

 <b>Escuela Politécnica Superior</b> <b>Ingeniería Informática</b> <b>Prácticas de Sistemas Informáticos 2</b>				
<b>Grupo</b>	2362	<b>Práctica</b>	3	<b>Fecha</b>
<b>Alumno/a</b>	Ramos Pedroviejo, Alba			
<b>Alumno/a</b>	Serrano Salas, Nicolás			

## Práctica 2: Seguridad y disponibilidad

### Ejercicio 1:

Preparar 3 máquinas virtuales desde cero (a partir de la VM en moodle) con acceso SSH entre ellas. Esta tarea es necesaria para la correcta gestión del cluster que definiremos en el próximo apartado. Las VMs las denominaremos:

- si2srv01: Dirección IP 10.X.Y.1, 768MB RAM
- si2srv02: Dirección IP 10.X.Y.2, 512MB RAM
- si2srv03: Dirección IP 10.X.Y.3, 512MB RAM

**RECUERDE RANDOMIZAR LAS DIRECCIONES MAC DE CADA COPIA ANTES DE INTENTAR USAR EL NODO.** En la primera máquina (10.X.Y.1), generaremos el par de claves con DSA. A continuación importaremos la clave pública en cada uno de los otros dos nodos (10.X.Y.2 y 10.X.Y.3). Probaremos a acceder por SSH desde .1 a .2 y .3, comprobando que no requiere la introducción de la clave. Obtener una evidencia del inicio remoto de sesión mediante la salida detallada (`ssh -v si2@10.X.Y.2` y `ssh -v si2@10.X.Y.3`). Anote dicha salida en la memoria de prácticas. Revisar y comentar la salida del mandato ssh.

Observamos en las siguientes capturas las salidas de los comandos `ssh -v si2@10.8.6.2` (figura 1) y `ssh -v si2@10.8.6.3` (figura 2). En ellas se puede observar como se intenta acceder mediante el protocolo ssh a las distintas máquinas virtuales y que estas utilizan la clave pública y privada que acabamos de generar en este apartado (se puede observar cómo se van probando diferentes claves existentes hasta quedarse con la que hemos generado nosotros), para no solicitar la contraseña del usuario a la hora de acceder a dicha máquina. Se puede observar cómo se va realizando todo el proceso de detección del tipo de clave y cómo se realiza la autenticación con la misma.

```

si2@si2srv01:~$ ssh -v si2@10.8.6.2
OpenSSH_5.3p1 Debian-3ubuntu7, OpenSSL 0.9.8k 25 Mar 2009
debug1: Reading configuration data /etc/ssh/ssh_config
debug1: Applying options for *
debug1: Connecting to 10.8.6.2 [10.8.6.2] port 22.
debug1: Connection established.
debug1: identity file /home/si2/.ssh/identity type -1
debug1: identity file /home/si2/.ssh/id_rsa type -1
debug1: identity file /home/si2/.ssh/id_dsa type 2
debug1: Checking blacklist file /usr/share/ssh/blacklist.DSA-1024
debug1: Checking blacklist file /etc/ssh/blacklist.DSA-1024
debug1: Remote protocol version 2.0, remote software version OpenSSH_5.3p1 Debian-3ubuntu7
debug1: match: OpenSSH_5.3p1 Debian-3ubuntu7 pat OpenSSH*
debug1: Enabling compatibility mode for protocol 2.0
debug1: Local version string SSH-2.0-OpenSSH_5.3p1 Debian-3ubuntu7
debug1: SSH2_MSG_KEXINIT sent
debug1: SSH2_MSG_KEXINIT received
debug1: kex: server->client aes128-ctr hmac-md5 none
debug1: kex: client->server aes128-ctr hmac-md5 none
debug1: SSH2_MSG_KEX_DH_GEX_REQUEST(1024<1024<8192) sent
debug1: expecting SSH2_MSG_KEX_DH_GEX_GROUP
debug1: SSH2_MSG_KEX_DH_GEX_INIT sent
debug1: expecting SSH2_MSG_KEX_DH_GEX_REPLY
debug1: Host '10.8.6.2' is known and matches the RSA host key.
debug1: Found key in /home/si2/.ssh/known_hosts:1
debug1: ssh_rsa_verify: signature correct
debug1: SSH2_MSG_NEWKEYS sent
debug1: expecting SSH2_MSG_NEWKEYS
debug1: SSH2_MSG_NEWKEYS received
debug1: SSH2_MSG_SERVICE_REQUEST sent
debug1: SSH2_MSG_SERVICE_ACCEPT received
debug1: Authentications that can continue: publickey,password
debug1: Next authentication method: publickey
debug1: Trying private key: /home/si2/.ssh/identity
debug1: Trying private key: /home/si2/.ssh/id_rsa
debug1: Offering public key: /home/si2/.ssh/id_dsa
debug1: Server accepts key: pkalg ssh-dss blen 433
debug1: read PEM private key done: type DSA
debug1: Authentication succeeded (publickey).
debug1: channel 0: new [client-session]
debug1: Requesting no-more-sessions@openssh.com
debug1: Entering interactive session.
debug1: Sending environment.
debug1: Sending env LANG = C
Linux si2srv02 2.6.32-33-generic #72-Ubuntu SMP Fri Jul 29 21:08:37 UTC 2011 i686 GNU/Linux
Ubuntu 10.04.3 LTS

```

**Figura 1: salida de ssh -v si2@10.8.6.2**

```

si2@si2srv01:~$ ssh -v si2@10.8.6.3
OpenSSH_5.3p1 Debian-3ubuntu7, OpenSSL 0.9.8k 25 Mar 2009
debug1: Reading configuration data /etc/ssh/ssh_config
debug1: Applying options for *
debug1: Connecting to 10.8.6.3 [10.8.6.3] port 22.
debug1: Connection established.
debug1: identity file /home/si2/.ssh/identity type -1
debug1: identity file /home/si2/.ssh/id_rsa type -1
debug1: identity file /home/si2/.ssh/id_dsa type 2
debug1: Checking blacklist file /usr/share/ssh/blacklist.DSA-1024
debug1: Checking blacklist file /etc/ssh/blacklist.DSA-1024
debug1: Remote protocol version 2.0, remote software version OpenSSH_5.3p1 Debian-3ubuntu7
debug1: match: OpenSSH_5.3p1 Debian-3ubuntu7 pat OpenSSH*
debug1: Enabling compatibility mode for protocol 2.0
debug1: Local version string SSH-2.0-OpenSSH_5.3p1 Debian-3ubuntu7
debug1: SSH2_MSG_KEXINIT sent
debug1: SSH2_MSG_KEXINIT received
debug1: kex: server->client aes128-ctr hmac-md5 none
debug1: kex: client->server aes128-ctr hmac-md5 none
debug1: SSH2_MSG_KEX_DH_GEX_REQUEST(1024<1024<8192) sent
debug1: expecting SSH2_MSG_KEX_DH_GEX_GROUP
debug1: SSH2_MSG_KEX_DH_GEX_INIT sent
debug1: expecting SSH2_MSG_KEX_DH_GEX_REPLY
debug1: Host '10.8.6.3' is known and matches the RSA host key.
debug1: Found key in /home/si2/.ssh/known_hosts:2
debug1: ssh_rsa_verify: signature correct
debug1: SSH2_MSG_NEWKEYS sent
debug1: expecting SSH2_MSG_NEWKEYS
debug1: SSH2_MSG_NEWKEYS received
debug1: SSH2_MSG_SERVICE_REQUEST sent
debug1: SSH2_MSG_SERVICE_ACCEPT received
debug1: Authentications that can continue: publickey,password
debug1: Next authentication method: publickey
debug1: Trying private key: /home/si2/.ssh/identity
debug1: Trying private key: /home/si2/.ssh/id_rsa
debug1: Offering public key: /home/si2/.ssh/id_dsa
debug1: Server accepts key: pkalg ssh-dss blen 433
debug1: read PEM private key done: type DSA
debug1: Authentication succeeded (publickey).
debug1: channel 0: new [client-session]
debug1: Requesting no-more-sessions@openssh.com
debug1: Entering interactive session.
debug1: Sending environment.
debug1: Sending env LANG = C
Linux si2srv03 2.6.32-33-generic #72-Ubuntu SMP Fri Jul 29 21:08:37 UTC 2011 i686 GNU/Linux
Ubuntu 10.04.3 LTS

