar la interfaz y el stub.

javac Registro.java

Compilada la clase Registro.

java -Djava.security.policy=registrar.policy Registro <IP_PC_2> <PUERTO_PC_2> Maquina1

Por ejemplo registramos tan solo una máquina de nombre Maquina1.

3. En el **PC 4** Hacemos lo mismo que en el paso 2 pero la última instrucción por ejemplo sería:

   java -Djava.security.policy=registrar.policy Registro <IP_PC_2> <PUERTO_PC_2> Maquina2

   Registramos la máquina de nombre Maquina2.

4. En el **PC 2** ejecutamos el script run_controlador.bat:

   set CLASSPATH=<ruta y nombre del archivo .jar>;<ruta donde esta el controlador>

   javac Controlador.java

   Compilamos el controlador.

   java -Djava.security.policy=registrar.policy Controlador <PUERTO_PC_2> <IP_PC_2> <PUERTO_PC_2> <2> Maquina1 Maquina2

   Se ha ejecutado el controlador. Ya se habrá creado el index y las demás páginas, además se estarán actualizando sus datos cada 2 segundos.

5. En el **PC 2** ejecutamos el script run_servidor_http.bat:

   javac ServidorConcurrente.java

   Compilado el ServidorConcurrente y HiloServidor.

   java ServidorConcurrente <IP_PC_2> <PUERTO_PC_2> <PUERTO_HTTP>

   Ejecutamos el ServidorConcurrente que es el servidor HTTP el cual está preparado para escuchar peticiones del cliente, el navegador del PC 1.

6. En el **PC 1** introducimos en un navegador la siguiente URL:

   IP_PC_2:PUERTO_HTTP/index.html

   Se mostrará el index que el controlador ha guardado en el directorio donde se encuentra el HiloServidor el cual lee el archivo y lo envía por HTTP.