SD	**Sistemas Distribuidos**
19/20	Práctica no guiada
	Payment Gateway **Sockets**

Preámbulo

El objetivo de esta práctica es que los alumnos extiendan y afiancen sus conocimientos sobre el uso de la tecnología de comunicación básica sockets, estudiada 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.

Esta práctica establece el marco inicial de un desarrollo que se ampliará durante el cuatrimestre y que permitirá al alumno 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 pasarela de pagos. Vamos a suponer que somos una organización que se dedica al proporcionar servicios de medios de pago con tarjeta para que los usuarios puedan incluirlos en sus páginas web.

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 va a emplear la tecnología de sockets para conectar las aplicaciones y procesos de diferentes máquinas.

El objetivo es el desarrollo de un sistema que permita desde una página web solicitar los datos de pago de un usuario (nombre del titular, tarjeta, cvv, fecha de caducidad e importe) y enviarlos a una pasarela que redirigirá su petición al procesador correspondiente.

De la misma forma el administrador del procesador podrá solicitar el estado del procesador y cambiarlo (activarlo o desactivarlo).

Peticiones de usuario	Autorización (Auth): Solicita el procesamiento de un pago. Devuelve Ok o Ko.
Peticiones de administrador	Estado (Status): Permite obtener o cambiar el estado del procesador. Límite inferior (Floor limit FL): Permite obtener o cambiar el importe más bajo para la aceptación de un pago. Límite superior (Upper limit UL): Permite obtener o cambiar el importe más alto para la aceptación de un pago.

Tabla 1. Peticiones posibles a los procesadores

Aplicación WEB – Sockets

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

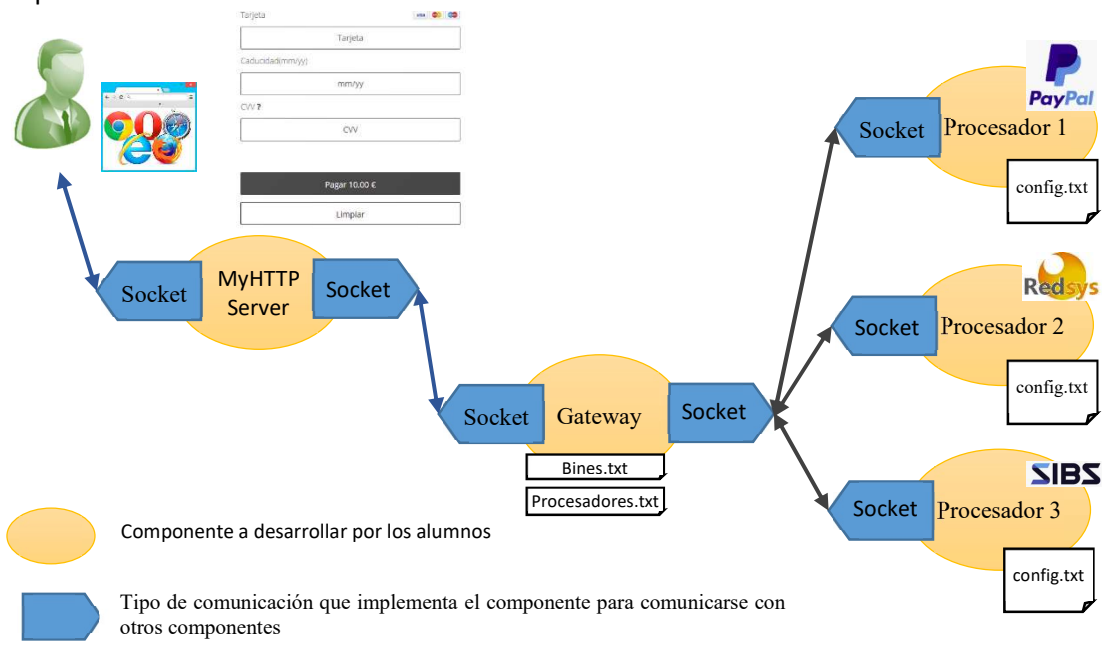


Imagen 1. Esquema conceptual del Sistema software, interconexiones entre los componentes y tipos de interfaces.