```

**Figura 2: salida de ssh -v si2@10.8.6.3**

## Ejercicio 2:

Realizar los pasos del apartado 4 con el fin de obtener una configuración válida del cluster SI2Cluster, con la topología indicada de 1 DAS y 2 nodos SSH de instancias. Inicie el cluster. Liste las instancias del cluster y verifique que los pids de los procesos Java (JVM) correspondientes están efectivamente corriendo en cada una de las dos máquinas virtuales. Adjunte evidencias a la memoria de la práctica.

Hemos verificado que no hay ningún proceso Java en ejecución en la máquinas 2 y 3 (figura 3). Podemos ver que el único proceso Java que aparece al hacer grep es el del propio comando, pero hemos comprobado con ps -afe1 que no haya ninguno.

```
0 S si2      1540  1539  0  80  0 -  1136 wait    10:54 pts/0    00:00:00 -bash
0 R si2      1545  1540  0  80  0 -   609 -    10:54 pts/0    00:00:00 ps -a
si2@si2srv02:~$ ps -afe1 | grep java
0 S si2      1551  1540  0  80  0 -   465 pipe_w 10:55 pts/0    00:00:00 grep java
si2@si2srv02:~$
```

```
si2@si2srv03:~$ ps -afe1 | grep java
0 S si2      1513  1507  0  80  0 -   465 pipe_w 10:59 tty1    00:00:00 grep
java
si2@si2srv03:~$
```

**Figura 3: salida de ps en las máquinas virtuales 2 y 3 para verificar que no haya procesos java ejecutándose**

En la siguiente figura observamos cómo se han creado ambos nodos, Node01 y Node02, en las máquinas 10.8.6.2 y 10.8.6.3 respectivamente.

```
si2@si2srv01:~$ asadmin --user admin --passwordfile /opt/SI2/passwordfile create-node-ssh --sshuser si2 --nodehost 10.8.6.2 --nodedir /opt/glassfish4
Node01
Command create-node-ssh executed successfully.
si2@si2srv01:~$ asadmin --user admin --passwordfile /opt/SI2/passwordfile create-node-ssh --sshuser si2 --nodehost 10.8.6.3 --nodedir /opt/glassfish4
Node01
remote failure: Configuration not added. A Node instance with a "Node01" name already exists in the configuration
Command create-node-ssh failed.
si2@si2srv01:~$ asadmin --user admin --passwordfile /opt/SI2/passwordfile create-node-ssh --sshuser si2 --nodehost 10.8.6.3 --nodedir /opt/glassfish4
Node02
Command create-node-ssh executed successfully.
si2@si2srv01:~$ asadmin --user admin --passwordfile /opt/SI2/passwordfile list-nodes
localhost:domain1 CONFIG
Node01 SSH 10.8.6.2
Node02 SSH 10.8.6.3
Command list-nodes executed successfully.
si2@si2srv01:~$
```

**Figura 4: proceso de creación de nodos desde la máquina 1 y listado de los mismos tras crearlos**

A continuación se observa, tanto por terminal como por glassfish, que ambos nodos se han creado correctamente y se puede hacer un ping a cada uno:

```

Archivo Editar Ver Buscar Terminal Ayuda
st2@st2srv01:~$ asadmin --user admin --passwordfile /opt/SI2/passwordfile
ping-node-ssh Node01
Successfully made SSH connection to node Node01 (10.8.6.2)
Command ping-node-ssh executed successfully.
st2@st2srv01:~$ asadmin --user admin --passwordfile /opt/SI2/passwordfile
ping-node-ssh Node02
Successfully made SSH connection to node Node02 (10.8.6.3)
Command ping-node-ssh executed successfully.
st2@st2srv01:~$
```

**Figura 5: visualización del ping realizado correctamente a ambos nodos, tanto por Glassfish como por terminal**

A continuación hemos comprobado que todos los hosts se conozcan entre sí, comprobando su fichero /etc/hosts y viendo que tengan los nombres y direcciones de las 3 máquinas. Esto se muestra en la figura 6. Asimismo, podemos comprobar en la terminal de la esquina superior derecha (máquina 1) que el cluster se lista correctamente y que actualmente no está activo porque solo lo hemos creado, de momento.

```

1. Crear el cluster:
asadmin create-cluster SI2Cluster
2. Listarlo:
st2@st2srv01:~$ export AS_ADMIN_USER=admin
st2@st2srv01:~$ export AS_ADMIN_PASSWORDFILE=/opt/SI2/passwordfile
st2@st2srv01:~$ asadmin create-cluster SI2Cluster
Command create-cluster executed successfully.
st2@st2srv01:~$ asadmin list-clusters
SI2Cluster not running
Command list-clusters executed successfully.

Linux st2srv03 2.6.32-33-generic #72-Ubuntu SMP Fri Jul 29 21:08:37 UTC 2011
6 GNU/Linux
Ubuntu 10.04.3 LTS

Welcome to Ubuntu!
 * Documentation: https://help.ubuntu.com/
New release 'precise' available.
Run 'do-release-upgrade' to upgrade to it.

Loading es
st2@st2srv03:~$ ps
 PID TTY      TIME CMD
1396 pts/1    00:00:00 bash
1412 pts/1    00:00:00 ps
st2@st2srv03:~$ cat /etc/hosts
cat: /etc/hosts: No such file or directory
st2@st2srv03:~$ cat /etc/hosts
10.8.6.1 st2srv01
10.8.6.2 st2srv02
10.8.6.3 st2srv03
10.8.6.4 st2srv04
127.0.0.1 localhost

# The following lines are desirable for IPv6 capable hosts
::1 localhost ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
st2@st2srv03:~$ _
```

```

st2@st2srv01:~$ cat /etc/hosts
10.8.6.1 st2srv01
10.8.6.2 st2srv02
10.8.6.3 st2srv03
10.8.6.4 st2srv04
127.0.0.1 localhost

# The following lines are desirable for IPv6 capable hosts
::1 localhost ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
st2@st2srv01:~$ _
```

```

st2@st2srv02:~$ cat /etc/hosts
10.8.6.1 st2srv01
10.8.6.2 st2srv02
10.8.6.3 st2srv03
10.8.6.4 st2srv04
127.0.0.1 localhost

# The following lines are desirable for IPv6 capable hosts
::1 localhost ip6-localhost ip6-loopback
fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
st2@st2srv02:~$ _
```

