

UNIVERSIDAD AUTÓNOMA



PRÁCTICAS DE SISTEMAS INFORMÁTICOS II

PRÁCTICA 1B

---

# Memoria

---

*Autores:*

Adrián FERNÁNDEZ

Santiago GONZÁLEZ-CARVAJAL

Pareja 1  
Grupo 2401

12 de marzo de 2019

# Índice

1. Cuestión 1	2
2. Ejercicio 1	2
3. Ejercicio 2	5
4. Cuestión 2	8
5. Ejercicio 3	11
6. Ejercicio 4	13
7. Ejercicio 5	19
8. Ejercicio 6	22
9. Ejercicio 7	23
10.Ejercicio 8	29
11.Ejercicio 9	33
12.Ejercicio 10	34
13.Ejercicio 11	35
14.Ejercicio 12	38
15.Ejercicio 13	40
16.Ejercicio 14	43

## 1. Cuestión 1

**Cuestión 1.** Abrir el archivo VisaDAOLocal.java y comprobar la definición de dicha interfaz. Anote en la memoria comentarios sobre las librerías Java EE importadas y las anotaciones utilizadas. ¿Para qué se utilizan?

Las librerías JEE importadas son javax.ejb.Local para poder utilizar la etiqueta @Local en una interfaz. La cual sirve para declarar una interfaz local de negocios de un “Bean” y, al ser aplicada a una interfaz, designar la misma como una interfaz local de negocios (local business interface).

## 2. Ejercicio 1

**Ejercicio 1.** Introduzca las siguientes modificaciones en el *bean* VisaDAOBean para convertirlo en un EJB de sesión *stateless* con interfaz local:

- Hacer que la clase implemente la interfaz local y convertirla en un EJB *stateless* mediante la anotación *Stateless*.

```
...
import javax.ejb.Stateless;
...
@Stateless(mappedName="VisaDAOBean")
public class VisaDAOBean extends DBTester implements VisaDAOLocal {
...
}
```

- Eliminar el constructor por defecto de la clase.
- Ajustar los métodos getPagos() a la interfaz definida en VisaDAOLocal.
- Incluye en la memoria cada fragmento de código donde se han ido añadiendo las modificaciones solicitadas.

## Código modificado:

En el fichero **VisaDAOBean.java** (además de eliminar el constructor):

```
1  /* A partir de la linea 13 */
2  package ssii2.visa.dao;
3  import ssii2.visa.*;
4
5  import javax.jws.WebMethod;
6  import javax.jws.WebParam;
7  import javax.jws.WebService;
8  import java.sql.Connection;
9  import java.sql.PreparedStatement;
10 import java.sql.ResultSet;
11 import java.sql.SQLException;
12 import java.sql.Statement;
13 import java.util.ArrayList;
14 import java.util.Arrays;
15
16 import javax.ejb.Stateless;
17
18 /**
19  * @author jaime
20  */
21 @Stateless(mappedName="VisaDAOBean")
22 public class VisaDAOBean extends DBTester implements VisaDAOLocal {
23     .
24     .
25     .
26
27     /* A partir de la linea 333 */
28     public PagoBean[] getPagos(String idComercio) {
29
30         PreparedStatement pstmt = null;
31         Connection pcon = null;
32         ResultSet rs = null;
33         PagoBean[] ret = null;
34         ArrayList<PagoBean> pagos = null;
35         String qry = null;
36
37         try {
38
39             // Crear una conexion u obtenerla del pool
40             pcon = getConnection();
41             qry = "SELECT PAGOS.QRY";
42             errorLog(qry + "[idComercio=" + idComercio + "]");
43
44             // La preparacion del statement
45             // es automaticamente tomada de un pool en caso
46             // de que ya haya sido preparada con anterioridad
47             pstmt = pcon.prepareStatement(qry);
48
49             pstmt.setString(1, idComercio);
50             rs = pstmt.executeQuery();
51
52             pagos = new ArrayList<PagoBean>();
53
54             while (rs.next()) {
55                 TarjetaBean t = new TarjetaBean();
```

```

56         PagoBean p = new PagoBean();
57         p.setIdTransaccion(rs.getString("idTransaccion"));
58         p.setIdComercio(rs.getString("idComercio"));
59         p.setImporte(rs.getFloat("importe"));
60         t.setNumero(rs.getString("numeroTarjeta"));
61         p.setTarjeta(t);
62         p.setCodRespuesta(rs.getString("codRespuesta"));
63         p.setIdAutorizacion(String.valueOf(rs.getInt("
            idAutorizacion"))));
64
65         pagos.add(p);
66     }
67
68     ret = new PagoBean[pagos.size()];
69     ret = pagos.toArray(ret);
70
71     // Cerramos / devolvemos la conexion al pool
72     pcon.close();
73
74     } catch (Exception e) {
75         errorLog(e.toString());
76
77     } finally {
78         try {
79             if (rs != null) {
80                 rs.close(); rs = null;
81             }
82             if (pstmt != null) {
83                 pstmt.close(); pstmt = null;
84             }
85             if (pcon != null) {
86                 closeConnection(pcon); pcon = null;
87             }
88         } catch (SQLException e) {
89             }
90     }
91
92     return ret;
93 }
94 .
95 .
96 .

```

### 3. Ejercicio 2

**Ejercicio 2.** Modificar el *servlet* *ProcesaPago* para que acceda al EJB local. Para ello, modificar el archivo *ProcesaPago.java* de la siguiente manera:

En la sección de preproceso, añadir las siguientes importaciones de clases que se van a utilizar:

```
...
import javax.ejb.EJB;
import ssii2.visa.VisaDAOLocal;
...
```

Se deberán eliminar estas otras importaciones que dejan de existir en el proyecto, como por ejemplo:

```
import ssii2.visa.VisaDAOWSService; // Stub generado automaticamente
import ssii2.visa.VisaDAOWS; // Stub generado automaticamente
```

Añadir como atributo de la clase el objeto proxy que permite acceder al EJB local, con su correspondiente anotación que lo declara como tal:

```
...
@EJB(name="VisaDAOBean", beanInterface=VisaDAOLocal.class)
private VisaDAOLocal dao;
...
```

En el cuerpo del servlet, eliminar la declaración de la instancia del antiguo webservice *VisaDAOWS*, así como el código necesario para obtener la referencia remota:

```
...
VisaDAOWS dao = null;
VisaDAOWSService service = new VisaDAOWSService();
dao = service.getVisaDAOWSPort()
```

Incluye en la memoria cada fragmento de código donde se han ido añadiendo las modificaciones solicitadas.

Eliminar también las referencias a *BindingProvider*.

