		<b>Escuela Politécnica Superior</b> <b>Ingeniería Informática</b> <b>Prácticas de Sistemas Informáticos 2</b>			
<b>Grupo</b>	<b>1391</b>	<b>Práctica</b>	<b>3</b>	<b>Fecha</b>	<b>09/05/2022</b>
<b>Alumno/a</b>	Gil Maroto, Lucía				
<b>Alumno/a</b>	Varela Sánchez, Daniel				
<b>Alumno/a</b>	Martín-Coello Juárez, Guillermo				

## Práctica 3: Seguridad y disponibilidad

### *Ejercicio número 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.

Generamos la clave pública en RSA y la importamos a los nodos 2 y 3. Después, iniciamos sesión remotamente y comprobamos que no se nos solicita una contraseña con la salida del comando `ssh -v si2@10.8.7.2` y del comando `ssh -v 3`.

Observamos en las imágenes como se intenta acceder a las distintas máquinas virtuales mediante el protocolo ssh y estas utilizan las claves públicas y privadas que hemos generado previamente.

```

sl2@sl2srv01:~$ ssh -v sl2@10.8.7.2
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 10.8.7.2 [10.8.7.2] port 22.
debug1: Connection established.
debug1: identity file /home/sl2/.ssh/identity type -1
debug1: identity file /home/sl2/.ssh/id_rsa type 1
debug1: Checking blacklist file /usr/share/ssh/blacklist.RSA-2048
debug1: Checking blacklist file /etc/ssh/blacklist.RSA-2048
debug1: identity file /home/sl2/.ssh/id_dsa type -1
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 '10.8.7.2' is known and matches the RSA host key.
debug1: Found key in /home/sl2/.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/sl2/.ssh/identity
debug1: Offering public key: /home/sl2/.ssh/id_rsa
debug1: Server accepts key: pkalg ssh-rsa blen 277
debug1: read PEM private key done: type RSA
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 sl2srv02 2.6.32-33-generic #72-Ubuntu SMP Fri Jul 29 21:08:37 UTC 2011 i686 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.

Last login: Tue May  3 07:12:59 2022 from 10.8.7.1
Loading es
sl2@sl2srv02:~$

```

```

sl2@sl2srv01:~$ ssh -v sl2@10.8.7.3
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 10.8.7.3 [10.8.7.3] port 22.
debug1: Connection established.
debug1: identity file /home/sl2/.ssh/identity type -1
debug1: identity file /home/sl2/.ssh/id_rsa type 1
debug1: Checking blacklist file /usr/share/ssh/blacklist.RSA-2048
debug1: Checking blacklist file /etc/ssh/blacklist.RSA-2048
debug1: identity file /home/sl2/.ssh/id_dsa type -1
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
The authenticity of host '10.8.7.3 (10.8.7.3)' can't be established.
RSA key fingerprint is d8:ba:48:3c:ad:db:fb:7a:86:e2:a5:f0:5e:3e:66:5e.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '10.8.7.3' (RSA) to the list of known hosts.
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/sl2/.ssh/identity
debug1: Offering public key: /home/sl2/.ssh/id_rsa
debug1: Server accepts key: pkalg ssh-rsa blen 277
debug1: read PEM private key done: type RSA
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 sl2srv03 2.6.32-33-generic #72-Ubuntu SMP Fri Jul 29 21:08:37 UTC 2011 i686 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.

Last login: Tue May  3 07:05:03 2022 from 10.10.5.4
Loading es
sl2@sl2srv03:~$

```

## Ejercicio número 2:

**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) correspondientes2 están efectivamente corriendo en cada una de las dos máquinas virtuales. Adjunte evidencias a la memoria de la práctica.**

Verificamos que no hay ningún proceso Java en ejecución en las máquinas 2 y 3 con el comando `ps -aefl | grep java`.

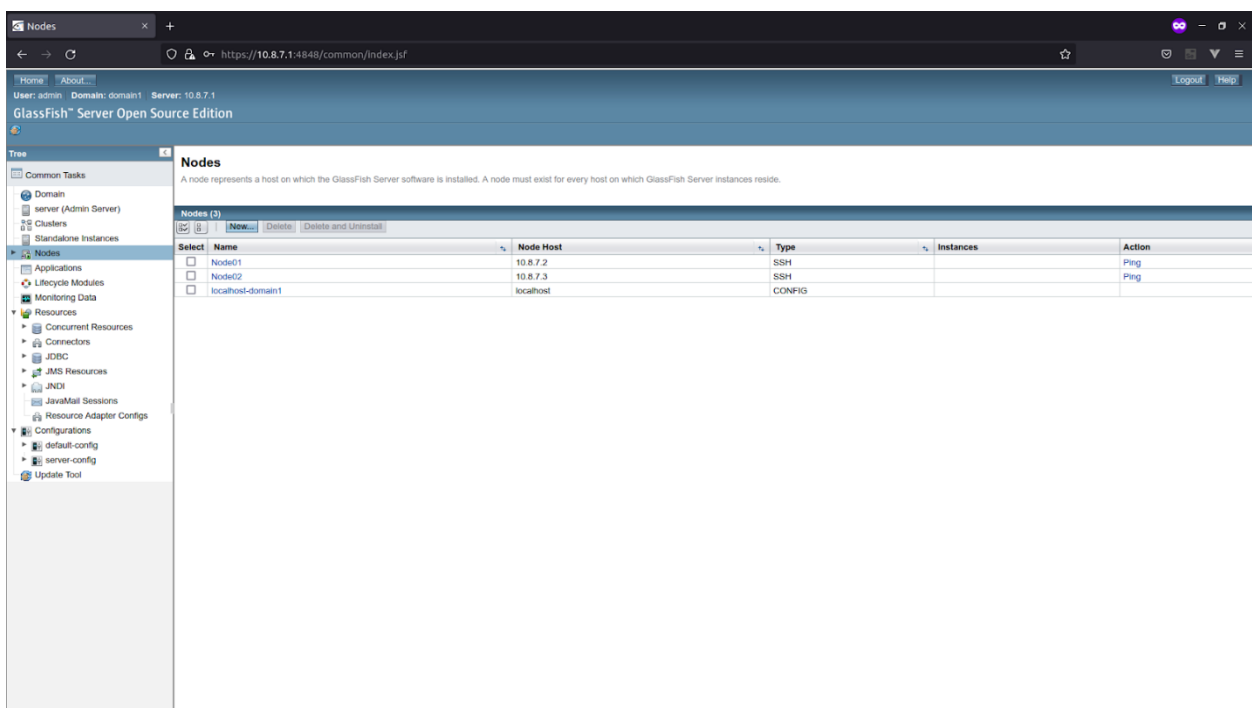
Creamos los nodos con los comandos que se indican en el pdf de la práctica y comprobamos el listado de nodos para ver que se han creado correctamente.

```
si2@si2srv01:~$ asadmin --user admin --passwordfile /opt/SI2/passwordfile list-nodes
localhost-domain1 CONFIG localhost
Node01 SSH 10.8.7.2
Node02 SSH 10.8.7.3
Command list-nodes executed successfully.
```

Hacemos un ping a cada nodo para comprobar que están escuchando.

```
si2@si2srv01:~$ asadmin --user admin --passwordfile /opt/SI2/passwordfile ping-node-ssh Node01
Successfully made SSH connection to node Node01 (10.8.7.2)
Command ping-node-ssh executed successfully.
si2@si2srv01:~$ asadmin --user admin --passwordfile /opt/SI2/passwordfile ping-node-ssh Node02
Successfully made SSH connection to node Node02 (10.8.7.3)
Command ping-node-ssh executed successfully.
```

Comprobamos la creación de los nodos desde el propio dominio de administración.