**Figura 6: ficheros /etc/hosts de cada una de las máquinas.**

**Izquierda: máquina 3.**

**Derecha, inferior: máquina 2.**

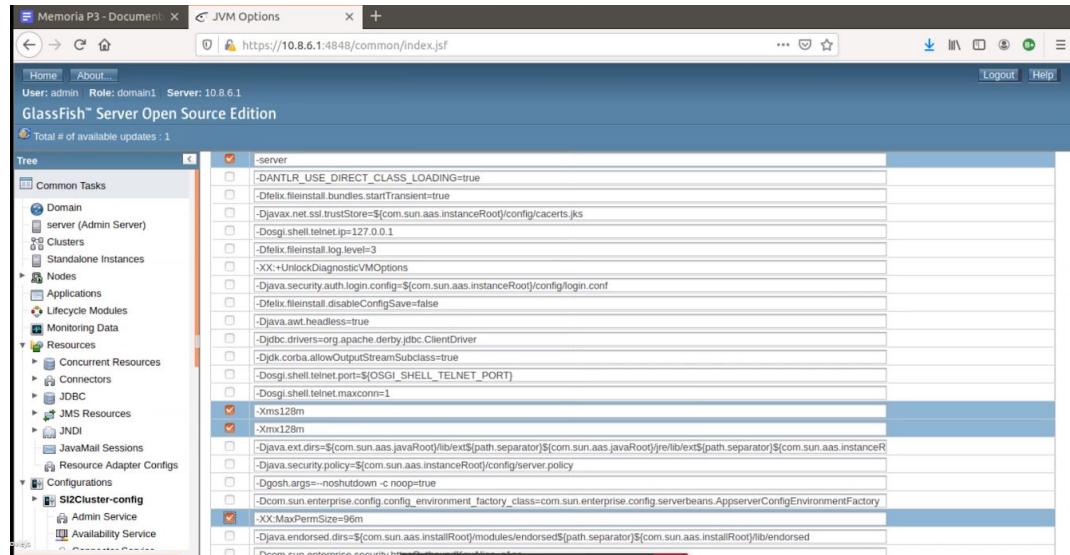
**Derecha, superior: máquina 1 (incluye listado del clúster)**

A continuación, instanciamos ambos nodos y comprobamos que el proceso haya sido correcto:

```
si2@si2srv01:~$ asadmin --user admin --passwordfile /opt/SI2/passwordfile
list-instances -l
Name      Host      Port    Pid   Cluster      State
Instance01 10.8.6.2 24848  --  SI2Cluster  not running
Instance02 10.8.6.3 24848  --  SI2Cluster  not running
Command list-instances executed successfully.
```

**Figura 7: listado de las instancias de los nodos creadas.**

Finalmente, iniciamos el cluster. Se han ajustado correctamente en Glassfish todas las configuraciones solicitadas, en la pestaña JVM Options:



**Figura 8:** configuración añadida en la pestaña JVM Options

En la siguiente imagen podemos observar que en la máquina 10.8.6.2 está efectivamente corriendo uno de los nodos.

```
si2@si2srv02:~$ ps aefl | grep java
0 1000 2228 2223 20 0 2220 760 - R pts/0 0:00 \_ps aefl TERM=xterm-256color SHELL=/bin/bash XDG_SESSION_COOKIE=edc12e8ab2029e
0ecd027fb94f15fe2f-1587055592.172018-2118143550 SSH_CLIENT=10.10.0.16 47926 22 SSH_TTY=/dev/pts/0 ANT_HOME=/opt/glassfish4/ant USER=s12 MAIL=/var/mail
/s12 PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/opt/glassfish4/glassfish/bin:/opt/glassfish4/glassfish/bootstrap/bin:/opt/glassfish4/antlr/bin:/opt/S12/jakarta-jmeter JVM_ARGS=-Duser.language=en PWD=/home/s12 JAVA_HOME=/usr/lib/jvm/java-7-oracle/ J2EE_HOME=/opt/glassfish4/glassfish LANG=C SHLVL=1 HOME=/home/s12 LOGNAME=s12 SSH_CONNECTION=10.10.0.16 47926 10.8.6.2 22 =/bin/ps
0 1000 2228 2223 20 0 1860 568 pipe_W+ pts/0 0:00 \_grep java TERM=xterm-256color SHELL=/bin/bash XDG_SESSION_COOKIE=edc12e8ab2029e
0ecd027fb94f15fe2f-1587055592.172018-2118143550 SSH_CLIENT=10.10.0.16 47926 22 SSH_TTY=/dev/pts/0 ANT_HOME=/opt/glassfish4/ant USER=s12 MAIL=/var/mail
/s12 PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/opt/glassfish4/glassfish/bin:/opt/glassfish4/glassfish/bootstrap/bin:/opt/glassfish4/antlr/bin:/opt/S12/jakarta-jmeter JVM_ARGS=-Duser.language=en PWD=/home/s12 JAVA_HOME=/usr/lib/jvm/java-7-oracle/ J2EE_HOME=/opt/glassfish4/glassfish LANG=C SHLVL=1 HOME=/home/s12 LOGNAME=s12 SSH_CONNECTION=10.10.0.16 47926 10.8.6.2 22 =/bin/grep
si2@si2srv02:~$
```

**Figura 9:** procesos Java ejecutándose en la máquina 2

En la siguiente imagen podemos observar que en la máquina 10.8.6.3 está efectivamente corriendo el otro de los nodos.

```
s1@si2srv03:~$ ps aefl | grep java
0 1000 2152 2147 20 0 2220 760 - R+ pts/0 0:00 \_ ps aefl TERM=xterm-256color SHELL=/bin/bash XDG_SESSION_COOKIE=edc12e8ab2029e
0xcdcd027fb94f15fe2-1587055732.875026-665645404 SSH_CLIENT=10.10.0.16 46546 22 SSH_TTY=/dev/pts/0 ANT_HOME=/opt/glassfish4/ant USER=s12 MAIL=/var/mail
s12 PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/bin:/usr/games:/opt/glassfish4/glassfish/bin:/opt/glassfish4/glassfish/bin:/opt/glassfish4/ant/bin:/opt/SI2/jakarta-jmeter JVM_ARGS=-Duser.language=en PWD=/home/s12 JAVA_HOME=/usr/lib/jvm/java-7-oracle/ J2EE_HOME=/opt/glassfish4/glassfish LANG=C SHLVL=1 HOME=/home/s12 LOGNAME=s12 SSH_CONNECTION=10.10.0.16 46546 10.8.6.3 22 _=/bin/ps
0 1000 2153 2147 20 0 1860 572 pipe_w+ pts/0 0:00 \_ grep java TERM=xterm-256color SHELL=/bin/bash XDG_SESSION_COOKIE=edc12e8ab2029e
0xcdcd027fb94f15fe2-1587055732.875026-665645404 SSH_CLIENT=10.10.0.16 46546 22 SSH_TTY=/dev/pts/0 ANT_HOME=/opt/glassfish4/ant USER=s12 MAIL=/var/mail
s12 PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/bin:/usr/games:/opt/glassfish4/glassfish/bin:/opt/glassfish4/glassfish/bin:/opt/glassfish4/ant/bin:/opt/SI2/jakarta-jmeter JVM_ARGS=-Duser.language=en PWD=/home/s12 JAVA_HOME=/usr/lib/jvm/java-7-oracle/ J2EE_HOME=/opt/glassfish4/glassfish LANG=C SHLVL=1 HOME=/home/s12 LOGNAME=s12 SSH_CONNECTION=10.10.0.16 46546 10.8.6.3 22 _=/bin/grep
s1@si2srv03:~$
```