## Código modificado:

En el fichero **ProcesaPago.java** (además de eliminar los imports generados por los stubs). También, en el resto de servlets del proyecto que hacen uso de VisaDAOWS, hemos realizado los mismos cambios. Además, en el servlet **GetPagos.java**, también hemos tenido que tener en cuenta el nuevo retorno del método `getPagos()`, que es un `PagoBean[]` en vez de un `ArrayList<PagoBean>` (como antes).

```
1  \* A partir de la linea 33 *\n2  package ssii2.controlador;\n3\n4  import java.io.IOException;\n5  import java.net.InetAddress;\n6  import java.net.NetworkInterface;\n7  import java.net.SocketException;\n8  import java.net.UnknownHostException;\n9  import java.util.Collections;\n10 import java.util.Enumeration;\n11 import javax.servlet.ServletException;\n12 import javax.servlet.http.HttpServletRequest;\n13 import javax.servlet.http.HttpServletResponse;\n14 import javax.servlet.http.HttpSession;\n15 import javax.xml.ws.WebServiceRef;\n16 import javax.xml.ws.BindingProvider;\n17 import ssii2.visa.*;\n18\n19 import javax.ejb.EJB;\n20 import ssii2.visa.VisaDAOLocal;\n21 .\n22 .\n23 .\n24\n25 \* A partir de la linea 58 *\n26 @EJB(name="VisaDAOBean", beanInterface=VisaDAOLocal.class)\n27 private VisaDAOLocal dao;\n28 .\n29 .\n30 .\n31\n32 \* A partir de la linea 140 *\n33 @Override\n34 protected void processRequest(HttpServletRequest request ,\n35                               HttpServletResponse response)\n36     throws ServletException , IOException {\n37\n38     TarjetaBean tarjeta = creaTarjeta(request);\n39     ValidadorTarjeta val = new ValidadorTarjeta();\n40     PagoBean pago = null;\n41\n42     // printAddresses(request, response);\n43     if (! val.esValida(tarjeta)) {\n44         request.setAttribute(val.getErrorName(), val.getErrorVisa());\n45         reenvia("/formdatosvisa.jsp", request, response);\n46         return;\n47     }\n48\n49     //String url_from_xml;
```

```

49 //VisaDAOWSService service = new VisaDAOWSService();
50 //VisaDAOWS dao = service.getVisaDAOWSPort();
51
52
53 //url_from_xml=getServletContext().getInitParameter("webmaster");
54
55 //BindingProvider bp = (BindingProvider) dao;
56 //bp.getRequestContext().put(BindingProvider.
    ENDPOINT_ADDRESS_PROPERTY, url_from_xml);
57 //VisaDAO dao = new VisaDAO();
58 try{
59     HttpSession session = request.getSession(false);
60     if (session != null) {
61         pago = (PagoBean) session.getAttribute(
            ComienzaPago.ATTRPAGO);
62     }
63     if (pago == null) {
64         pago = creaPago(request);
65         boolean isdebug = Boolean.valueOf(request.
            getParameter("debug"));
66         dao.setDebug(isdebug);
67         boolean isdirectConnection = Boolean.valueOf(
            request.getParameter("directConnection"))
            ;
68         dao.setDirectConnection(isdirectConnection);
69         boolean usePrepared = Boolean.valueOf(request
            .getParameter("usePrepared"));
70         dao.setPrepared(usePrepared);
71     }
72
73     // Almacenamos la tarjeta en el pago
74     pago.setTarjeta(tarjeta);
75
76     if (!dao.compruebaTarjeta(tarjeta)) {
77         enviaError(new Exception("Tarjeta no autorizada:"),
            request, response);
78         return;
79     }
80     pago = dao.realizaPago(pago);
81     if (pago == null) {
82         enviaError(new Exception("Pago incorrecto"), request,
            response);
83         return;
84     }
85
86     request.setAttribute(ComienzaPago.ATTRPAGO, pago);
87     if (session != null) session.invalidate();
88     reenvia("/pagoexito.jsp", request, response);
89     return;
90 }catch(Exception e){
91     enviaError(new Exception("Pago incorrecto."), request, response
        );
92 }
93 }
94 .
95 .
96 .

```



## 4. Cuestión 2

**Cuestión 2.** Abrir el archivo *application.xml* y explicar su contenido. Verifique el contenido de todos los archivos .jar / .war / .ear que se han construido hasta el momento (empleando el comando jar-tvf). Anote sus comentarios y evidencias en la memoria.

### Código de application.xml:

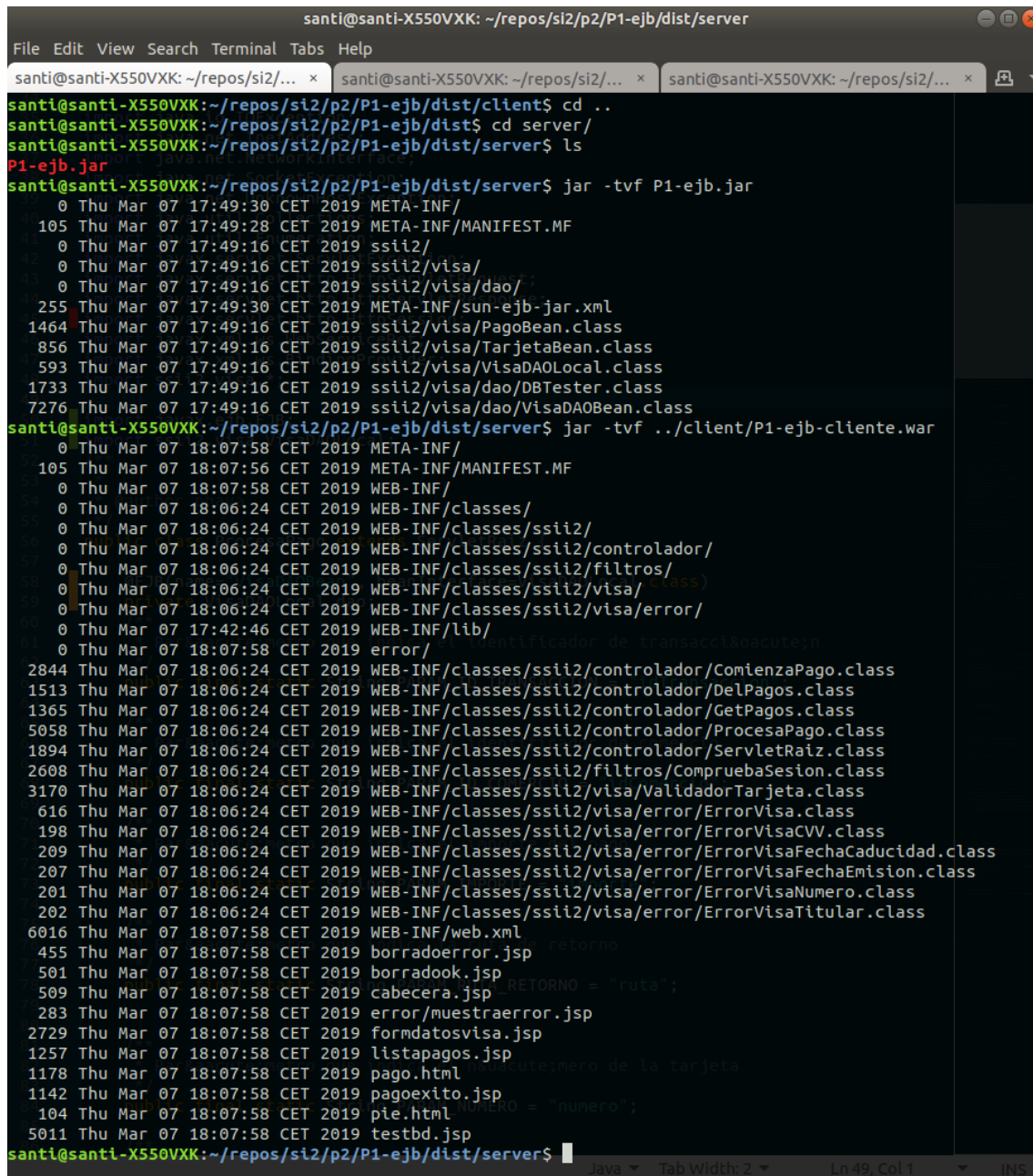
```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <application version="5" xmlns="http://java.sun.com/xml/ns/javaee"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.
   sun.com/xml/ns/javaee/application_5.xsd">
3   <display-name>P1-ejb</display-name>
4   <module>
5     <ejb>P1-ejb.jar</ejb>
6   </module>
7   <module>
8     <web>
9       <web-uri>P1-ejb-cliente.war</web-uri>
10      <context-root>/P1-ejb-cliente</context-root>
11    </web>
12  </module>
13 </application>
```

Este fichero define la aplicación para que la máquina donde se va a desplegar la misma pueda identificar sus módulos. Podemos ver el nombre de la aplicación, el .jar del servidor y el .war del cliente con su nombre.

## Evidencias:

El archivo **P1-ejb.jar** (comprimido del servidor) contiene las especificaciones para desplegar la base de datos, junto con **PagoBean** y **TarjetaBean** para implementar el acceso a la BD.