The screenshot shows the GlassFish Administration Console interface. The left sidebar contains a tree view with categories like Common Tasks, Domain, Clusters, Standalone Instances, Nodes, Applications, Lifecycle Modules, Monitoring Data, Resources, Concurrent Resources, Connectors, J2EE, JMS Resources, JNDI, JavaMail Sessions, Resource Adapter Configs, Configurations, default-config, server-config, and Update Tool. The main content area is titled 'Nodes' and contains a table with the following data:

Select	Name	Node Host	Type	Instances	Action
<input type="checkbox"/>	Node01	10.8.7.2	SSH		Ping
<input type="checkbox"/>	Node02	10.8.7.3	SSH		Ping
<input type="checkbox"/>	localhost-domain1	localhost	CONFIG		

Exportamos las variables de usuario y archivo de contraseñas, creamos el cluster y listamos para comprobar que se ha creado correctamente.

```

si2@si2srv01:~$ export AS_ADMIN_USER=admin
si2@si2srv01:~$ export AS_ADMIN_PASSWORDFILE=/opt/SI2/passwordfile
si2@si2srv01:~$ asadmin create-cluster SI2Cluster
Command create-cluster executed successfully.
si2@si2srv01:~$ asadmin list-clusters
SI2Cluster not running
Command list-clusters executed successfully.
si2@si2srv01:~$ cat /etc/host
cat: /etc/host: No such file or directory
si2@si2srv01:~$ cat /etc/hosts
10.8.7.1 si2srv01
10.8.7.2 si2srv02
10.8.7.3 si2srv03
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

```

Creamos las instancias que nos piden y las listamos para comprobar que están creadas correctamente:

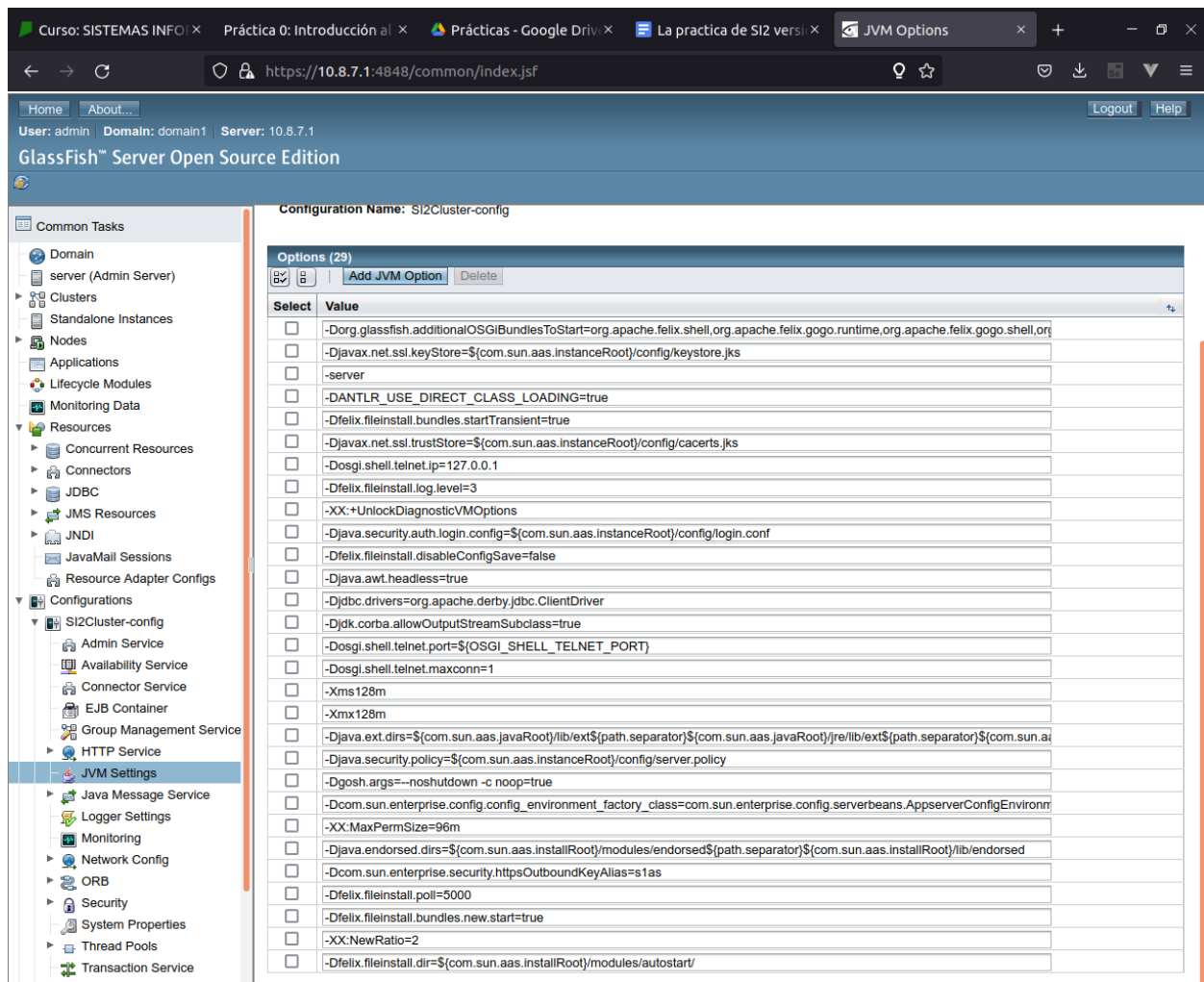
```

si2@si2srv01:~$ asadmin --user admin --passwordfile /opt/SI2/passwordfile create-
instance --cluster SI2Cluster --node Node01 Instance01
Command _create-instance-filesystem executed successfully.
Port Assignments for server instance Instance01:
OSGI_SHELL_TELNET_PORT=26666
JAVA_DEBUGGER_PORT=29009
JMS_PROVIDER_PORT=27676
HTTP_LISTENER_PORT=28080
IIOP_SSL_LISTENER_PORT=23820
ASADMIN_LISTENER_PORT=24848
IIOP_SSL_MUTUALAUTH_PORT=23920
JMX_SYSTEM_CONNECTOR_PORT=28686
HTTP_SSL_LISTENER_PORT=28181
IIOP_LISTENER_PORT=23700
The instance, Instance01, was created on host 10.8.7.2
Command create-instance executed successfully.
si2@si2srv01:~$ asadmin --user admin --passwordfile /opt/SI2/passwordfile create-
instance --cluster SI2Cluster --node Node02 Instance02
Command _create-instance-filesystem executed successfully.
Port Assignments for server instance Instance02:
OSGI_SHELL_TELNET_PORT=26666
JAVA_DEBUGGER_PORT=29009
JMS_PROVIDER_PORT=27676
HTTP_LISTENER_PORT=28080
IIOP_SSL_LISTENER_PORT=23820
ASADMIN_LISTENER_PORT=24848
IIOP_SSL_MUTUALAUTH_PORT=23920
JMX_SYSTEM_CONNECTOR_PORT=28686
HTTP_SSL_LISTENER_PORT=28181
IIOP_LISTENER_PORT=23700
The instance, Instance02, was created on host 10.8.7.3
Command create-instance executed successfully.

si2@si2srv01:~$ asadmin --user admin --passwordfile /opt/SI2/passwordfile list-in
stances -l
Name      Host      Port  Pid  Cluster  State
Instance01 10.8.7.2  24848 --   SI2Cluster  not running
Instance02 10.8.7.3  24848 --   SI2Cluster  not running
Command list-instances executed successfully.
si2@si2srv01:~$ asadmin --user admin --passwordfile /opt/SI2/passwordfile start-c
luster SI2Cluster
Command start-cluster executed successfully.