*Figura 10: procesos Java ejecutándose en la máquina 3*

### Ejercicio 3:

Pruebe a realizar un pago individualmente en cada instancia. Para ello, identifique los puertos en los que están siendo ejecutados cada una de las dos instancias (IPs 10.X.Y.2 y 10.X.Y.3 respectivamente). Puede realizar esa comprobación directamente desde la consola de administración, opción Applications, acción Launch, observando los Web Application Links generados. Realice un único pago en cada nodo. Verifique que el pago se ha anotado correctamente el nombre de la instancia y la dirección IP. Anote sus observaciones (puertos de cada instancia) y evidencias (captura de pantalla de la tabla de pagos).

Primero hemos modificado el bean Pago para incorporar la nueva funcionalidad de obtención de la instancia y la ip, como observamos a continuación:

```
private String instancia;
private String ip;

/**
 * @return instancia del cluster
 */
public String getInstance() {
    return instancia;
}

/**
 * @param instancia la instancia del cluster
 */
public void setInstancia(String instancia) {
    this.instancia = instancia;
}

/**
 * @return ip de la instancia
 */
public String getIp() {
    return ip;
}

/**
 * @param ip ip de la instancia
 */
public void setIp(String ip) {
    this.ip = ip;
}
```

**Figura 11: setters y getters para los nuevos atributos creados en el bean Pago**

En la figura 12 observamos que se han añadido las anteriores funciones en la función creaPago de los servlets ComienzaPago y ProcesaPago. Se ha contemplado la excepción UnknownHostException que podría ocurrir al obtener la dirección del host.

```
private PagoBean creaPago(HttpServletRequest request) {
    PagoBean pago = new PagoBean();
    pago.setIdTransaccion(request.getParameter(PARAM_ID_TRANSACCION));
    pago.setIdComercio(request.getParameter(PARAM_ID_COMERCIO));
    double impd=-1.0;
    try {
        impd = Double.parseDouble(request.getParameter(PARAM_IMPORTE));
        pago.setInstancia(System.getProperty("com.sun.aas.instanceName"));
        pago.setIp(java.net.InetAddress.getLocalHost().getHostAddress());
    } catch (NumberFormatException e) {
        impd = -1.0;
    } catch (NullPointerException e) {
        impd = -1.0;
    } catch(UnknownHostException e){
        pago.setIp("localhost");
    }
    pago.setImporte(impd);
    pago.setRutaRetorno(request.getParameter(PARAM_RUTA_RETORNO));

    return pago;
}
```

**Figura 12: incorporación de la nueva funcionalidad de IP e Instancia en los servlets ProcesaPago y ComienzaPago (en ambos se han añadido las mismas líneas de código)**

También se ha modificado en VisaDAO la forma en la que insertamos el pago para que incluya ambos parámetros, tanto sin prepared statement (figura 14) como con él (figuras 13 y 15).

```
private static final String INSERT_PAGOS_QRY =
    "insert into pago(" +
    "idTransaccion,importe,idComercio,numeroTarjeta, instancia, ip)" +
    " values (?,?,?,?,?,?)";
```

**Figura 13: nueva query para insertar pagos usando prepared statements e incorporando los nuevos campos**

```
String getQryInsertPago(PagoBean pago) {
    String qry = "insert into pago(" +
        "idTransaccion," +
        "importe,idComercio," +
        "numeroTarjeta, instancia, ip)" +
        " values (" +
        "?,?,?,?," +
        pago.getIdTransaccion() + ",," +
        pago.getImporte() + ",," +
        pago.getIdComercio() + ",," +
        pago.getTarjeta().getNumero() + ",," +
        pago.getInstancia() + ",," +
        pago.getIp() + ",," +
        ")";
    return qry;
}
```

**Figura 14: nueva inserción de pago en la BD para incorporar los nuevos campos (inserción sin prepared statement)**

```
/* TODO Usar prepared statement si
RealisPrepared() == true */
/***************************************/
if (isPrepared() == true) {
    String insert = INSERT_PAGOS_QRY;
    errorLog(insert);
    pstmt = con.prepareStatement(insert);
    pstmt.setString(1, pago.getIdTransaccion());
    pstmt.setDouble(2, pago.getImporte());
    pstmt.setString(3, pago.getIdComercio());
    pstmt.setString(4, pago.getTarjeta().getNumero());
    pstmt.setString(5, pago.getInstancia());
    pstmt.setString(6, pago.getIp());
    ret = false;
    if (!pstmt.execute()
        && pstmt.getUpdateCount() == 1) {
        ret = true;
    }
}
```

**Figura 15: nueva inserción de pago en la BD para incorporar los nuevos campos (inserción utilizando prepared statement)**

A continuación se puede observar que se han modificado los ficheros `postgresql.properties` y `build.properties` como se indica en el apartado 5.2, incorporando nuestras direcciones IP (10.8.6.1), el nuevo nombre P3 y el clúster como target.

```

build.properties
1 # Propiedades de despliegue de aplicacion de Visa
2 nombre=P3
3 build=${basedir}/build
4
5
6 dist=${basedir}/dist
7
8
9 src=${basedir}/src
10
11
12 web=${basedir}/web
13
14
15 paquete=ssii2
16 war=${nombre}.war
17
18
19
20 asadmin=${as.home}/bin/asadmin
21 as.home=${env.J2EE_HOME}
22 as_llb=${as.home}/llb
23 as_user=admin
24 as.host=10.8.6.1
25
26 as.port=4848
27 as.passwordfile=${basedir}/passwordfile
28 as.target=SI2Cluster
29

postgresql.properties
1 # Propiedades de la BD postgresql
2
3 # Parametros propios de postgresql
4 db.name=visa
5 db.user=alumnodb
6 db.password=*****
7 db.port=5432
8 db.host=10.8.6.1
9 # Recursos y pools asociados
10 db.pool.name=VisaPool
11 db.jdbc.resource.name=jdbc/VisaDB
12 db.url=jdbc:postgresql://${db.host}:${db.port}/${db.name}
13 db.client.host=10.8.6.1
14 db.client.port=4848
15
16 db.delimiter=
17 db.driver=org.postgresql.Driver
18 db.datasource=org.postgresql.ds.PGConnectionPoolDataSource
19 db.username=SQL92
20
21 # Herramientas
22 db.createdb=/usr/bin/createdb
23 db.dropdb=/usr/bin/dropdb
24
25 # Scripts de creacion / borrado
26 db.create/src=./sql/create.sql
27 db.insert/src=./sql/insert.sql
28 db.delete/src=./sql/drop.sql

```

**Figura 16: ficheros `build.properties` y `postgresql.properties` con las modificaciones indicadas en el apartado 5.2**

Hemos comprobado desde la consola de administración de Glassfish (Applications - Launch) los puertos en los están siendo ejecutadas ambas instancias, teniendo puertos para conexión http y https:

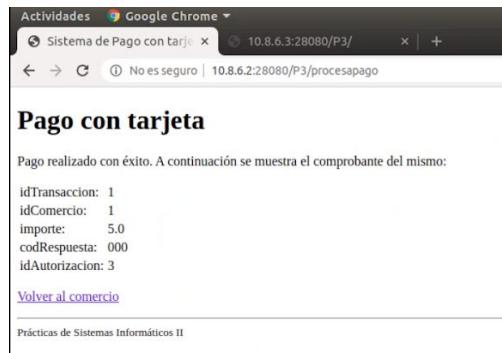
#### Web Application Links

If the server or listener is not running, the link may not work. In this event, check the status of the server instance. After launching the web application, use the browser's Back button to return to this screen.