El archivo **P1-ejb-cliente.war** (comprimido del cliente) contiene las especificaciones para desplegar los “servlets”, junto con los filtros y el validador de tarjetas.



```
santi@santi-X550VXK: ~/repos/si2/p2/P1-ejb/dist/server
File Edit View Search Terminal Tabs Help
santi@santi-X550VXK: ~/repos/si2/... x santi@santi-X550VXK: ~/repos/si2/... x santi@santi-X550VXK: ~/repos/si2/... x
santi@santi-X550VXK:~/repos/si2/p2/P1-ejb/dist/client$ cd ..
santi@santi-X550VXK:~/repos/si2/p2/P1-ejb/dist$ cd server/
santi@santi-X550VXK:~/repos/si2/p2/P1-ejb/dist/server$ ls
P1-ejb.jar
santi@santi-X550VXK:~/repos/si2/p2/P1-ejb/dist/server$ jar -tvf P1-ejb.jar
0 Thu Mar 07 17:49:30 CET 2019 META-INF/
105 Thu Mar 07 17:49:28 CET 2019 META-INF/MANIFEST.MF
0 Thu Mar 07 17:49:16 CET 2019 ssi2/
0 Thu Mar 07 17:49:16 CET 2019 ssi2/visa/
0 Thu Mar 07 17:49:16 CET 2019 ssi2/visa/dao/
255 Thu Mar 07 17:49:30 CET 2019 META-INF/sun-ejb-jar.xml
1464 Thu Mar 07 17:49:16 CET 2019 ssi2/visa/PagoBean.class
856 Thu Mar 07 17:49:16 CET 2019 ssi2/visa/TarjetaBean.class
593 Thu Mar 07 17:49:16 CET 2019 ssi2/visa/VisaDAOLocal.class
1733 Thu Mar 07 17:49:16 CET 2019 ssi2/visa/dao/DBTester.class
7276 Thu Mar 07 17:49:16 CET 2019 ssi2/visa/dao/VisaDAOBean.class
santi@santi-X550VXK:~/repos/si2/p2/P1-ejb/dist/server$ jar -tvf ../client/P1-ejb-cliente.war
0 Thu Mar 07 18:07:58 CET 2019 META-INF/
105 Thu Mar 07 18:07:56 CET 2019 META-INF/MANIFEST.MF
0 Thu Mar 07 18:07:58 CET 2019 WEB-INF/
0 Thu Mar 07 18:06:24 CET 2019 WEB-INF/classes/
0 Thu Mar 07 18:06:24 CET 2019 WEB-INF/classes/ssi2/
0 Thu Mar 07 18:06:24 CET 2019 WEB-INF/classes/ssi2/controlador/
0 Thu Mar 07 18:06:24 CET 2019 WEB-INF/classes/ssi2/filtros/
0 Thu Mar 07 18:06:24 CET 2019 WEB-INF/classes/ssi2/visa/
0 Thu Mar 07 18:06:24 CET 2019 WEB-INF/classes/ssi2/visa/error/
0 Thu Mar 07 17:42:46 CET 2019 WEB-INF/lib/
0 Thu Mar 07 18:07:58 CET 2019 error/
2844 Thu Mar 07 18:06:24 CET 2019 WEB-INF/classes/ssi2/controlador/ComienzaPago.class
1513 Thu Mar 07 18:06:24 CET 2019 WEB-INF/classes/ssi2/controlador/DelPagos.class
1365 Thu Mar 07 18:06:24 CET 2019 WEB-INF/classes/ssi2/controlador/GetPagos.class
5058 Thu Mar 07 18:06:24 CET 2019 WEB-INF/classes/ssi2/controlador/ProcesaPago.class
1894 Thu Mar 07 18:06:24 CET 2019 WEB-INF/classes/ssi2/controlador/ServletRaiz.class
2608 Thu Mar 07 18:06:24 CET 2019 WEB-INF/classes/ssi2/filtros/CompruebaSesion.class
3170 Thu Mar 07 18:06:24 CET 2019 WEB-INF/classes/ssi2/visa/ValidadorTarjeta.class
616 Thu Mar 07 18:06:24 CET 2019 WEB-INF/classes/ssi2/visa/error/ErrorVisa.class
198 Thu Mar 07 18:06:24 CET 2019 WEB-INF/classes/ssi2/visa/error/ErrorVisaCVV.class
209 Thu Mar 07 18:06:24 CET 2019 WEB-INF/classes/ssi2/visa/error/ErrorVisaFechaCaducidad.class
207 Thu Mar 07 18:06:24 CET 2019 WEB-INF/classes/ssi2/visa/error/ErrorVisaFechaEmision.class
201 Thu Mar 07 18:06:24 CET 2019 WEB-INF/classes/ssi2/visa/error/ErrorVisaNumero.class
202 Thu Mar 07 18:06:24 CET 2019 WEB-INF/classes/ssi2/visa/error/ErrorVisaTitular.class
6016 Thu Mar 07 18:07:58 CET 2019 WEB-INF/web.xml
455 Thu Mar 07 18:07:58 CET 2019 borradoerror.jsp
501 Thu Mar 07 18:07:58 CET 2019 borradook.jsp
509 Thu Mar 07 18:07:58 CET 2019 cabecera.jsp
283 Thu Mar 07 18:07:58 CET 2019 error/muestraerror.jsp
2729 Thu Mar 07 18:07:58 CET 2019 formdatosvisa.jsp
1257 Thu Mar 07 18:07:58 CET 2019 listapagos.jsp
1178 Thu Mar 07 18:07:58 CET 2019 pago.html
1142 Thu Mar 07 18:07:58 CET 2019 pagoexito.jsp
104 Thu Mar 07 18:07:58 CET 2019 pie.html
5011 Thu Mar 07 18:07:58 CET 2019 testbd.jsp
santi@santi-X550VXK:~/repos/si2/p2/P1-ejb/dist/server$
```

Figura 1: Contenido del .jar del servidor (arriba) y del .war del cliente(abajo).

El archivo **P1-ejb.ear** (comprimido de la aplicación) contiene las especificaciones para desplegar la aplicación completa en dos máquinas (cliente y servidor).

```
santi@santi-X550VXK:~/repos/si2/p2/P1-ejb/dist$ jar -tvf P1-ejb.ear
 0 Thu Mar 07 18:26:22 CET 2019 META-INF/
105 Thu Mar 07 18:26:20 CET 2019 META-INF/MANIFEST.MF
508 Sat Feb 11 23:33:00 CET 2012 META-INF/application.xml
20991 Thu Mar 07 18:07:58 CET 2019 P1-ejb-cliente.war
7174 Thu Mar 07 17:49:30 CET 2019 P1-ejb.jar
santi@santi-X550VXK:~/repos/si2/p2/P1-ejb/dist$
```

Figura 2: Contenido del .ear de la aplicación.

## 5. Ejercicio 3

**Ejercicio 3.** Preparar los PCs con el esquema descrito y realizar el despliegue de la aplicación:

- Editar el archivo *build.properties* para que las propiedades *as.host.client* y *as.host.server* contengan la dirección IP del servidor de aplicaciones. Indica qué valores y porqué son esos valores.
- Editar el archivo *postgresql.properties* para la propiedad *db.client.host* y *db.host* contengan las direcciones IP adecuadas para que el servidor de aplicaciones se conecte al postgresql, ambos estando en servidores diferentes. Indica qué valores y porqué son esos valores.

Desplegar la aplicación de empresa  
ant desplegar

### Código modificado:

En el fichero **build.properties**:

```
1 # Propiedades de despliegue de aplicacion de Visa
2 nombre=P1-ejb
3 build=${basedir}/build
4 build.client=${build}/client
5 build.server=${build}/server
6 dist=${basedir}/dist
7 dist.client=${dist}/client
8 dist.server=${dist}/server
9 src=${basedir}/src
10 src.client=${src}/client
11 src.server=${src}/server
12 web=${basedir}/web
13 conf=${basedir}/conf
14 conf.server=${conf}/server
15 conf.application=${conf}/application
16 paquete=ssii2
17 war=${nombre}-cliente.war
18 jar=${nombre}.jar
19 ear=${nombre}.ear
20 asadmin=${as.home}/bin/asadmin
21 as.home=${env.J2EE_HOME}
22 as.lib=${as.home}/lib
23 as.user=admin
24 as.host.client=10.1.1.2
25 as.host.server=10.1.1.2
26 as.port=4848
27 as.passwordfile=${basedir}/passwordfile
28 as.target=server
```

Hemos puesto a 10.1.1.2 las variables *as.host.server* y *as.host.client* para alojar el cliente y el servidor de la aplicación en la máquina virtual 2.