```

Una vez creadas las instancias del cluster, las modificamos desde la consola de administración como se nos indica en el pdf de la práctica. Nos queda la siguiente configuración:



### Ejercicio número 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 haya 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).

Preparamos la práctica para el despliegue en el cluster.

- Primero, renombramos la carpeta P1-base a P3.
- Copiamos los archivos que nos piden (listado.csv e insert.sql).
- Realizamos las modificaciones sobre el código:

```
-- Tabla con pagos autorizados
-- Siempre vienen precedidos por una transaccion existente
CREATE TABLE pago
(
  -- idAutorizacion se autogenera con cada inserción
  idAutorizacion serial not null,
  idTransaccion char(16) not null,
  codRespuesta char(3) not null default '000',
  importe float not null,
  idComercio char(16) not null,
  numeroTarjeta char(19) not null references tarjeta,
  fecha timestamp not null default current timestamp,
  CONSTRAINT Pago_UC unique(idTransaccion, idComercio),
  PRIMARY KEY (idAutorizacion)
  instancia varchar(50) not null,
  IP varchar(50) not null,
);
```

```
/**
 * @return String instancia
 */
public String getInstancia() {
    return instancia;
}

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

/**
 * @return String ip
 */
public String getIP() {
    return IP;
}

/**
 * @param ip string ip
 */
public void setIP(String ip) {
    this.IP = ip;
}
```

```
-- Tabla con pagos autorizados
-- Siempre vienen precedidos por una transaccion existente
CREATE TABLE pago
(
  -- idAutorizacion se autogenera con cada inserción
  idAutorizacion serial not null,
  instancia varchar(50),
  IP varchar(50),
  idTransaccion char(16) not null,
  codRespuesta char(3) not null default '000',
  importe float not null,
  idComercio char(16) not null,
  numeroTarjeta char(19) not null references tarjeta,
  fecha timestamp not null default current_timestamp,
  CONSTRAINT Pago_UC unique(idTransaccion, idComercio),
  PRIMARY KEY (idAutorizacion)
);

-- INSERT INTO pago(idAutorizacion,idTransaccion, codRespuesta, importe, idComercio, numeroTarjeta)
-- VALUES (NEXTVAL(pago_idAutorizacion_seq), 1,'000', 123.00, '0000000000000000', '1111 2222 3333 4444');
```

```
/**
 * Crea un bean de pago a partir de los parámetros de la petición;
 * @param request objeto de petición;
 * @return bean que contiene el pago a realizar
 * @see ssii2.visa.PagoBean
 */
private PagoBean creaPago(HttpServletRequest request) {
    PagoBean pago = new PagoBean();
    pago.setInstancia(System.getProperty("com.sun.aas.instanceName"));
    pago.setIP(java.net.InetAddress.getLocalHost().getHostAddress());
    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));
    } catch (NumberFormatException e) {
        impd = -1.0;
    } catch (NullPointerException e) {
        impd = -1.0;
    }
    pago.setImporte(impd);
    pago.setRutaRetorno(request.getParameter(PARAM_RUTA_RETORNO));
    return pago;
}
```

```
/**
 * Crea un bean de pago a partir de los parámetros de la petición;
 * @param request objeto de petición;
 * @return bean que contiene el pago a realizar
 * @see ssii2.visa.PagoBean
 */
private PagoBean creaPago(HttpServletRequest request) {
    PagoBean pago = new PagoBean();
    pago.setInstancia(System.getProperty("com.sun.aas.instanceName"));
    pago.setIP(java.net.InetAddress.getLocalHost().getHostAddress());
    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));
    } catch (NumberFormatException e) {
        impd = -1.0;
    } catch (NullPointerException e) {
        impd = -1.0;
    }

    pago.setImporte(impd);
    pago.setRutaRetorno(request.getParameter(PARAM_RUTA_RETORNO));
    return pago;
}
```



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

```
/**
 * getQryInsertPago
 */
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;
}
```

```
/* TODO Usar prepared statement si
isPrepared() == 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;
    }
}
```

```
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=${env.J2EE_HOME}
22 as.lib=${as.home}/lib
23 as.user=admin
24 as.host=10.8.7.1
25
26
27 as.port=4848
28 as.passwordfile=${basedir}/passwordfile
29 as.target=SI2Cluster
30
```

```
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=10.8.7.1
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=10.8.7.1
14 db.client.port=4848
15
16 db.delimiter=;
17 db.driver=org.postgresql.Driver
18 db.datasource=org.postgresql.ds.PGConnectionPoolDataSource
19 db.vendorname=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
29
```

Tras las modificaciones, hemos desplegado la aplicación y buscado las instancias para cada uno de los nodos, desde launch en la consola de administración.

The screenshot shows the GlassFish Server administration console. The top part displays the 'Applications' tab, which includes a table of deployed applications. The table has columns for 'Select', 'Name', 'Deployment Order', 'Status', 'Engines', and 'Action'. The application 'P3' is listed with a deployment order of 100 and a status of 'Enabled on 1 of 1 Target(s)'. The bottom part of the screenshot shows the 'Web Application Links' for the application 'P3', listing two instances with their respective URLs.