**Application Name:** P3  
**Links:**  
[Instance01] <http://10.8.6.2:28080/P3>  
[Instance01] <https://10.8.6.2:28181/P3>  
[Instance02] <http://10.8.6.3:28080/P3>  
[Instance02] <https://10.8.6.3:28181/P3>

**Figura 17: direcciones donde se ejecuta cada instancia**

Observamos que en el nodo 10.8.6.2 se realiza correctamente el pago.



**Figura 18: pago correcto desde el la instancia 01**

Observamos que en el nodo 10.6.2.3 también se realiza correctamente el pago.

The screenshot shows a browser window with two tabs: 'Sistema de Pago con tarjeta' and another tab which is partially visible. The main content area is titled 'Pago con tarjeta' and displays a success message: 'Pago realizado con éxito. A continuación se muestra el comprobante del mismo:'. Below this, there is a table with transaction details:

idTransaccion:	2
idComercio:	1
importe:	7.0
codRespuesta:	000
idAutorizacion:	4

At the bottom of the page, there is a link 'Volver al comercio' and a footer note 'Prácticas de Sistemas Informáticos II'.

**Figura 19: pago correcto desde el la instancia 02**

Por último observamos que se han añadido los campos nuevos en la base de datos cuando se guardan los pagos.

	Idautorizacion [PK] serial	Instancia character varying(50)	Ip character varying(50)	Idtransaccion character(16)	codrespuesta character(3)	importe double precision	Idcomercio character(16)	numerotarjeta character(19)	fecha timestamp without time zone
1	3	Instance01	10.8.6.2	1	000	5	1	1111 2222 3333 4444	2020-04-16 11:10:41.946691
2	4	Instance02	10.8.6.3	2	000	7	1	1111 2222 3333 4444	2020-04-16 11:11:54.38951
*									

**Figura 20: base de datos con los dos pagos realizados. Observar los nuevos campos, IP e Instancia, correctamente incorporados.**

## Ejercicio 4:

Para ejecutar este ejercicio primero hemos añadido la configuración del balanceador de carga en la VM 10.8.6.1, indicando IP y puerto de las instancias, los cuales hemos obtenido en el ejercicio anterior.

```
ProxyRequests Off

<Proxy balancer://SI2Cluster>
    BalancerMember http://10.8.6.2:28080 route=Instance01
    BalancerMember http://10.8.6.3:28080 route=Instance02
</Proxy>

<Location /P3>
    Order: allow,deny
    Allow from all
    ProxyPass balancer://SI2Cluster/P3 stickysession=JSESSIONID|jsessionid scolonpathdelim=On
    ProxyPassReverse balancer://SI2Cluster/P3
</Location>

<Location /balancer-manager>
    SetHandler balancer-manager
</Location>

~
```

**Figura 21: fichero de configuración proxy\_balancer.conf de la máquina 1 con la configuración indicada en el enunciado.**

Probar la influencia de jvmRoute en la afinidad de sesión.

1. Eliminar todas las cookies del navegador
2. Sin la propiedad jvmRoute, acceder a la aplicación P3 a través de la URL del balanceador: http://10.X.Y.1/P3
3. Completar el pago con datos de tarjeta correctos
4. Repetir los pagos hasta que uno falle debido a la falta de afinidad de sesión.

A la primera conseguimos que falle el pago.



Figura 22: pago fallido al eliminar cookies y ejecutar un pago sin tener jvmRoute

5. Mostrar la cookie “JSESSIONID” correspondiente a la URL del balanceador.



Figura 23: cookie generada para el pago anterior (sin campo que indique la instancia en la que se ejecuta)

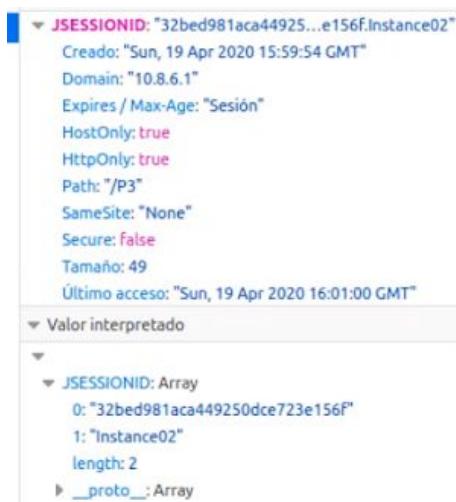
6. Añadir la propiedad “jvmRoute” al cluster y rearrancar el cluster.
7. Eliminar todas las cookies del navegador
8. Acceso a la aplicación P3 a través de la URL del balanceador: <http://10.X.Y.1/P3>
9. Completar el pago con datos de tarjeta correctos. Se pueden repetir los pagos y no fallarán.

Hemos comprobado que ahora funciona varias veces, y a la primera.



*Figura 24: resultado de realizar varios pagos, mostrando siempre pago exitoso.*

#### 10. Mostrar la cookie “JSESSIONID” correspondiente a la URL del balanceador.



*Figura 25: cookie generada para el pago anterior (con campo que indica la instancia en la que se ejecuta en el apartado de “valor interpretado”)*

**Mostrar las pantallas y comentar: las diferencias en el contenido de las cookies respecto a jvmRoute, y cómo esta diferencia afecta a la afinidad y por qué.**

Podemos observar que en la segunda cookie (figura 25) se añade el nombre de la instancia donde se va a ejecutar la petición, mientras que en la primera cookie (figura 23) no tenemos esa información. Además, en la segunda cookie aparece el desplegable de “Valor interpretado”, cosa que en la primera no, y esta además es de mayor tamaño, 49 frente a 32, debido a la información extra que hemos descrito.

Debido a esta diferencia, que viene de haber usado jvmRoute, podemos conocer en qué instancia del clúster se está ejecutando un pago. De esta forma, todos los demás pagos de ese mismo cliente llevarán asociada esa instancia en la cookie, de forma que se ejecuten en la misma instancia. Con esto se facilita la gestión de memoria y el balanceo de carga sin necesidad de reenviar información entre los distintos nodos, sino que se manda el mismo cliente siempre al mismo nodo.

¿Se podría, en general, usar el valor \${com.sun.aas.hostName} para la propiedad jvmRoute, en lugar de \${com.sun.aas.instanceName}?