En el fichero **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=10.1.1.1
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=10.1.1.2
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
```

Hemos puesto db.host a 10.1.1.1 para que la base de datos se cree en la máquina virtual 1, mientras que hemos indicado que la aplicación que utilizará la base de datos se encuentra en la máquina virtual 2 poniendo 10.1.1.2 como valor de db.client.host.

## 6. Ejercicio 4

**Ejercicio 4.** Comprobar el correcto funcionamiento de la aplicación mediante llamadas directas a través de las páginas *pago.html* y *testbd.jsp* (sin *directconnection*). Realice un pago. Lístelo. Elimínelo. Téngase en cuenta que la aplicación se habrá desplegado bajo la ruta */P1-ejb-cliente*.

Incluya en la memoria de prácticas todos los pasos necesarios para resolver este ejercicio así como las evidencias obtenidas. Se pueden incluir por ejemplo capturas de pantalla.

### Pasos realizados:

Los pasos realizados para resolver el ejercicio han sido, como hemos visto anteriormente: la compilación y empaquetado del servidor, la compilación y empaquetado del cliente, el empaquetado de la aplicación, y, finalmente, el despliegue de la misma. A partir de aquí, simplemente hemos realizado, listado y borrado los pagos como veremos en la figuras que aparecen a continuación.

Consola de administración:

Figura 3: Consola de administración de Glassfish donde aparece la aplicación desplegada en local.

En pago.html:

Sistema de Pago con tarjeta X +

← → ↻ 🏠 ⓘ 10.1.1.2:8080/P1-ejb-cliente/comienzapago

## Pago con tarjeta

Numero de visa:

Titular:

Fecha Emisión:

Fecha Caducidad:

CVV2:

---

Id Transacción: 43  
Id Comercion: 22  
Importe: 222.0

---

Prácticas de Sistemas Informáticos II

Figura 4: Formulario de pago.

Sistema de Pago con tarjeta X +

← → ↻ 🏠 ⓘ 10.1.1.2:8080/P1-ejb-cliente/procesapago

## Pago con tarjeta

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

idTransaccion:  
idComercio: 22  
importe: 222.0  
codRespuesta: 000  
idAutorizacion: 4

[Volver al comercio](#)

---

Prácticas de Sistemas Informáticos II

Figura 5: Pago correcto.





Figura 6: Listado de pagos tras realizar el pago.



Figura 7: Resultado obtenido al borrar el pago.



Figura 8: Listado de pagos tras borrar el pago.

En testbd.html:

Sistema de Pago con tarjeta X +

← → ↻ 🏠 ⓘ 10.1.1.2:8080/P1-ejb-cliente/testbd.jsp

## Pago con tarjeta

### Proceso de un pago

Id Transacción:

Id Comercio:

Importe:

Numero de visa:

Titular:

Fecha Emisión:

Fecha Caducidad:

CVV2:

Modo debug: ☐ True ☐ False

Direct Connection: ☐ True ☐ False

Use Prepared: ☐ True ☐ False

Figura 9: Formulario de pago.

Sistema de Pago con tarjeta X +

← → ↻ 🏠 ⓘ 10.1.1.2:8080/P1-ejb-cliente/procesapago

## Pago con tarjeta

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

idTransaccion:

idComercio: 43

importe: 125.0

codRespuesta: 000

idAutorizacion: 5

[Volver al comercio](#)

---

Prácticas de Sistemas Informáticos II

Figura 10: Pago correcto.

Sistema de Pago con tarjeta × +

← → ↺ 🏠

10.1.1.2:8080/P1-ejb-cliente/getpagos

# Pago con tarjeta

Lista de pagos del comercio 43

idTransaccion	Importe	codRespuesta	idAutorizacion
65	125.0	000	5

[Volver al comercio](#)

---

Prácticas de Sistemas Informáticos II

Figura 11: Listado de pagos tras realizar el pago.

## 7. Ejercicio 5

**Ejercicio 5.** Realizar los cambios indicados en P1-ejb-servidor-remoto y preparar los PCs con el esquema de máquinas virtuales indicado. Compilar, empaquetar y desplegar de nuevo la aplicación P1-ejb como servidor de EJB remotos de forma similar a la realizada en el Ejercicio 3 con la Figura 2 como entorno de despliegue. Esta aplicación tendrá que desplegarse en la máquina virtual del PC2.

Se recomienda replegar la aplicación anterior (EJB local) antes de desplegar ésta.

Incluye en la memoria cada fragmento de código donde se han ido añadiendo las modificaciones solicitadas así como detallando los pasos realizados.

### Pasos realizados:

Los pasos realizados han sido los indicados con las modificaciones de código que aparecen a continuación.

### Código modificado:

En el fichero **VisaDAORemote.java**:

```
1  /* A partir de la linea 13 */
2  package ssii2.visa;
3
4  import java.sql.Connection;
5  import java.sql.PreparedStatement;
6  import java.sql.ResultSet;
7  import java.sql.SQLException;
8  import java.sql.Statement;
9  import java.util.ArrayList;
10 import javax.ejb.Remote;
11
12 @Remote
13 public interface VisaDAORemote {
14     public boolean compruebaTarjeta(TarjetaBean tarjeta);
15     public PagoBean realizaPago(PagoBean pago);
16     public PagoBean[] getPagos(String idComercio);
17     public int delPagos(String idComercio);
18     public boolean isDebug();
19     public boolean isPrepared();
20     public void setPrepared(boolean prepared);
21     public void setDebug(boolean debug);
22     public int getDirectConnectionCount();
23     public int getDSNConnectionCount();
24     public boolean isDirectConnection();
25     public void setDirectConnection(boolean directConnection);
26 }
```

En el fichero **VisaDAOBean.java**:

```
1  /* A partir de la linea 13 */
2  package ssii2.visa.dao;
```

```

3 import ssii2.visa.*;
4
5 import javax.jws.WebMethod;
6 import javax.jws.WebParam;
7 import javax.jws.WebService;
8 import java.sql.Connection;
9 import java.sql.PreparedStatement;
10 import java.sql.ResultSet;
11 import java.sql.SQLException;
12 import java.sql.Statement;
13 import java.util.ArrayList;
14 import java.util.Arrays;
15
16 import javax.ejb.Stateless;
17
18 /**
19  * @author jaime
20  */
21 @Stateless(mappedName="VisaDAOBean")
22 public class VisaDAOBean extends DBTester implements VisaDAOLocal,
23     VisaDAORemote {
24
25     .
26     .
27     .

```

En el fichero **PagoBean.java**:

```

1 /* A partir de la linea 8 */
2 package ssii2.visa;
3
4 import java.io.Serializable;
5
6 /**
7  *
8  * @author jaime
9  */
10 public class PagoBean implements Serializable {
11     .
12     .
13     .

```

En el fichero **TarjetaBean.java**:

```

1 /* A partir de la linea 7 */
2 package ssii2.visa;
3
4 import java.io.Serializable;
5
6 public class TarjetaBean implements Serializable {
7     .
8     .
9     .

```

Evidencias:

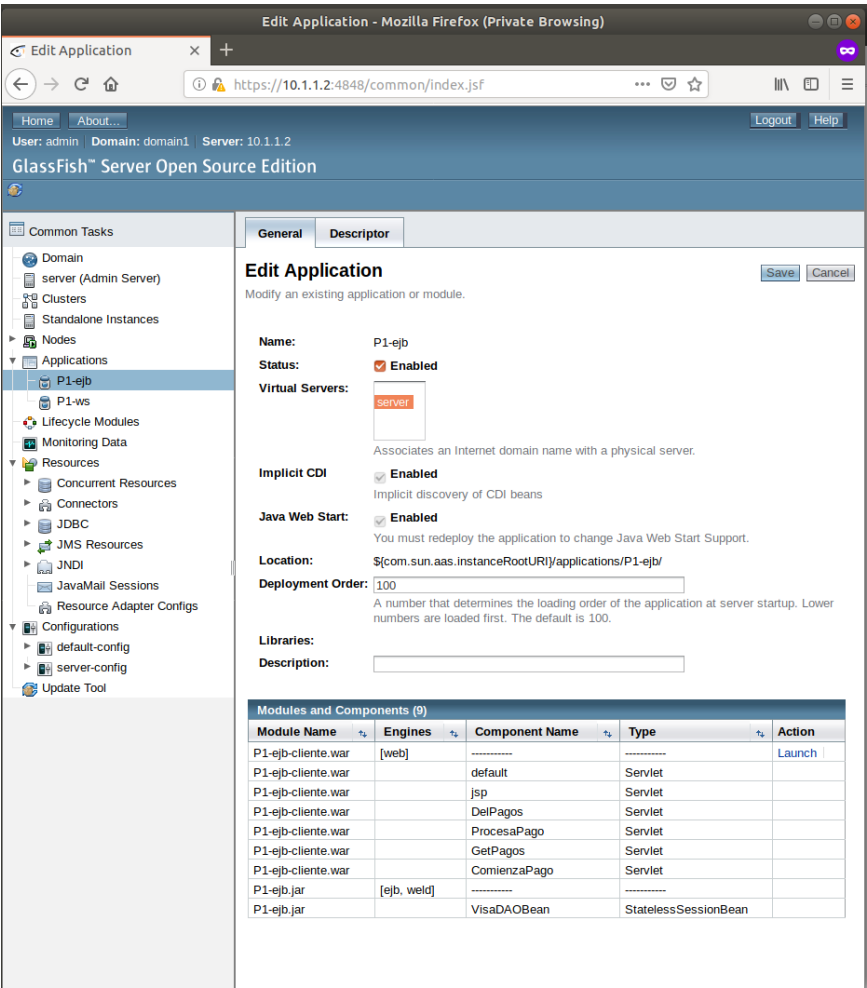


Figura 12: Aplicación desplegada en remoto.

## 8. Ejercicio 6

**Ejercicio 6.** Realizar los cambios comentados en la aplicación P1-base para convertirla en P1-ejb-clienteremoto. Compilar, empaquetar y desplegar de nuevo la aplicación en otra máquina virtual distinta a la de la aplicación servidor, es decir, esta aplicación cliente estará desplegada en la MV del PC1 tal y como se muestra en el diagrama de despliegue de la Figura 2. Conectarse a la aplicación cliente y probar a realizar un pago. Comprobar los resultados e incluir en la memoria evidencias de que el pago ha sido realizado de forma correcta.

### Evidencias:



Sistema de Pago con tarjeta X +

← → ↻ 🏠 ⓘ 10.1.1.1:8080/P1-ejb-cliente-remoto/comienzapago

### Pago con tarjeta

Numero de visa:

Titular:

Fecha Emisión:

Fecha Caducidad:

CVV2:

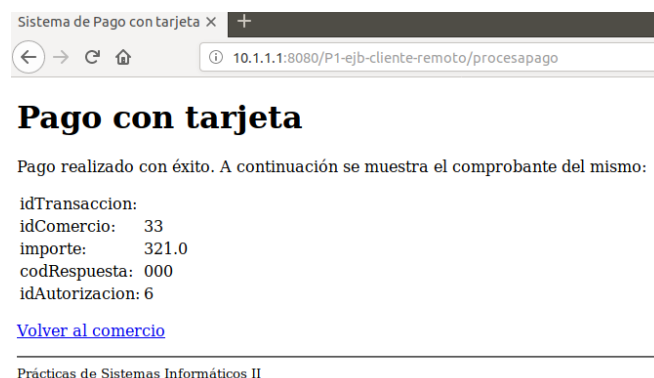
---

Id Transacción: 56  
Id Comercio: 33  
Importe: 321.0

---

Prácticas de Sistemas Informáticos II

Figura 13: Formulario de pago.



Sistema de Pago con tarjeta X +

← → ↻ 🏠 ⓘ 10.1.1.1:8080/P1-ejb-cliente-remoto/procesapago

### Pago con tarjeta

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

idTransaccion:  
idComercio: 33  
importe: 321.0  
codRespuesta: 000  
idAutorizacion: 6

[Volver al comercio](#)

---

Prácticas de Sistemas Informáticos II

Figura 14: Pago realizado correctamente.

6	56	000	321	33	4735 7643 8617 9641	07/03/19 13:22
---	----	-----	-----	----	---------------------	----------------

Figura 15: Comprobación de que el pago se ha guardado correctamente en la BD.

## 9. Ejercicio 7

**Ejercicio 7.** Modificar la aplicación VISA para soportar el campo saldo:

**Archivo TarjetaBean.java:**

- Añadir el atributo saldo y sus métodos de acceso:

```
private double saldo;
```

**Archivo VisaDAOBean.java:**

- Importar la definición de la excepción EJBException que debe lanzar el servlet para indicar que se debe realizar un rollback:

```
import javax.ejb.EJBException;
```

- Declarar un prepared statement para recuperar el saldo de una tarjeta de la base de datos.
- Declarar un prepared statement para insertar el nuevo saldo calculado en la base de datos.
- Modificar el método realizaPago con las siguientes acciones:
  - Recuperar el saldo de la tarjeta a través del prepared statement declarado anteriormente.
  - Comprobar si el saldo es mayor o igual que el importe de la operación. Si no lo es, retornar denegando el pago (idAutorizacion= null y pago retornado=null).
  - Si el saldo es suficiente, decrementarlo en el valor del importe del pago y actualizar el registro de la tarjeta para reflejar el nuevo saldo mediante el prepared statement declarado anteriormente.
  - Si lo anterior es correcto, ejecutar el proceso de inserción del pago y obtención del idAutorizacion, tal como se realizaba en la práctica anterior (este código ya debe estar programado y no es necesario modificarlo).
  - En caso de producirse cualquier error a lo largo del proceso (por ejemplo, si no se obtiene el idAutorizacion porque la transacción está duplicada), lanzar una excepción EJBException para retornar al cliente.
- Modificar el servlet ProcesaPago para que capture la posible interrupción EJBException lanzada por realizaPago, y, en caso de que se haya lanzado, devuelva la página de error mediante el método enviaError (recordar antes de retornar que se debe invalidar la sesión, si es que existe).
- Incluye en la memoria cada fragmento de código donde se han ido añadiendo las modificaciones solicitadas.



## Código modificado:

En el fichero **TarjetaBean.java**:

```
1  /* A partir de la linea 7 */
2  package ssii2.visa;
3
4  public class TarjetaBean {
5
6      private String numero;
7      private String titular;
8      private String fechaEmision;
9      private String fechaCaducidad;
10     private String codigoVerificacion; /* CVV2 */
11     private double saldo;
12
13
14 }
```

En el fichero **VisaDAOBean.java**:

```
1  /* A partir de la linea 13 */
2  package ssii2.visa.dao;
3  import ssii2.visa.*;
4
5  import javax.jws.WebMethod;
6  import javax.jws.WebParam;
7  import javax.jws.WebService;
8  import java.sql.Connection;
9  import java.sql.PreparedStatement;
10 import java.sql.ResultSet;
11 import java.sql.SQLException;
12 import java.sql.Statement;
13 import java.util.ArrayList;
14 import java.util.Arrays;
15
16 import javax.ejb.Stateless;
17
18 import javax.ejb.EJBException;
19
20
21
22
23 /* A partir de la linea 57 */
24     private static final String SELECT_SALDO_QRY =
25         "select saldo from tarjeta " +
26         "where numeroTarjeta = ?";
27     private static final String INSERT_SALDO_QRY =
28         "update tarjeta set saldo=saldo-? " +
29         "where numeroTarjeta = ?";
30
31
32
33
34 /* A partir de la linea 213 */
35     public synchronized PagoBean realizaPago(PagoBean pago) {
36         Connection con = null;
37         Statement stmt = null;
```