Select	Name	Deployment Order	Status	Engines	Action
<input type="checkbox"/>	P3	100	Enabled on 1 of 1 Target(s)	web	Launch   Redeploy   Reload

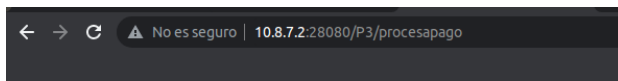
**Web Application Links**

Application Name: P3

Links:

- [Instance01] http://10.8.7.2:28080/P3
- [Instance01] https://10.8.7.2:28181/P3
- [Instance02] http://10.8.7.3:28080/P3
- [Instance02] https://10.8.7.3:28181/P3

Desde cada uno de los nodos realizamos el pago y comprobamos que son correctos:



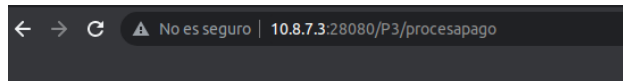
## Pago con tarjeta

Pago realizado con éxito. A continuación se muestra el comprobante del mismo:

idTransaccion: 34  
idComercio: 1  
importe: 10.0  
codRespuesta: 000  
idAutorizacion: 1

[Volver al comercio](#)

Prácticas de Sistemas Informáticos II



## Pago con tarjeta

Pago realizado con éxito. A continuación se muestra el comprobante del mismo:

idTransaccion: 243  
idComercio: 1  
importe: 10.0  
codRespuesta: 000  
idAutorizacion: 2

[Volver al comercio](#)

Prácticas de Sistemas Informáticos II

## Ejercicio número 4:

### Ejercicio 4. Probar la influencia de jvmRoute en la afinidad de sesión.

Para este ejercicio, el primer paso es crear el siguiente archivo:

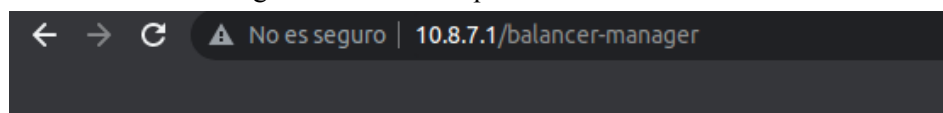
```
si2@si2srv01:/etc/apache2/mods-enabled$ cat /etc/apache2/mods-available/proxy_balancer.conf
ProxyRequests Off

<Proxy balancer://SI2Cluster>
    BalancerMember http://10.8.7.2:28080 route=Instance01
    BalancerMember http://10.8.7.3:28080 route=Instance02
</Proxy>

<Location /P3>
    Order allow,deny
    Allow from all
    ProxyPass balancer://SI2Cluster/P3 stickysession=JSESSIONID|jsessionid scolopathdeli
n=0n
    ProxyPassReverse balancer://SI2Cluster/P3
</Location>

<Location /balancer-manager>
    SetHandler balancer-manager
</Location>
si2@si2srv01:/etc/apache2/mods-enabled$
```

Después, abrimos el balancer manager desde la URL que nos indican:



## Load Balancer Manager for 10.8.7.1

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://10.8.7.2:28080">http://10.8.7.2:28080</a>	Instance01		1	0	Ok	0	0	0
<a href="http://10.8.7.3:28080">http://10.8.7.3:28080</a>	Instance02		1	0	Ok	0	0	0

Apache/2.2.14 (Ubuntu) Server at 10.8.7.1 Port 80



Cambiamos la configuración desactivando jvmRoute:

Configuration Name: SI2Cluster-config

Dynamic Reconfiguration: ☒

Additional Properties (11)

Select	Instance Variable Name	Default Value	Instance Values
<input type="checkbox"/>	ASADMIN_LISTENER_PORT	24848	Instance Values
<input type="checkbox"/>	HTTP_LISTENER_PORT	28080	Instance Values
<input type="checkbox"/>	HTTP_SSL_LISTENER_PORT	28181	Instance Values
<input type="checkbox"/>	IIOP_LISTENER_PORT	23700	Instance Values
<input type="checkbox"/>	IIOP_SSL_LISTENER_PORT	23820	Instance Values
<input type="checkbox"/>	IIOP_SSL_MUTUALAUTH_PORT	23920	Instance Values
<input type="checkbox"/>	JAVA_DEBUGGER_PORT	29009	Instance Values
<input type="checkbox"/>	JMS_PROVIDER_PORT	27676	Instance Values
<input type="checkbox"/>	JMX_SYSTEM_CONNECTOR_PORT	28686	Instance Values
<input type="checkbox"/>	OSGI_SHELL_TELNET_PORT	26666	Instance Values
<input type="checkbox"/>	jvmRoute	\$(com.sun.aas.instanceName)	Instance Values

Para esta prueba, realizamos pagos con datos correctos y el jvmRoute desactivado, hasta que se produce un pago incorrecto debido a la falta de afinidad de sesión. En nuestro caso, falla el primer intento de pago.

The screenshot shows a web browser window with the title "Sistema de Pago con tarjeta". The address bar shows the URL "10.8.7.1/P3/procesapago". The page content displays "Pago con tarjeta" and "Pago incorrecto". Below this, it says "Prácticas de Sistemas Informáticos II".

The browser's developer tools are open, showing the "Almacenamiento" (Storage) tab. The left sidebar lists storage types: "Almacenamiento en caché", "Almacenamiento local", "Almacenamiento de sesión", "Cookies", and "Indexed DB". The "Cookies" section is expanded, showing a cookie for "http://10.8.7.1".

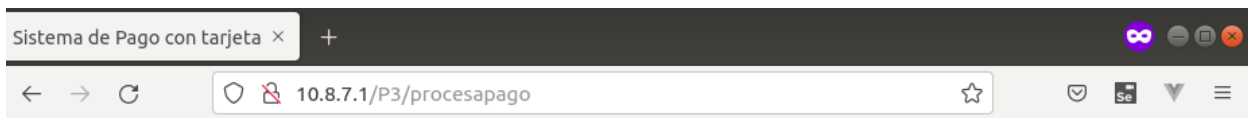
The main area of the developer tools shows a table of storage elements. The selected cookie is:

Nombre	Valor	Domain	Path	Expires / Max-Age
JSESSIO...	Fd8c98fd77b8...	10.8.7.1	/P3	Sesión

The "Datos" (Details) pane for this cookie shows the following information:

- JSESSIONID:** "Fd8c98fd77b80095794b6c52ba64"
- Creado:** "Wed, 04 May 2022 16:12:10 GMT"
- Domain:** "10.8.7.1"
- Expires / Max-Age:** "Sesión"
- HostOnly:** true
- HttpOnly:** true
- Path:** "/P3"
- SameSite:** "None"
- Secure:** false
- Tamaño:** 38
- Último acceso:** "Wed, 04 May 2022 16:13:10 GMT"

Como se puede observar, la cookie obtenida al realizar el pago incorrecto carece de los parámetros que, de otra manera, al añadir el atributo jvmRoute, llevaría consigo la cookie para garantizar el correcto funcionamiento del pago. A continuación, añadimos el atributo y ejecutamos nuevamente un pago.