Los componentes software que el alumno 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.
- Gateway: este componente se encarga de comunicarse con todos los componentes Procesadores desplegados y acceder a sus métodos bajo demanda del componente MyHTTPServer.

- Procesador_X: componente que implementa la autorización o denegación de transacciones con tarjeta. Utilizará un archivo ".txt" para almacenar los valores de configuración.

Aun cuando los profesores han entregado el código de la práctica guiada en Java, los componentes pueden ser desarrollados en el lenguaje de preferencia del alumno: Java, C/C++ o Python asumiendo el alumno la responsabilidad de su conocimiento.

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, C/C++ o Python 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

Será suficiente con aceptar peticiones realizadas mediante el método GET si bien el alumno puede optar por implementar otras peticiones como PUT o POST a su criterio. 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

En caso de 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 los procesadores para obtener o establecer un valor, con lo que MyHTTPServer deberá conectarse con gateway y hacer la petición adecuada, recibir respuesta y esta respuesta transmitirla al usuario.

Es deseable que el alumno 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 gateway: si no se ha podido establecer una conexión con la pasarela.

Gateway o pasarela

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 **gateway**.

La función de esta pasarela es seleccionar el procesador de pagos y dar acceso a las diferentes funcionalidades que nos ofrecen los procesadores y que son encapsulados a través de componentes de negocio distribuidos haciendo uso de sockets. Para ello, el gateway 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/gateway**SD**/...

deban invocar al gateway. 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 “gatewaySD” deberá invocar a **Gateway**.

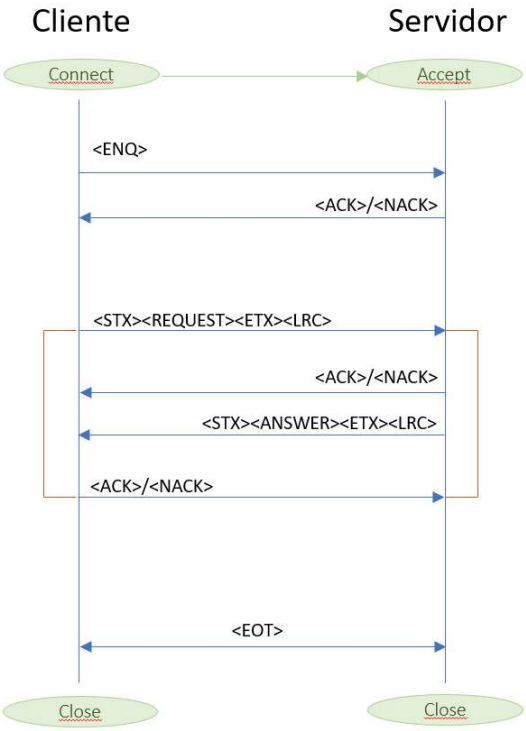
Una vez que el servidor Web detecte este patrón, encapsulará los datos de la petición y se los pasará al gateway. Un ejemplo de encapsulación de datos podrías consistir en la siguiente tabla, ante una petición con la siguiente url:

http://192.168.1.1:3000/gatewaySD/status?proc=1

Objeto Request	Descripción	Ejemplo
ResourcePath	Ruta al recurso a continuación de gatewaySD	Estado
Parameters []	Colección de parámetros incluidos en la petición GET	procesador=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 la pasarela

Esto quiere decir, que el servidor detecta que variables y el procesador al que quiere acceder y realiza una petición a la pasarela. El protocolo de comunicación entre servidor y pasarela y procesadores se ajustará al estándar de empaquetado <STX><DATA><ETX><LRC> que corresponde al siguiente flujo:



Donde:

- <REQUEST> y <ANSWER>: Contienen el mensaje transmitido entre ambos puntos con los campos separados por un carácter determinado:
    - Ej.: <REQUEST>: Código Operación#campo1#...#campo n

      <STX>1#ANGEL FUENTES#4000100020003000#10#123#0224<ETX><LRC> que expresa el código de operación (en este caso 1 significando autorización a modo de ejemplo).