```

38 PagoBean returned_pago = null;
39 ResultSet rs = null;
40 boolean ret = false;
41 String codRespuesta = "999"; // En principio, denegado
42 PreparedStatement pstmt = null;
43
44 if (pago.getIdTransaccion() == null) {
45     return null;
46 }
47
48 // Registrar el pago en la base de datos
49 try {
50     // Obtener conexion
51     con = getConnection();
52
53     double saldo_tarjeta = -1.0;
54     // Insertar en la base de datos el pago
55     String saldo_string=SELECT_SALDO_QRY;
56     errorLog(saldo_string);
57     pstmt = con.prepareStatement(saldo_string);
58     pstmt.setString(1, pago.getTarjeta().getNumero());
59     rs = pstmt.executeQuery();
60     if(rs.next()){
61         saldo_tarjeta = rs.getDouble("saldo");
62     }
63
64     double importe_pago = pago.getImporte();
65
66     if(saldo_tarjeta < importe_pago){
67         pago.setIdAutorizacion(null);
68
69         return null;
70     }
71
72     String saldo_qry=INSERT_SALDO_QRY;
73     errorLog(saldo_qry);
74     pstmt = con.prepareStatement(saldo_qry);
75     pstmt.setDouble(1, importe_pago);
76     pstmt.setString(2, pago.getTarjeta().getNumero());
77     ret = false;
78     if (!pstmt.execute()
79         && pstmt.getUpdateCount() == 1) {
80         ret = true;
81     }
82
83     if (isPrepared() == true) {
84         String insert = INSERT_PAGOS_QRY;
85         errorLog(insert);
86         pstmt = con.prepareStatement(insert);
87         pstmt.setString(1, pago.getIdTransaccion());
88         pstmt.setDouble(2, pago.getImporte());
89         pstmt.setString(3, pago.getIdComercio());
90         pstmt.setString(4, pago.getTarjeta().getNumero());
91         ret = false;
92         if (!pstmt.execute()
93             && pstmt.getUpdateCount() == 1) {
94             ret = true;
95         }

```

```

96
97     } else {
98         /*****
99         stmt = con.createStatement();
100         String insert = getQryInsertPago(pago);
101         errorLog(insert);
102         ret = false;
103         if (!stmt.execute(insert)
104             && stmt.getUpdateCount() == 1) {
105             ret = true;
106         }
107     } /*****/
108
109     // Obtener id. autorizacion
110     if (ret) {
111         if (isPrepared() == true) {
112             String select = SELECT_PAGO_TRANSACCION_QRY;
113             errorLog(select);
114             pstmt = con.prepareStatement(select);
115             pstmt.setString(1, pago.getIdTransaccion());
116             pstmt.setString(2, pago.getIdComercio());
117             rs = pstmt.executeQuery();
118         } else {
119             /*****/
120
121             String select = getQryBuscaPagoTransaccion(pago);
122             errorLog(select);
123             rs = stmt.executeQuery(select);
124
125             } /*****/
126         if (rs.next()) {
127             pago.setIdAutorizacion(String.valueOf(rs.getInt("
128                 idAutorizacion")));
129             pago.setCodRespuesta(rs.getString("codRespuesta"));
130         } else {
131             ret = false;
132         }
133     }
134
135 } catch (Exception e) {
136     errorLog(e.toString());
137     ret = false;
138     throw new EJBException("EJBException: "+e.getMessage());
139 } finally {
140     try {
141         if (rs != null) {
142             rs.close(); rs = null;
143         }
144         if (stmt != null) {
145             stmt.close(); stmt = null;
146         }
147         if (pstmt != null) {
148             pstmt.close(); pstmt = null;
149         }
150         if (con != null) {
151             closeConnection(con); con = null;
152         }

```

```

153         } catch (SQLException e) {
154             throw new EJBException("EJBException: "+e.getMessage());
155         }
156     }
157
158     if(ret == false){
159         return null;
160     } else{
161         returned_pago = new PagoBean();
162         returned_pago.setIdTransaccion(returned_pago.getIdTransaccion());
163         returned_pago.setIdComercio(pago.getIdComercio());
164         returned_pago.setImporte(pago.getImporte());
165         returned_pago.setRutaRetorno(pago.getRutaRetorno());
166         returned_pago.setTarjeta(pago.getTarjeta());
167         returned_pago.setIdAutorizacion(pago.getIdAutorizacion());
168         returned_pago.setCodRespuesta(pago.getCodRespuesta());
169
170         return returned_pago;
171     }
172 }

```

En el fichero **ProcesaPago.java**:

```

1  @Override
2  protected void processRequest(HttpServletRequest request,
3  HttpServletResponse response)
4  throws ServletException, IOException {
5
6      TarjetaBean tarjeta = creaTarjeta(request);
7      ValidadorTarjeta val = new ValidadorTarjeta();
8      PagoBean pago = null;
9
10     // printAddresses(request, response);
11     if (! val.esValida(tarjeta)) {
12         request.setAttribute(val.getErrorName(), val.getErrorVisa());
13         reenvia("/formdatosvisa.jsp", request, response);
14         return;
15     }
16
17     //String url_from_xml;
18
19     //VisaDAOWSService service = new VisaDAOWSService();
20     //VisaDAOWS dao = service.getVisaDAOWSPort();
21
22     //url_from_xml=getServletContext().getInitParameter("webmaster");
23
24     //BindingProvider bp = (BindingProvider) dao;
25     //bp.getRequestContext().put(BindingProvider.
26     ENDPOINT_ADDRESS_PROPERTY, url_from_xml);
27     //VisaDAO dao = new VisaDAO();
28     try{
29         HttpSession session = request.getSession(false);
30         if (session != null) {
31             pago = (PagoBean) session.getAttribute(
32                 ComienzaPago.ATTRPAGO);
33         }
34         if (pago == null) {

```