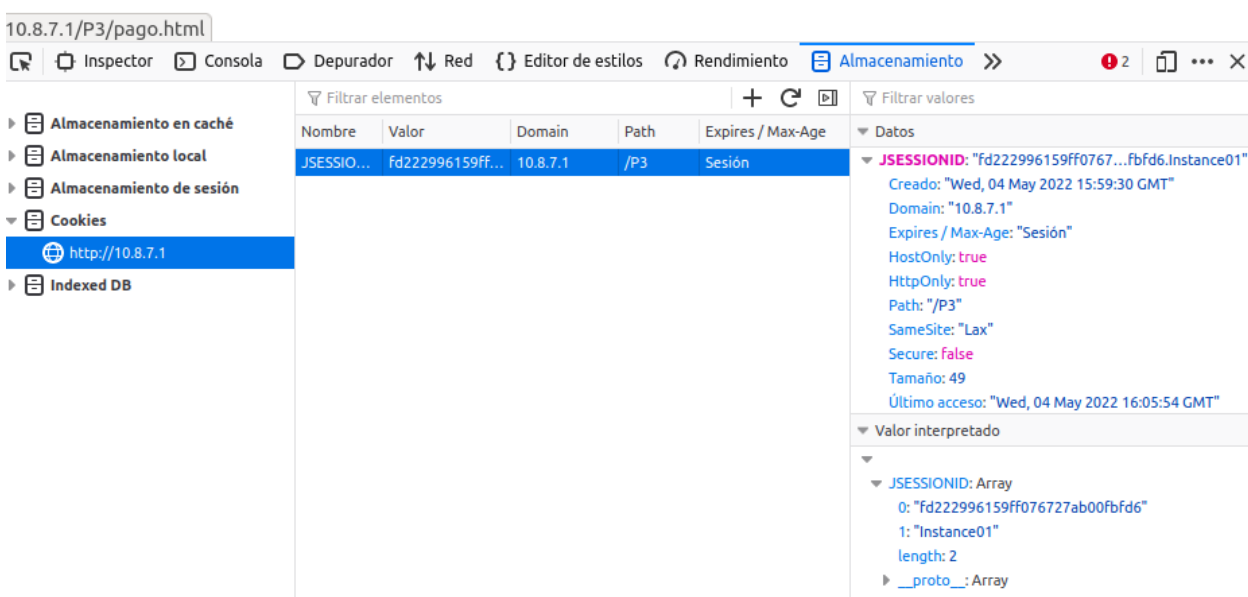
## Pago con tarjeta

Pago realizado con éxito. A continuación se muestra el comprobante del mismo:

idTransaccion: 2341  
idComercio: 1  
importe: 1.0  
codRespuesta: 000  
idAutorizacion: 6

[Volver al comercio](#)

Prácticas de Sistemas Informáticos II



En esta ocasión, como podemos observar, el valor interpretado de la cookie se muestra correctamente, con la respectiva instancia en la que se ejecuta.

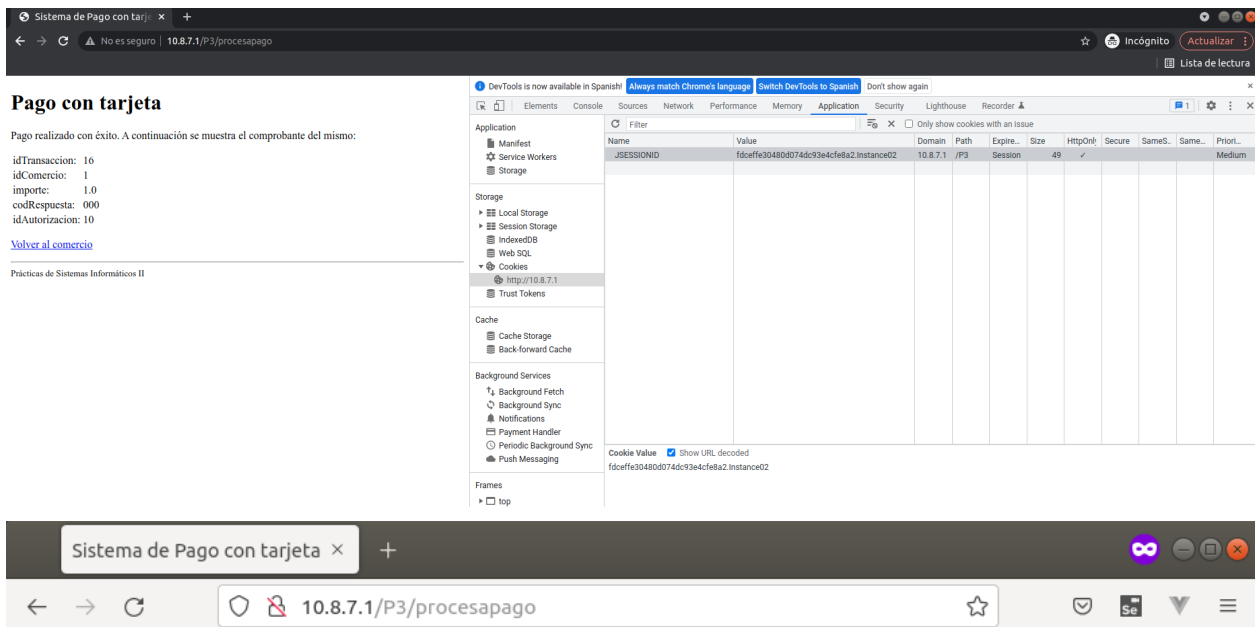
¿Se podría, en general, usar el valor `${com.sun.aas.hostName}` para la propiedad `jvmRoute`, en lugar de `${com.sun.aas.instanceName}`?

No porque como podemos observar en la imagen la cookie lleva asociado el nombre de la instancia no de la máquina (porque lo desconoce). Por tanto, si utilizáramos host name, el navegador no sabría a qué instancia enviar la petición del cliente para ejecutarse.

### Ejercicio número 5:

**Probar el balanceo de carga y la afinidad de sesión, realizando un pago directamente contra la dirección del cluster**

A continuación ejecutamos diferentes pagos desde diferentes lugares para comprobar que se distribuyan correctamente a las diferentes instancias de manera balanceada.