- <LRC>: Se define como el XOR(MESSAGE) byte a byte y sirve para validar que la transmisión del mensaje se ha realizado satisfactoriamente y ha sido recibida de forma correcta por el destinatario el cual responderá al mismo con un <ACK> o <NACK>
-
    - Ej.: XOR(1#ANGEL FUENTES#4000100020003000#10#123#0224)= 1^#^A^N^G^E^L^sp^F^U^E^N^T^E^S^#^4^0^0^0^1^0^0^0^2^0^0^0^3^0^0^ 0^#^1^0^#^1^2^3^#^0^2^2^4 = **8 en Ascii (56 en decimal)**

A partir de la petición recibida, la pasarela debe ser capaz de invocar al componente distribuido (procesador) correcto. Para ello deberá realizar disponer de un archivo de configuración llamado **Bines.txt** donde se especificará a que procesador se debe llamar de acuerdo con el primer dígito

de la tarjeta introducida. Para poder realizar la conexión a cada procesador existirá otro archivo **Procesadores.txt** en el que figurarán la IP y Puerto de cada procesador.

Un ejemplo del archivo Bines.txt puede ser:

```
1#3    'Expresa que si la tarjeta empieza por 1 se enviará al procesador 3
2#3
3#1
4#2
5#1
6#1
7#2
8#2
9#2
```

Y el archivo Procesadores.txt:

```
1#192.168.0.100#9001
2#192.168.0.101#9010
3#192.168.0.101#9010
```

Posteriormente, la pasarela o un componente especializado, debe generar el contenido HTML con la información obtenida de los procesadores 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 "gatewaySD") 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 /gatewaySD/auth?name=AngelFuentes& card=4000100020003000&amount=10&cvv=123&exp=0224 HTTP 1.1"
- El servidor detecta el patrón de recurso dinámico (la subcadena "gatewaySD"), por tanto, es un recurso dinámico.
- El servidor encapsula en un mensaje una petición para gateway indicando que desea procesar el pago de la tarjeta del titular Angel Fuentes con número 4000100020003000 por 10€ con CVV 123 y fecha de caducidad 02/24. Se conecta la pasarela y le envía la petición, quedando a la espera de respuesta.
- La pasarela recibe la petición de servidor, detecta los parámetros de la petición, buscar en su archivo Bines.txt a qué procesador debe enviar las tarjetas que empiezan con un "4" e invoca al componente distribuido "Procesador X" para obtener respuesta (aceptación o denegación de la transacción, fecha y hora, número de autorización e importe autorizado.). La pasarela recibe la respuesta del "Procesador X", con la cual 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 /gatewaySD/Totales?procesador=1 HTTP 1.1”
- El servidor detecta el patrón de recurso dinámico (la subcadena “gatewaySD”), por tanto, es un recurso dinámico.
- El servidor encapsula en un mensaje una petición para gateway indicando que desea obtener los Totales del procesador 1. Se conecta a gateway y le envía la petición, quedando a la espera de respuesta.
- La pasarela 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.

Como se ha descrito al inicio del apartado de especificación de este documento, las peticiones dinámicas a enviar al Gateway que MyHttpServer puede recibir son las siguientes:

Peticiones de usuario	**Autorización (Auth):** Solicita el procesamiento de un pago. Devuelve: Ok (Aceptada, código autorización, fecha y hora, importe autorizado) o Ko (Denegada, causa). Ej.: http://IP:Puerto/gatewaySD/auth?name=AngelFuentes&card=4000100020003000&amount=10&cvv=123&exp=0224 Respuesta: Aceptada. Código de Autorización:0010. 10-02-2020. 10€
Peticiones de administrador	**Estado (Status):** Permite obtener o cambiar el estado del procesador. Ej.: http://IP:Puerto/gatewaySD/status?proc=1 Solicita el estado del procesador 1 Respuesta: Ok. Activo Ej.: http://IP:Puerto/gatewaySD/status?proc=1&set=0 Cambia el estado del procesador 1 a desactivado Respuesta: Ok. Desactivado. El procesador no admite más peticiones.
	Límite inferior (Floor limit Fl): Permite obtener o cambiar el importe más bajo para la aceptación de un pago de un determinado procesador. Ej.: http://IP:Puerto/gatewaySD/fl?proc=1 Solicita el valor del floor limit del procesador 1 Respuesta: Ok. Importe Floor Limite=10€. Ej.: http://IP:Puerto/gatewaySD/fl?proc=1&set=20 Cambia el valor del floor limit del procesador 1 a 20€ Respuesta: Ok. Nuevo Importe límite inferior de aceptación establecido en 20€.
	Límite inferior (Upper limit Ul): Permite obtener o cambiar el importe más bajo para la aceptación de un pago de un determinado procesador.

	Ej.: http://IP:Puerto/gatewaySD/ul?proc=1 Solicita el valor del upper limit del procesador 1 Respuesta: Ok. Importe del límite superior=100€. Ej.: http://IP:Puerto/gatewaySD/ul?proc=1&set=200 Cambia el valor del upper limit del procesador 1 a 200€ Respuesta: Ok. Nuevo Importe límite superior de aceptación establecido en 200€.

Consideraciones sobre Gateway

Como puede verse, la pasarela es un intermediario entre el servidor que atiende las peticiones HTTP y los componentes distribuidos Procesadores. 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 gateway. 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, los Procesadores, por tanto, tras realizar una de sus operaciones, devuelven el resultado (nunca una página web HTML u otro documento), siendo la pasarela el que, al recibir dicha respuesta, compone una página HTML de respuesta que envía al servidor.

El alumno proporcionará la utilidad de tener una URL del servidor que permita obtener un listado de todos los procesadores junto con el acceso a todas sus características de configuración, por ejemplo si el usuario introduce la url: http://IP:puerto/gatewaySD/index , que esta url produzca como resultado una página html con la lista de todos los procesadores y el acceso a cada variable (estado y límites de aceptación). 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) la pasarela son los siguientes:

- Variable no válida: se solicita información de una variable de la estación que no existe, por ejemplo “Totales”.
- El procesador no existe: se solicita información de un procesador que no existe, por ejemplo “procesador=123” si esta no ha sido registrada.
- En general todos los problemas que puedan surgir de introducción de cualquier URL no válida o sintácticamente incorrecta. Se debe evitar que derivado de una URL mal escrita cualquier parte del sistema entre en fallo o bloqueo.
- Igualmente se deberá tener en cuenta las posibles incidencias generadas en cualquier punto del sistema global tales como que un procesador no exista, no responde en tiempo o se ha caído, el Gateway no responde, etc.

Procesadores

Cada componente distribuido encapsulara la funcionalidad de un procesador. Los parámetros de configuración de cada procesador deben poder ser obtenidos en cualquier momento bajo petición, así como debe ser posible modificar su valor desde el usuario. Los valores de configuración serán guardados en un archivo denominado "Config.txt". Así por ejemplo el procesador 1 podrá tener el siguiente archivo:

```
ID=1
Estado=ON
Floor=2
Upper=10000
```

Los valores serán inicialmente establecidos y podrán cambiarse manualmente en cualquier momento desde la petición del navegador según se ha explicado anteriormente. Estos son los valores que deberá devolver Procesador cuando sea invocado uno de sus métodos. Los valores también deben poder ser cambiados 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 procesadores por ejemplo "1", "2" y "3", y cada una de ellos deberá poseer un archivo "Config.txt". Estos archivos pueden haber sido creados manualmente antes de lanzar el sistema o pueden ser creados por defecto al arrancar la aplicación.