```

32         pago = creaPago(request);
33         boolean isdebug = Boolean.valueOf(request.
34             getParameter("debug"));
35         dao.setDebug(isdebug);
36         boolean isdirectConnection = Boolean.valueOf(
37             request.getParameter("directConnection"))
38             ;
39         dao.setDirectConnection(isdirectConnection);
40         boolean usePrepared = Boolean.valueOf(request
41             .getParameter("usePrepared"));
42         dao.setPrepared(usePrepared);
43     }
44     // Almacenamos la tarjeta en el pago
45     pago.setTarjeta(tarjeta);
46
47     if (!dao.compruebaTarjeta(tarjeta)) {
48         enviaError(new Exception("Tarjeta no autorizada:"),
49             request, response);
50         return;
51     }
52     pago = dao.realizaPago(pago);
53     if (pago == null) {
54         enviaError(new Exception("Pago incorrecto"), request,
55             response);
56         return;
57     }
58     request.setAttribute(ComienzaPago.ATTRPAGO, pago);
59     if (sesion != null) sesion.invalidate();
60     reenvia("/pagoexitos.jsp", request, response);
61     return;
62 } catch (Exception e){
63     HttpSession sesion = request.getSession(false);
64     if (sesion != null) sesion.invalidate();
65     enviaError(new Exception("Pago incorrecto: "+e.getMessage()),
66         request, response);
67 }
68 }

```

## 10. Ejercicio 8

**Ejercicio 8.** Desplegar y probar la nueva aplicación creada.

- Probar a realizar pagos correctos. Comprobar que disminuye el saldo de las tarjetas sobre las que realice operaciones. Añadir a la memoria las evidencias obtenidas.
- Realice una operación con identificador de transacción y de comercio duplicados. Compruebe que el saldo de la tarjeta especificada en el pago no se ha variado.
- Incluya en la memoria de prácticas todos los pasos necesarios para resolver este ejercicio así como las evidencias obtenidas. Se pueden incluir por ejemplo capturas de pantalla.

### Pasos realizados:

Los pasos realizados los veremos en las capturas a continuación.

### Evidencias:

	numerotarjeta	titular	validadesde	validahasta	codigoverificacion	saldo
1	1111 2222 3333 4444	Jose Garcia	11/09	11/20	123	1000

Figura 16: Saldo inicial de la tarjeta.

Sistema de Pago con tarjeta × +

← → ↻ 🏠 ⓘ 10.1.1.2:8080/P1-ejb-cliente/comienzapago

## Pago con tarjeta

Numero de visa:

Titular:

Fecha Emisión:

Fecha Caducidad:

CVV2:

---

Id Transacción: 123  
Id Comercion: 2  
Importe: 99.0

---

Prácticas de Sistemas Informáticos II

Figura 17: Formulario de pago.

Sistema de Pago con tarjeta × +

← → ↻ 🏠 ⓘ 10.1.1.2:8080/P1-ejb-cliente/procesapago

## Pago con tarjeta

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

idTransaccion:  
idComercio: 2  
importe: 99.0  
codRespuesta: 000  
idAutorizacion: 1

[Volver al comercio](#)

---

Prácticas de Sistemas Informáticos II

Figura 18: Pago correcto.

86	1111 2222 3333 4444	Jose Garcia	11/09	11/20	123	901
----	---------------------	-------------	-------	-------	-----	-----

Figura 19: Saldo de la tarjeta tras la transacción.

10.1.1.2:8080/P1-ejb-cliente × +

← → ↻ 🏠 ⓘ 10.1.1.2:8080/P1-ejb-cliente/pago.html

Id Transacción:

Id Comercio:

Importe:

Figura 20: Probamos a hacer una transacción con ID repetido.

Sistema de Pago con tarjeta × +

← → ↻ 🏠 ⓘ 10.1.1.2:8080/P1-ejb-cliente/comienzapago

## Pago con tarjeta

Numero de visa:

Titular:

Fecha Emisión:

Fecha Caducidad:

CVV2:

---

Id Transacción: 123  
Id Comercion: 2  
Importe: 101.0

---

Prácticas de Sistemas Informáticos II

Figura 21: Formulario de pago.





Figura 22: Error en el pago, ponemos el error devuelto por la base de datos por razones de debugging, en una aplicación real esto sería una muy mala práctica por razones de seguridad.

86	1111 2222 3333 4444	Jose Garcia	11/09	11/20	123	901
----	---------------------	-------------	-------	-------	-----	-----

Figura 23: El saldo de la tarjeta tras el pago erróneo permanece igual.

# 11. Ejercicio 9

**Ejercicio 9.** En la máquina virtual donde se encuentra el servidor de aplicaciones (10.X.Y.2), declare manualmente la factoría de conexiones empleando la consola de administración, tal y como se adjunta en la **Figura 4**.

Incluye una captura de pantalla donde se muestre dicha consola de administración con los cambios solicitados.

## Evidencias:

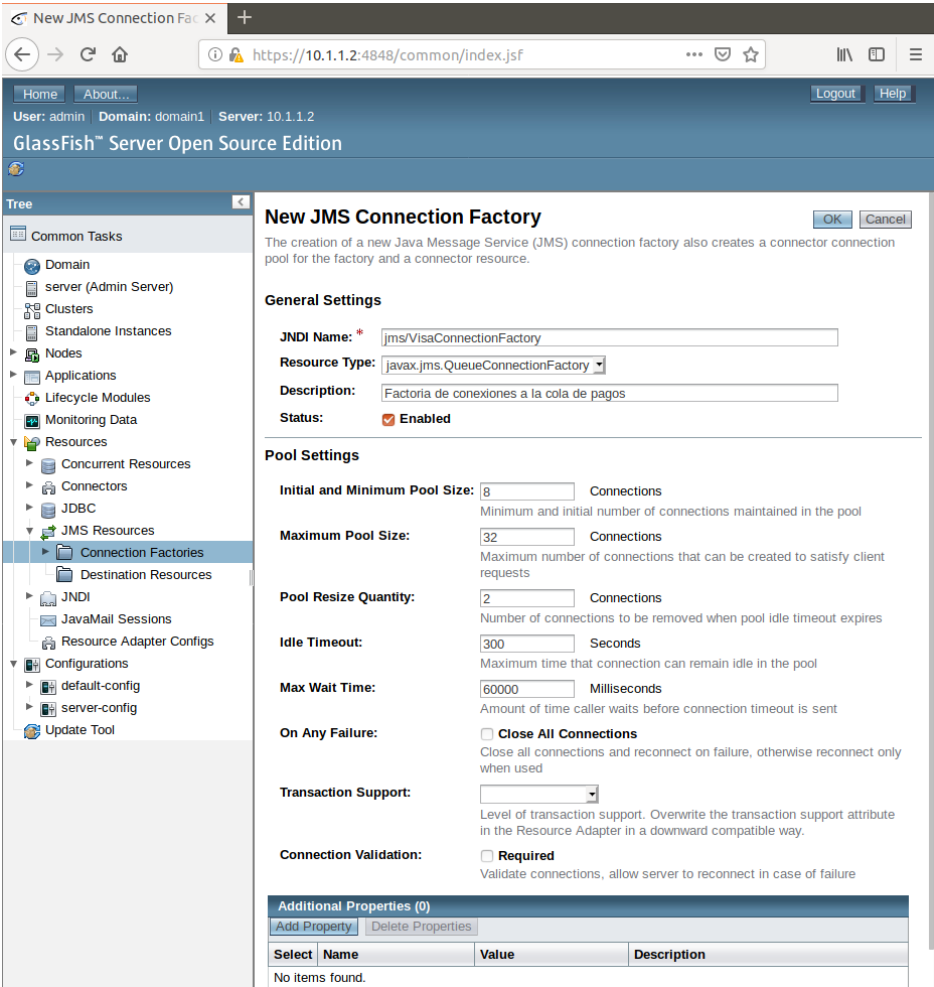


Figura 24: Creación de una Connection Factory.

Connection Factories (2)				
<div><div></div><div>New</div><div>Delete</div><div>Enable</div><div>Disable</div></div>				
Select	JNDI Name	Logical JNDI Name	Enabled	Resource Type
<input type="checkbox"/>	jms/_defaultConnectionFactory	java.comp.DefaultJMSConnectionFactory	<input checked="" type="checkbox"/>	javax.jms.ConnectionFactory
<input type="checkbox"/>	jms/VisaConnectionFactory		<input checked="" type="checkbox"/>	javax.jms.QueueConnectionFactory

Figura 25: Prueba de que la Connection Factory se ha creado con éxito.

## 12. Ejercicio 10

**Ejercicio 10.** En la máquina virtual donde se encuentra el servidor de aplicaciones (10.X.Y.2), declare manualmente la conexión empleando la consola de administración, tal y como se adjunta en la **Figura 5**.

Incluye una captura de pantalla donde se muestre dicha consola de administración con los cambios solicitados.

### Evidencias:

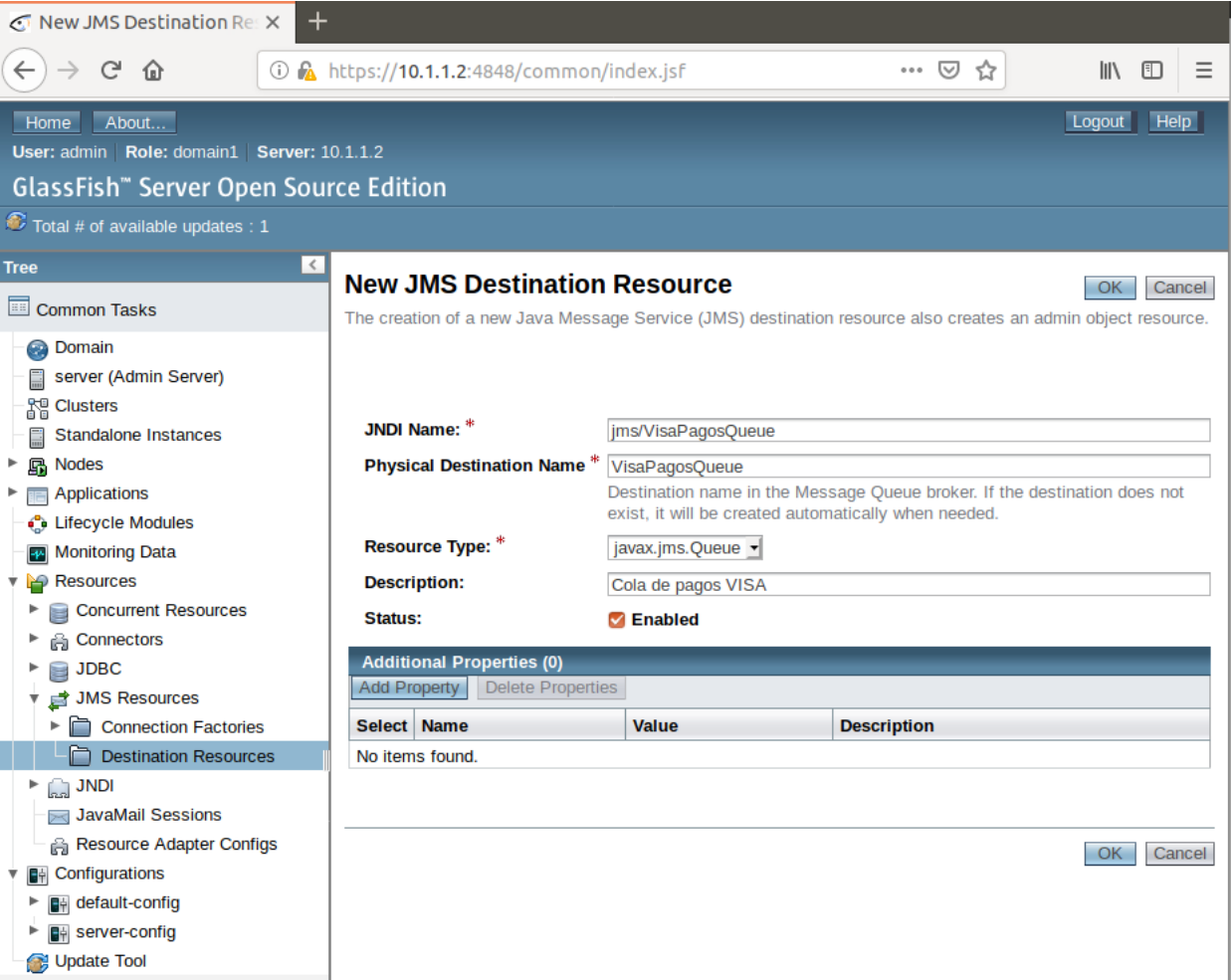


Figura 26: Creación de una Cola de Mensajes.

Select	JNDI Name	Enabled	Resource Type	Description
<input type="checkbox"/>	jms/VisaPagosQueue	<input checked="" type="checkbox"/>	javax.jms.Queue	Cola de pagos VISA

Figura 27: Prueba de que la Cola de Mensajes se ha creado con éxito.

## 13. Ejercicio 11

### Ejercicio 11.

- Modifique el fichero sun-ejb-jar.xml para que el MDB conecte adecuadamente a su *connection factory*.
- Incluya en la clase VisaCancelacionJMSBean:
  - Consulta SQL necesaria para actualizar el código de respuesta a valor 999, de aquella autorización existente en la tabla de pagos cuyo idAutorizacion coincida con lo recibido por el mensaje.
  - Consulta SQL necesaria para rectificar el saldo de la tarjeta que realizó el pago.
  - Método onMessage() que implemente ambas actualizaciones. Para ello tome de ejemplo el código SQL de ejercicios anteriores, de modo que se use un *prepared statement* que haga bind del idAutorizacion para cada mensaje recibido.
  - Control de errores en el método onMessage.

Incluye en la memoria cada fragmento de código donde se han ido añadiendo las modificaciones solicitadas.

### Código modificado:

En el fichero **sun-ejb-jar.xml**:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE sun-ejb-jar PUBLIC "-//Sun Microsystems, Inc.//DTD
   Application Server 9.0 EJB 3.0//EN" "http://www.sun.com/software/
   appserver/dtds/sun-ejb-jar_3_0-0.dtd">
3 <sun-ejb-jar>
4   <enterprise-beans>
5     <ejb>
6       <ejb-name>VisaCancelacionJMSBean</ejb-name>
7       <mdb-connection-factory>
8         <jndi-name>jms/VisaConnectionFactory</jndi-name>
9       </mdb-connection-factory>
10    </ejb>
11  </enterprise-beans>
12 </sun-ejb-jar>
```