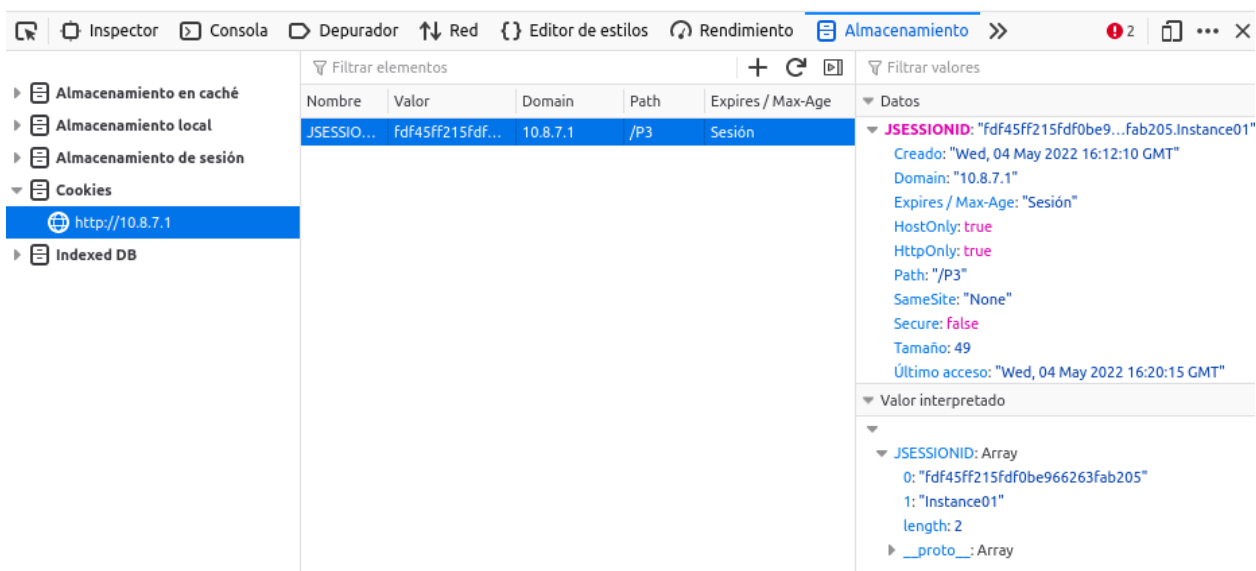
## Pago con tarjeta

Pago realizado con éxito. A continuación se muestra el comprobante del mismo:

idTransaccion: 36  
 idComercio: 1  
 importe: 1.0  
 codRespuesta: 000  
 idAutorizacion: 11

[Volver al comercio](#)

Prácticas de Sistemas Informáticos II



Como podemos observar, las diferentes peticiones de pago se han ido balanceando entre las diferentes instancias disponibles. A continuación revisamos el estado del balanceador para comprobar la calidad de su ejecución.

Sistema de Pago con tarjeta x Balancer Manager x +

← → ↻ No es seguro | 10.8.7.1/balancer-manager

# Load Balancer Manager for 10.8.7.1

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://10.8.7.2:28080">http://10.8.7.2:28080</a>	Instance01		1	0	Ok	26	18K 27K
<a href="http://10.8.7.3:28080">http://10.8.7.3:28080</a>	Instance02		1	0	Ok	24	16K 24K

---

Apache/2.2.14 (Ubuntu) Server at 10.8.7.1 Port 80

Como podemos observar las diferentes peticiones se han distribuido de manera balanceada entre ambas instancias, lo que nos lleva a pensar que la distribución se realiza utilizando el algoritmo round-robin.

### Ejercicio número 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 -aef | 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.

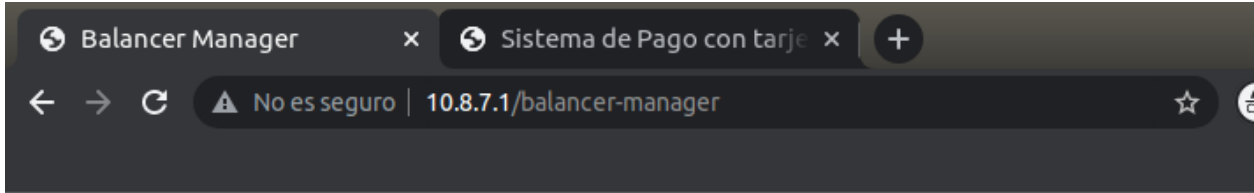
Para comenzar obtenemos la instancia con menos elecciones hasta el momento. Ésta, como se puede observar en el ejercicio anterior es la instancia 2. Primero obtenemos su PID con el siguiente comando:

```
si2@si2srv01:~$ asadmin --user admin --passwordfile /opt/SI2/passwordfile list-instances -l
Name      Host      Port  Pid   Cluster  State
Instance01 10.8.7.2  24848 2368  SI2Cluster  running
Instance02 10.8.7.3  24848 2312  SI2Cluster  running
Command list-instances executed successfully.
si2@si2srv01:~$
```

A continuación, eliminamos el proceso desde su máquina virtual correspondiente:

```
si2@si2srv03:~$ kill -9 2312
si2@si2srv03:~$ ps -aef|grep java
si2      2517  1361  0 09:27 pts/0    00:00:00 grep java
si2@si2srv03:~$
```

Por último, comprobamos que el balancer manager considera errónea la instancia recién parada.



# Load Balancer Manager for 10.8.7.1

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://10.8.7.2:28080">http://10.8.7.2:28080</a>	Instance01		1	0	Ok	30	21K	31K
<a href="http://10.8.7.3:28080">http://10.8.7.3:28080</a>	Instance02		1	0	Err	25	16K	24K

Apache/2.2.14 (Ubuntu) Server at 10.8.7.1 Port 80

Como la instancia 2 ha sido desactivada, se ejecutan pagos para comprobar que ahora no se redirijan a la instancia 2:

**Pago con tarjeta**

Pago realizado con éxito. A continuación se muestra el comprobante del mismo:

idTransaccion: 96  
idComercio: 1  
importe: 1.0  
codRespuesta: 000  
idAutorizacion: 12

[Volver al comercio](#)

Prácticas de Sistemas Informáticos II

Name	Value	Do...	Pa...	Ex...	Siz...	Ht...	Se...	Sa...	Sa...	P...
JSESSIONID	fe7320889e62e05809533198a5b8.Instance01	10...	/P3	Se...	49	✓				M...

Cookie Value ☒ Show URL decoded  
fe7320889e62e05809533198a5b8.Instance01

Como se puede observar, los pagos se redirigen a la instancia 1.

## Ejercicio número 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. Comentar qué sucede con los nuevos pagos.

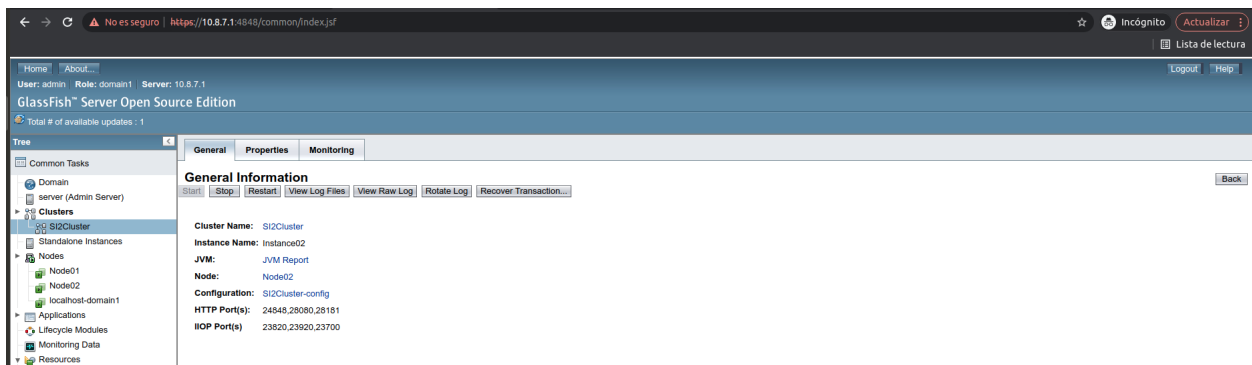
Iniciamos de nuevo la instancia previamente desactivada.