The screenshot shows a web browser window with the URL [10.8.6.1/P3/procesa](http://10.8.6.1/P3/procesa). The page title is "Pago con tarjeta" and the message is "Pago incorrecto". Below the page, it says "Prácticas de Sistemas Informáticos II". To the right of the browser window is a "Valor" (Value) analysis tool. It shows a cookie named "JSESSIONID" with the value "33dd694b2db390b6...1b2063d.si2srv0". The tool displays various properties of the cookie, such as creation date ("Sun, 19 Apr 2020 16:17:57 GMT"), domain ("10.8.6.1"), and path ("/P3"). It also shows the cookie was last accessed at "Sun, 19 Apr 2020 16:20:33 GMT". A section titled "Valor interpretado" (Interpreted Value) shows the cookie value as an array with two elements: "33dd694b2db390b6bb3541b2063d" and "si2srv02".

**Figuras 26 y 27: resultado de pago incorrecto al realizar un pago con la nueva configuración de jvmRoute, y contenido de la cookie en ese caso**

Podemos observar que la cookie en este caso lleva asociado el nombre de la máquina en la que se ejecuta la petición del cliente, no el nombre de la instancia como en el otro caso. Por eso, **no se debería usar** esto, ya que el cluster solo conoce los nombres de las instancias y sus direcciones IP, no los nombres de las máquinas asociadas a dichas instancias.

### Ejercicio 5:

Probar el balanceo de carga y la afinidad de sesión, realizando un pago directamente contra la dirección del cluster <http://10.X.Y.1/P3> desde distintos ordenadores. Comprobar que las peticiones se reparten entre ambos nodos del cluster, y que se mantiene la sesión iniciada por cada usuario sobre el mismo nodo.

Comentad la información mostrada en la página del Load Balancer Manager.

The screenshot shows two separate browser windows. Both windows have the same URL: <http://10.8.6.1/P3/procesa>. The first window shows a session ID of "34975ca7d1df90694855c817e284.Instance02". The second window shows a session ID of "34a16df1fbe4606ef91d0c59428b.Instance01". Both windows show the same message: "Pago incorrecto". The browser interface includes navigation buttons, a search bar, and a status bar indicating the URL and the current page.

**Figura 28: cookies generadas para dos pagos en dos navegadores distintos**

## Load Balancer Manager for 10.8.6.1

Server Version: Apache/2.2.14 (Ubuntu)  
Server Built: Nov 3 2011 03:31:27

### LoadBalancer Status for balancer://si2cluster

StickySession	Timeout	FailoverAttempts	Method			
JSESSIONID jsessionid 0	1		byrequests			
Worker URL	Route	RouteRedir	Factor Set	Status	Elected To	From
<a href="http://10.8.6.2:28080">http://10.8.6.2:28080</a>	Instance01		1	0 Ok	21	13K 22K
<a href="http://10.8.6.3:28080">http://10.8.6.3:28080</a>	Instance02		1	0 Ok	19	12K 22K

Apache/2.2.14 (Ubuntu) Server at 10.8.6.1 Port 80

Figura 29: estado del balanceador de carga

Hemos realizado las peticiones en dos navegadores distintos, al no poder hacerlo con dos ordenadores distintos, y observamos en la figura 28 que cada una se ejecuta en una instancia diferente (y si volvemos a realizar pagos en cada navegador, se ejecutan en la instancia que diga la cookie, siempre la misma). Además podemos ver en la pantalla del Load Balancer Manager (figura 29) que las peticiones se están distribuyendo de forma equitativa entre ambos nodos, repartiendo la carga correctamente entre ellos.

### Ejercicio 6:

**Comprobación del proceso de fail-over.** Parar la instancia del cluster que haya tenido menos elecciones hasta el momento. Para ello, identificaremos el pid (identificador del proceso java) de la instancia usando las herramientas descritas en esta práctica o el mandato ‘ps –af | grep java’. Realizaremos un kill -9 pid en el nodo correspondiente. Vuelva a realizar peticiones y compruebe (accediendo a la página /balancer-manager y revisando el contenido de la base de datos) que el anterior nodo ha sido marcado como “erróneo” y que todas las peticiones se dirijan al nuevo servidor. Adjunte la secuencia de comandos y evidencias obtenidas en la memoria de la práctica.

Observamos en el balanceador de carga de la figura 29 que la instancia que más de usa es la 01, así que eliminaremos la 02, como observamos en la figura 30 (en la que podemos ver que no está ejecutándose, ya que tiene status Err). Tras realizar diversos pagos, estos van dirigidos a la instancia 01, por lo tanto comprobamos en la misma figura que efectivamente la 02 ya no se usa.

Load Balancer Manager for 10.8.6.1

Server Version: Apache/2.2.14 (Ubuntu)  
Server Built: Nov 3 2011 03:31:27

LoadBalancer Status for balancer://si2cluster

StickySession	Timeout	FailoverAttempts	Method			
JSESSIONID jsessionid 0	1		byrequests			
Worker URL	Route	RouteRedir	Factor Set	Status	Elected To	From
<a href="http://10.8.6.2:28080">http://10.8.6.2:28080</a>	Instance01		1	0 Ok	29	17K 30K
<a href="http://10.8.6.3:28080">http://10.8.6.3:28080</a>	Instance02		1	0 Err	20	12K 22K

Apache/2.2.14 (Ubuntu) Server at 10.8.6.1 Port 80

Figura 30: estado del balanceador de carga tras parar la instancia 02

### Lista de comandos utilizados:

1. Saber el pid del proceso que queremos matar:

```
asadmin --user admin --passwordfile /opt/SI2/passwordfile list-instances -l
```

El comando muestra las instancias del clúster junto con los pid de los procesos que las ejecutan. Nos interesa el pid de la 02, que vemos que es 2876 y que se está ejecutando en la máquina 3 en la figura 31.

```
si2@si2srv01:/etc/apache2/mods-enabled$ asadmin --user admin --passwordfile /opt/SI2/passwordfile list-instances -l
Name      Host     Port   Pid   Cluster   State
Instance01 10.8.6.2 24848 2838  SI2Cluster running
Instance02 10.8.6.3 24848 2876  SI2Cluster running
Command list-instances executed successfully.
```

**Figura 31: visualizar los pid de las instancias**

2. En la máquina virtual correspondiente a dicha instancia (la máquina 3): *kill -9 pid*

```
si2@si2srv03:~$ kill -9 2876
```

**Figura 32: matar el proceso que ejecuta la instancia en una máquina virtual**

3. Volver a ejecutar el primer comando para observar que el estado de dicha instancia ha cambiado de running a not running:

```
asadmin --user admin --passwordfile /opt/SI2/passwordfile list-instances -l
```

```
si2@si2srv01:/etc/apache2/mods-enabled$ asadmin --user admin --passwordfile /opt/SI2/passwordfile list-instances -l
Name      Host     Port   Pid   Cluster   State
Instance01 10.8.6.2 24848 2838  SI2Cluster running
Instance02 10.8.6.3 24848  --    SI2Cluster not running
Command list-instances executed successfully.
```

**Figura 33: visualizar la información de las instancias para verificar que ya no se está ejecutando la 02**

## Ejercicio 7:

Comprobación del proceso de fail-back. Inicie manualmente la instancia detenida en el comando anterior. Verificar la activación de la instancia en el gestor del balanceador. Incluir todas las evidencias en la memoria de prácticas y comentar qué sucede con los nuevos pagos. Consulte los apéndices para información detallada de comandos de gestión individual de las instancias.