Guía mínima de despliegue

Para la correcta evaluación de la práctica es necesario comprobar que la aplicación distribuida solicitada es desplegada 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 la pasarela, 2 componentes de estación meteorológica, proporcionando el siguiente escenario:

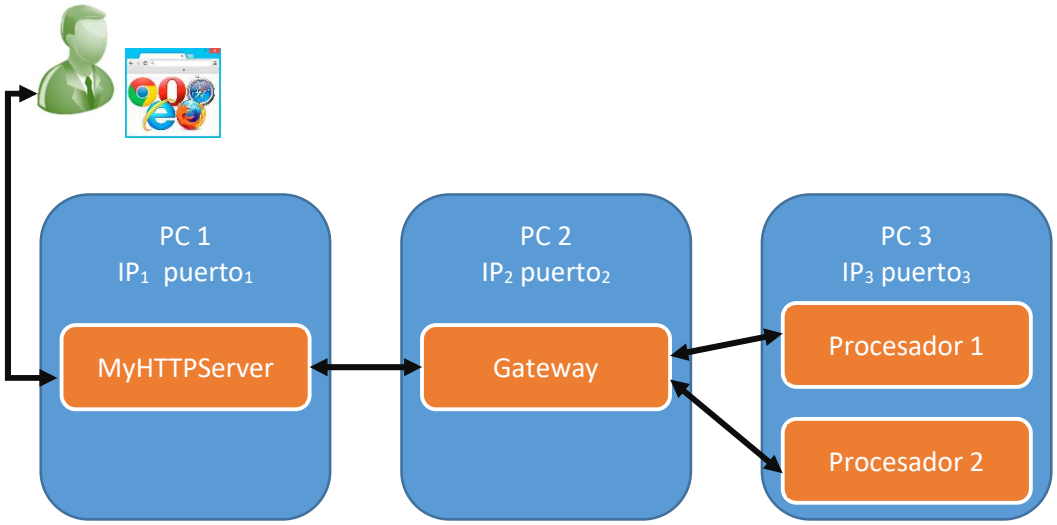


Imagen 2. Escenario físico mínimo para el despliegue de la práctica.

Por supuesto los procesadores podrían ser desplegados 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, la pasarela en otro PC y los procesadores en otro.

Para facilitar al alumno el despliegue se permite el uso de máquinas virtuales, por lo que el alumno 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 los procesadores con su puerto de escucha.
- Iniciar la pasarela indicando puerto de escucha y validando las conexiones a los distintos procesadores según su archivo de configuración.
- Iniciar servidor indicando puerto de escucha del servidor y puerto de conexión al gateway.

Entregables y evaluación

La evaluación de la práctica se realizará en los laboratorios. Se podrá realizar en parejas. Los alumnos deben desplegar por ellos mismos la práctica que resuelve el enunciado anterior. Deben desplegar un sistema completo, es decir, un servidor, un gateway y al menos dos procesadores, todos ellos interconectados entre sí. **Este es el requisito indispensable para poder realizar la corrección**. Además, deben poderse evaluar positiva o negativamente todos los apartados que aparecerán en la Guía de corrección que se entregará a tal propósito. 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. Los alumnos deberán presentar para la evaluación el documento **"Guía de corrección"** cumplimentado para que el profesor pueda validar los apartados implementados.

Los alumnos deberán entregar, además, mediante la funcionalidad de evaluación del UACloud antes de la fecha establecida a su profesor de prácticas una **memoria de prácticas**, con el código fuente y compilados generados, así como 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 los nombres, apellidos y DNI de los alumnos, año académico y el título de la práctica.
- Un informe donde se indique el nombre de los componentes software desarrollados y una descripción de cada uno de ellos, explicando y enviando además el código del servidor, gateway y procesadores que el alumno haya creado.
- 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 la pasarela con todos los procesadores registrados, las páginas de resultados al consultar alguna variable de un procesador o del gateway y las páginas de error implementadas.

Cada profesor de prácticas podrá solicitar a los alumnos cualquier otra evidencia que el profesor considere adecuada para poder formalizar la evaluación.