**Nodes**

A node represents a host on which the GlassFish Server software is installed. A node must exist for every host on which GlassFish Server instances reside.

Select	Name	Node Host	Type	Instances	Action
<input type="checkbox"/>	Node01	10.8.7.2	SSH	Instance01 Running	Ping
<input type="checkbox"/>	Node02	10.8.7.3	SSH	Instance02 Running	Ping
<input type="checkbox"/>	localhost-domain1	localhost	CONFIG		





Tras esto hacemos de nuevo varios pagos para comprobar el balance.

Sistema de Pago con tarjeta × +

← → ↻ 10.8.7.1/P3/procesapago ☆

# Pago con tarjeta

Pago realizado con éxito. A continuación se muestra el comprobante del mismo:

idTransaccion: 45  
idComercio: 1  
importe: 1.0  
codRespuesta: 000  
idAutorizacion: 13

[Volver al comercio](#)

Prácticas de Sistemas Informáticos II

Inspector Consola Depurador ↕ Red {} Editor de estilos ⚙ Rendimiento Almacenamiento >> 2

Almacenamiento en caché

Almacenamiento local

Almacenamiento de sesión

Cookies

http://10.8.7.1

Indexed DB

Filtrar elementos

Nombre	Valor	Domain	Path	Expires / Max-Age
JSESSIO...	fed4f6716279...	10.8.7.1	/P3	Sesión

Filtrar valores

Datos

JSESSIONID: "fed4f67162797a24f...69c4e.Instance01"

Creado: "Wed, 04 May 2022 16:12:10 GMT"

Domain: "10.8.7.1"

Expires / Max-Age: "Sesión"

HostOnly: true

HttpOnly: true

Path: "/P3"

SameSite: "None"

Secure: false

Tamaño: 49

Último acceso: "Wed, 04 May 2022 16:35:35 GMT"

Valor interpretado

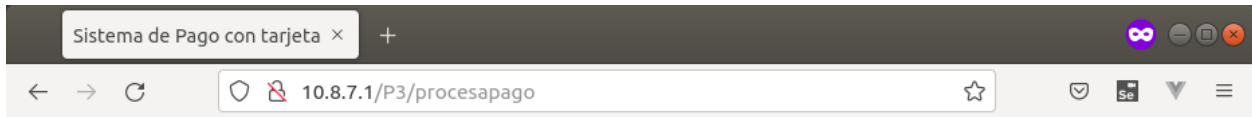
JSESSIONID: Array

0: "fed4f67162797a24ff9a44069c4e"

1: "Instance01"

length: 2

\_\_proto\_\_: Array



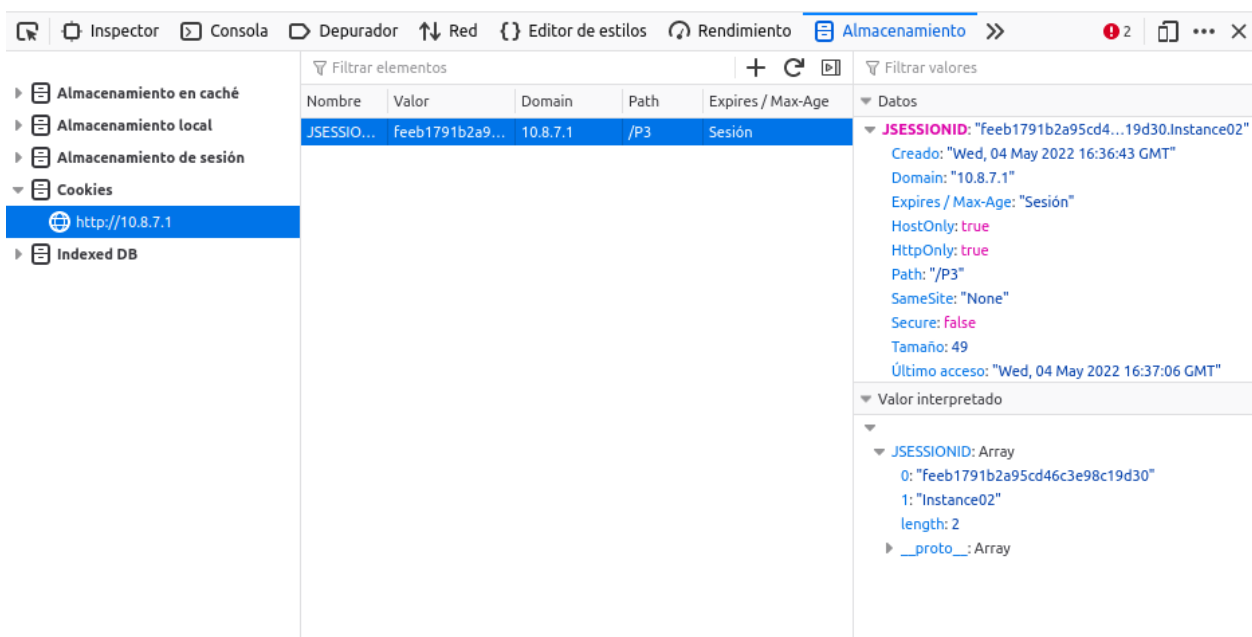
## Pago con tarjeta

Pago realizado con éxito. A continuación se muestra el comprobante del mismo:

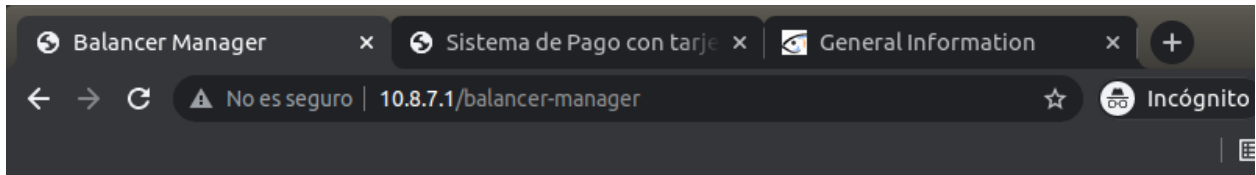
idTransaccion: 86  
idComercio: 1  
importe: 1.0  
codRespuesta: 000  
idAutorizacion: 14

[Volver al comercio](#)

Prácticas de Sistemas Informáticos II



Podemos observar que cada pago va a una de las instancias. Finalmente comprobamos el balance manager para observar el comportamiento de manera más visual.