Tras haber realizado algunos pagos con la Instance02 parada (como observamos en la figura 34), procedemos a arrancarla.

StickySession	Timeout	FailoverAttempts	Method
JSESSIONID sessionid	0	1	byrequests

Worker URL	Route	RouteRedir	Factor	Set	Status	Elected To	From
http://10.8.6.2:28080	Instance01		1	0	Ok	45	31K 55K
http://10.8.6.3:28080	Instance02		1	0	Err	17	8.0K 15K

**Figura 34: estado del balanceador antes de rearrancar la instancia 02, tras realizar algunos pagos**

Para rearrancarla hemos ido a la consola de glassfish y, tras seleccionar Instance02, hemos clicado start, ya que estaba en el status Stopped. En la figura podemos observar dicha pantalla y observamos que se ha rearrancado correctamente y por eso aparece en estado Running.

Select	Name	LB Weight	Configuration	Node	Status
	Instance01	100	SI2Cluster-config	Node01	<span style="color: green;">Running</span>
	Instance02	100	SI2Cluster-config	Node02	<span style="color: green;">Running</span>

**Figura 35: instancia 02 rearrancada manualmente utilizando Glassfish**

Vamos a proceder a realizar pagos para ver cómo reacciona el balanceador. Dichos pagos se realizarán borrando las cookies entre medias, con el fin de observar cómo distribuye las peticiones el balanceador.

Observamos que el primer pago lo ha enviado a la instancia 02:

StickySession	Timeout	FailoverAttempts	Method
JSESSIONID sessionid 0	1		byrequests

Worker URL	Route	RouteRedir	Factor	Set	Status	Elected	To	From
<a href="http://10.8.6.2:28080">http://10.8.6.2:28080</a>	Instance01		1	0	Ok	46	31K	56K
<a href="http://10.8.6.3:28080">http://10.8.6.3:28080</a>	Instance02		1	0	Ok	19	9.4K	18K

Apache/2.2.14 (Ubuntu) Server at 10.8.6.1 Port 80

**Figura 36: estado del balanceador antes tras realizar un pago después de rearrancar la instancia 02**

El segundo pago se observa que lo ha enviado a la instancia 01:

### Load Balancer Manager for 10.8.6.1

Server Version: Apache/2.2.14 (Ubuntu)  
Server Built: Nov 3 2011 03:31:27

#### LoadBalancer Status for balancer://si2cluster

StickySession	Timeout	FailoverAttempts	Method
JSESSIONID sessionid 0	1		byrequests

Worker URL	Route	RouteRedir	Factor	Set	Status	Elected	To	From
<a href="http://10.8.6.2:28080">http://10.8.6.2:28080</a>	Instance01		1	0	Ok	49	33K	60K
<a href="http://10.8.6.3:28080">http://10.8.6.3:28080</a>	Instance02		1	0	Ok	19	9.4K	18K

Apache/2.2.14 (Ubuntu) Server at 10.8.6.1 Port 80

**Figura 37: estado del balanceador antes tras realizar dos pagos después de rearrancar la instancia 02**

El tercer pago lo ha enviado a la instancia 02:

### Load Balancer Manager for 10.8.6.1

Server Version: Apache/2.2.14 (Ubuntu)  
Server Built: Nov 3 2011 03:31:27

#### LoadBalancer Status for balancer://si2cluster

StickySession	Timeout	FailoverAttempts	Method
JSESSIONID sessionid 0	1		byrequests

Worker URL	Route	RouteRedir	Factor	Set	Status	Elected	To	From
<a href="http://10.8.6.2:28080">http://10.8.6.2:28080</a>	Instance01		1	0	Ok	49	33K	60K
<a href="http://10.8.6.3:28080">http://10.8.6.3:28080</a>	Instance02		1	0	Ok	22	11K	22K

Apache/2.2.14 (Ubuntu) Server at 10.8.6.1 Port 80

**Figura 38: estado del balanceador antes tras realizar tres pagos después de rearrancar la instancia 02**

El cuarto pago lo ha enviado a la instancia 01:

### Load Balancer Manager for 10.8.6.1

Server Version: Apache/2.2.14 (Ubuntu)  
Server Built: Nov 3 2011 03:31:27

#### LoadBalancer Status for balancer://si2cluster

StickySession	Timeout	FailoverAttempts	Method
JSESSIONID sessionid 0	1		byrequests

Worker URL	Route	RouteRedir	Factor	Set	Status	Elected	To	From
<a href="http://10.8.6.2:28080">http://10.8.6.2:28080</a>	Instance01		1	0	Ok	52	35K	64K
<a href="http://10.8.6.3:28080">http://10.8.6.3:28080</a>	Instance02		1	0	Ok	22	11K	22K

Apache/2.2.14 (Ubuntu) Server at 10.8.6.1 Port 80

**Figura 39: estado del balanceador antes tras realizar cuatro pagos después de rearrancar la instancia 02**

Pensábamos que el balanceador funcionaba por probabilidades, es decir, que con mayor probabilidad mandaría una petición a la instancia que llevase más tiempo parada (para así distribuir mejor la carga). Sin embargo, hemos comprobado al realizar los pagos de las figuras 36, 37, 38 y 39 que manda una petición a una instancia y otra a la otra. Esto parece indicar que está siguiendo un algoritmo de Round Robin, y al final es un balanceo correcto, porque está distribuyendo equitativamente las nuevas peticiones que llegan al tener dos instancias (cuando solo había una activa, es lógico que todas las peticiones vayan a ella y no hay nada que balancear, pero cuando se reactiva la otra, no tiene sentido tratar de equilibrar las cargas totales mandando todo a esta última, ya que se estaría infrautilizando la otra).

## Ejercicio 8:

Fallo en el transcurso de una sesión.

- Desde un navegador, comenzar una petición de pago introduciendo los valores del mismo en la pantalla inicial y realizando la llamada al servlet ComienzaPago.
- Al presentarse la pantalla de "Pago con tarjeta", leer la instancia del servidor que ha procesado la petición y detenerla. Se puede encontrar la instancia que ha procesado la petición revisando la cookie de sesión (tiene la instancia como sufijo), el balancer-manager o el server.log de cada instancia.
- Completar los datos de la tarjeta de modo que el pago fuera válido, y enviar la petición.
- Observar la instancia del clúster que procesa el pago, y razonar las causas por las que se rechaza la petición.

Partimos con el balanceador en el siguiente estado:

The screenshot shows the Load Balancer Manager interface for version 10.8.6.1. At the top, it says "Load Balancer Manager for 10.8.6.1". Below that, it shows the server version as Apache/2.2.14 (Ubuntu) and the build date as Nov 3 2011 03:31:27. The main section is titled "LoadBalancer Status for balancer://si2cluster". It shows two workers: Instance01 and Instance02. Both workers have a route of "http://10.8.6.2:28080" and a factor of 1. The status for both is "Ok". The elected worker is Instance01, and the from worker is Instance02. A table below shows the sticky sessions:

StickySession	Timeout	FailoverAttempts	Method
JSESSIONID sessionid 0	1	0	byrequests

At the bottom, it says "Apache/2.2.14 (Ubuntu) Server at 10.8.6.1 Port 80".