## Load Balancer Manager for 10.8.7.1

Server Version: Apache/2.2.14 (Ubuntu)

Server Built: Nov 3 2011 03:31:27

### LoadBalancer Status for balancer://si2cluster

StickySession	Timeout	FailoverAttempts	Method
JSESSIONIDjsessionid 0	1		byrequests

Worker URL	Route	RouteRedir	Factor	Set	Status	Elected	To	From
<a href="http://10.8.7.2:28080">http://10.8.7.2:28080</a>	Instance01	1	0	Ok	35	24K	34K	
<a href="http://10.8.7.3:28080">http://10.8.7.3:28080</a>	Instance02	1	0	Ok	28	19K	28K	

Apache/2.2.14 (Ubuntu) Server at 10.8.7.1 Port 80

En el balance manager se observa cómo de nuevo las peticiones llegan a ambas instancias y se ejecutan de manera balanceada. Esto significa que en ningún momento se intenta compensar el número de peticiones en ambas instancias (ésto supondría que una de las instancias no estaría durante el tiempo que dura la compensación).

### Ejercicio número 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.

En primer lugar comenzamos un pago y comprobamos la instancia en la que se va a ejecutar:

Balancer Manager x Sistema de Pago con tarjeta x General Information x +

No es seguro | 10.8.7.1/P3/comienzapago

## Pago con tarjeta

Numero de visa:   
 Titular:   
 Fecha Emisión:   
 Fecha Caducidad:   
 CVV2:

Id Transacción: 69  
 Id Comercion: 1  
 Importe: 1.0

Prácticas de Sistemas Informáticos II

DevTools is now available in Spanish | Always match Chrome's language | Switch DevTools to Spanish | Don't show again

Application | Filter | Only show cookies with an issue

Name	Value	Domain	Path	Expires	Size	HttpOnly	Secure	SameSite	Priority
treeForm_tree-hi	treeForm:tree-clusterTreeNode:SI2Cluster	10.8.7.1	/	Session	56				Medium
JSESSIONID	f67320889e62e05809533198a5b8 Instance01	10.8.7.1	/P3	Session	49	✓			Medium

Cookie Value: ☒ Show URL decoded  
 f67320889e62e05809533198a5b8 Instance01

Como se puede observar se va a realizar en la instancia 1. A continuación paramos dicha instancia.

Balancer Manager x Sistema de Pago con tarjeta x Nodes x +

No es seguro | https://10.8.7.1:4848/common/index.jsf

Home | About | User: admin Role: domain1 Server: 10.8.7.1 | Logout | Help

GlassFish Server Open Source Edition

Total # of available updates: 1

Tree

- Common Tasks
- Domain
- server (Admin Server)
- Clusters
- SI2Cluster
- Standalone Instances
- Nodes
  - Node01
  - Node02
  - localhost-domain1

## Nodes

A node represents a host on which the GlassFish Server software is installed. A node must exist for every host on which GlassFish Server instances reside.

Nodes (3)

Select	Name	Node Host	Type	Instances	Action
<input type="checkbox"/>	Node01	10.8.7.2	SSH	Instance01 Stopped	Ping
<input type="checkbox"/>	Node02	10.8.7.3	SSH	Instance02 Running	Ping
<input type="checkbox"/>	localhost-domain1	localhost	CONFIG		

Procedemos a ejecutar el pago:

No es seguro | 10.8.7.1/P3/comienzapago

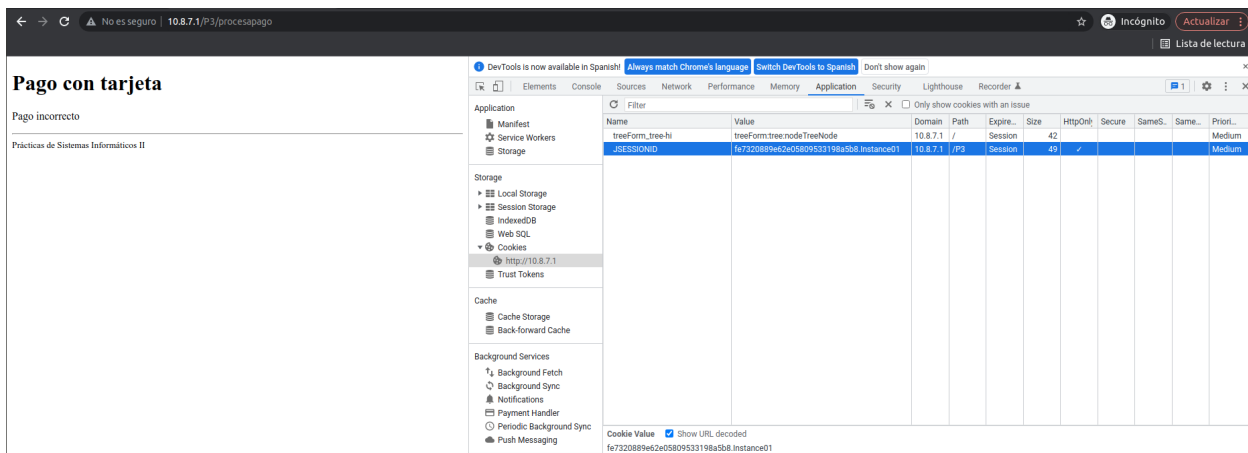
## Pago con tarjeta

Numero de visa:   
 Titular:   
 Fecha Emisión:   
 Fecha Caducidad:   
 CVV2:

Id Transacción: 69  
 Id Comercion: 1  
 Importe: 1.0

Prácticas de Sistemas Informáticos II

Como era de esperar, como la instancia ha sido parada el pago no se puede ejecutar correctamente:



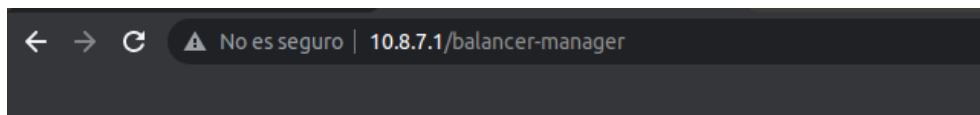
## Ejercicio número 9:

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.

Etiqueta	# Muestras	Media	Mediana	90% Line	95% Line	99% Line	Mín	Máx	% Error	Rendimie...	Kb/sec	Sent KB/sec
P3	1000	3	3	6	8	12	2	22	0,00%	243,0/sec	524,33	0,00
Total	1000	3	3	6	8	12	2	22	0,00%	243,0/sec	524,33	0,00

Hemos añadido el archivo P3.jmx a la entrega de la práctica para que se vean los parámetros de las pruebas utilizados para la ejecución.



## Load Balancer Manager for 10.8.7.1

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://10.8.7.2:28080">http://10.8.7.2:28080</a>	Instance01		1	0	Ok	500	376K	907K
<a href="http://10.8.7.3:28080">http://10.8.7.3:28080</a>	Instance02		1	0	Ok	500	376K	907K

Apache/2.2.14 (Ubuntu) Server at 10.8.7.1 Port 80

¿Qué algoritmo de reparto parece haber seguido el balanceador?

Tras haber realizado un número tan grande de pruebas y comprobar que el reparto de la carga de trabajo en ambas instancias es la misma, podemos confirmar la teoría que arrastramos de ejercicios

anteriores y que el algoritmo de reparto que sigue el balanceador sea con alta probabilidad round-robin.