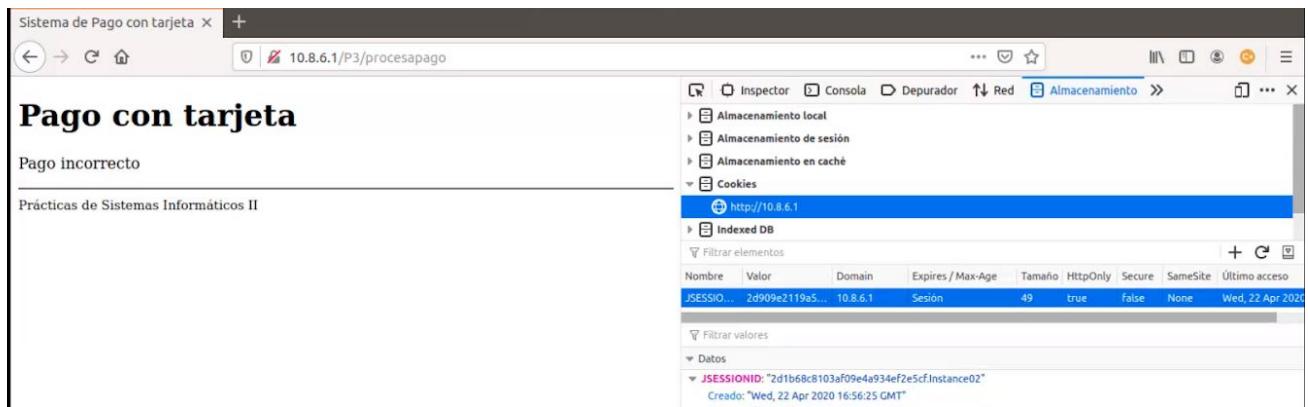
Figura 40: estado del balanceador al comenzar el ejercicio

Tras llamar a ComienzaPago, observamos la cookie y el balanceador y comprobamos que la petición se está ejecutando en Instance02.

The screenshot shows the Load Balancer Manager interface for version 10.8.6.1. The left side is identical to Figure 40. On the right, there is a detailed sidebar for a specific cookie. The cookie ID is "JSESSIONID: 2d1b68c8103af09e4a934ef2e5cf.Instance02". The sidebar shows the creation date as "Wed, 22 Apr 2020 16:56:25 GMT", the domain as "10.8.6.1", and the path as "/3". Other properties shown include "Expires / Max-Age: Sesión", "HostOnly: true", "HttpOnly: true", "SameSite: None", "Secure: false", and "Tamaño: 49". The last access time is also listed as "Wed, 22 Apr 2020 16:56:25 GMT".

Figura 41: estado del balanceador al iniciar una petición de pago y cookie asociada a la petición

En este momento hemos parado Instance02, que es en la que se está procesando el pago, y observamos que al enviar la petición da Pago Incorrecto (figura 42), porque está tratando de ejecutarlo en Instance02, que es la que tenía asociada ese cliente y que está parada.



**Figura 42: fallo al completar el pago, dirigido a la instancia 02, que está detenida**

The screenshot shows the 'Load Balancer Manager for 10.8.6.1' interface. It displays the 'LoadBalancer Status for balancer://si2cluster'. Under 'StickySession', it shows 'JSESSIONID|sessionid 0' with a 'Timeout' of 1 and 'Method' of 'byrequests'. The 'Worker URL' section lists two workers: 'http://10.8.6.2:28080 Instance01' and 'http://10.8.6.3:28080 Instance02'. Both workers have a 'Route' of 1, a 'RouteRedir' of 0, a 'Factor Set' of 53, and an 'Elected To' of '36K 64K' for the first worker, and '25' for the second. The footer indicates 'Apache/2.2.14 (Ubuntu) Server at 10.8.6.1 Port 80'.

**Figura 43: estado del balanceador al enviar el pago tras haber detenido instance02**

Podemos observar que, si se está realizando la petición en una instancia en concreto, si esta se para, nos dará un pago incorrecto. Esto se debe a que intentará seguir ejecutándolo en la instancia a la que fue asociado inicialmente, y como está parada (por eso vemos que incrementa en uno las peticiones a esta instancia) lo mandará a la otra instancia (vemos que las peticiones de esta también se incrementan en uno), pero esta no tiene los datos del usuario y por eso da pago incorrecto. Todo este proceso de incremento de peticiones en cada instancia lo observamos en la figura 43.

## Ejercicio 9:

**Modificar el script de pruebas JMeter desarrollado durante la P2. (P2.jmx) Habilitar un ciclo de 1000 pruebas en un solo hilo contra la IP del cluster y nueva URL de la aplicación: <http://10.X.Y.1/P3>**

**Eliminar posibles pagos previos al ciclo de pruebas. Verificar el porcentaje de pagos realizados por cada instancia, así como (posibles) pagos correctos e incorrectos. ¿Qué algoritmo de reparto parece haber seguido el balanceador? Comente todas sus conclusiones en la memoria de prácticas.**

Primero borramos los pagos de la base de datos y rearrancamos el cluster y el apache para que el balanceador muestre todo a 0 inicialmente.

Abrimos el jmeter, cambiamos los parámetros correspondientes (se adjunta el fichero P3.jmx) y realizamos la ejecución, que observamos en la figura 44 que se hace correctamente, sin ningún pago incorrecto (% Error es 0%, comprobando asimismo en el árbol de resultados que el contenido de la respuesta sea "pago realizado con éxito" y que en la base de datos se almacenen correctamente).

Etiqueta	# Muestras	Media	Mediana	90% Line	95% Line	99% Line	Mín	Máx	% Error	Rendimien...	Kb/sec	Sent KB/se
P3	1000	19	12	20	24	34	8	2912	0,00%	47,2/sec	64,85	0,0
Total	1000	19	12	20	24	34	8	2912	0,00%	47,2/sec	64,85	0,0

Figura 44: resultado de las pruebas de P3.jmx

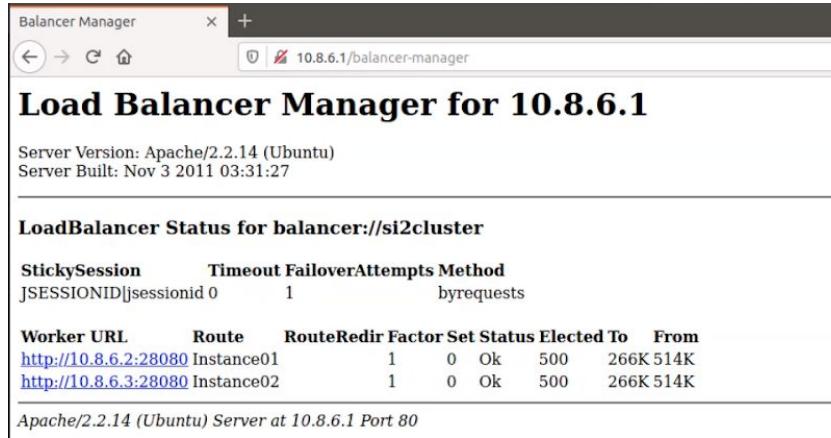


Figura 45: estado del balanceador tras realizar las pruebas de JMeter

Podemos observar en la figura 45 que el reparto de las peticiones ha sido perfecto, por lo que podemos confirmar nuestra teoría del ejercicio 7 de que se está ejecutando un algoritmo **round-robin**, ya que distribuye todas las peticiones de igual manera entre las dos instancias.