En el fichero **VisaCancelacionJMSBean.java**:

```
1 package ssii2.visa;
2
3 import java.sql.Connection;
4 import java.sql.PreparedStatement;
5 import java.sql.ResultSet;
6 import java.sql.SQLException;
7 import javax.ejb.EJBException;
8 import javax.ejb.MessageDriven;
```

```

9 import javax.ejb.MessageDrivenContext;
10 import javax.ejb.ActivationConfigProperty;
11 import javax.jms.MessageListener;
12 import javax.jms.Message;
13 import javax.jms.TextMessage;
14 import javax.jms.JMSException;
15 import javax.annotation.Resource;
16 import java.util.logging.Logger;
17
18 /**
19  * @author jaime
20  */
21 @MessageDriven(mappedName = "jms/VisaPagosQueue")
22 public class VisaCancelacionJMSBean extends DBTester implements
    MessageListener {
23     static final Logger logger = Logger.getLogger("VisaCancelacionJMSBean
        ");
24     @Resource
25     private MessageDrivenContext mdc;
26
27     private static final String UPDATE_CANCELA_QRY =
28         "update pago set codRespuesta=? where idAutorizacion=?";
29
30     private static final String SELECT_SALDO_TARJETA_QRY =
31         "select numeroTarjeta, importe from pago where idAutorizacion=?";
32
33     private static final String ROLLBACK_SALDO_QRY =
34         "update tarjeta set saldo=saldo+? where numeroTarjeta=?";
35
36     public VisaCancelacionJMSBean() {
37     }
38
39     public void onMessage(Message inMessage) {
40         TextMessage msg = null;
41         Connection con = null;
42         ResultSet rs = null;
43         String qry = null;
44         PreparedStatement pstmt = null;
45         boolean ret=true;
46         String numero_tarjeta="";
47         double importe_pago_saldo=0.0;
48
49         try {
50             if (inMessage instanceof TextMessage) {
51                 msg = (TextMessage) inMessage;
52                 logger.info("MESSAGE BEAN: Message received: " + msg.
                    getText());
53
54                 con = getConnection();
55
56                 String cancela_qry=UPDATE_CANCELA_QRY;
57                 pstmt = con.prepareStatement(cancela_qry);
58                 String cod_respuesta = "999";
59                 pstmt.setString(1, cod_respuesta);
60                 pstmt.setInt(2, Integer.parseInt(msg.getText()));
61                 ret = false;
62                 if (!pstmt.execute()
63                     && pstmt.getUpdateCount() == 1) {

```

```

64         ret = true;
65     }
66
67     String select_qry=SELECT_SALDO_TARJETA_QRY;
68     pstmt = con.prepareStatement(select_qry);
69     pstmt.setInt(1, Integer.parseInt(msg.getText()));
70     rs = pstmt.executeQuery();
71     if(rs.next()){
72         numero_tarjeta=rs.getString("numeroTarjeta");
73         importe_pago_saldo=rs.getDouble("importe");
74     }
75
76     String rollback_qry=ROLLBACK_SALDO.QRY;
77     pstmt = con.prepareStatement(rollback_qry);
78     pstmt.setDouble(1, importe_pago_saldo);
79     pstmt.setString(2, numero_tarjeta);
80     ret = false;
81     if (!pstmt.execute()
82         && pstmt.getUpdateCount() == 1) {
83         ret = true;
84     }
85     } else {
86         logger.warning(
87             "Message of wrong type: "
88             + inMessage.getClass().getName());
89     }
90 } catch (JMSEException e) {
91     e.printStackTrace();
92     mdc.setRollbackOnly();
93 } catch (Throwable te) {
94     te.printStackTrace();
95 }
96 try{
97     if (rs != null) {
98         rs.close(); rs = null;
99     }
100    if (pstmt != null) {
101        pstmt.close(); pstmt = null;
102    }
103    if (con != null) {
104        closeConnection(con); con = null;
105    }
106 }
107 catch(Exception e){
108
109 }
110 }
111
112
113 }

```

## 14. Ejercicio 12

**Ejercicio 12.** Implemente ambos métodos en el cliente proporcionado. Deje comentado el método de acceso por la clase *InitialContext* de la API de JNDI. Indique en la memoria de prácticas qué ventajas podrían tener uno u otro método.

Incluye en la memoria cada fragmento de código donde se han ido añadiendo las modificaciones solicitadas.

### Código modificado:

En el fichero **VisaQueueMessageProducer.java**:

```
1 //...
2 public class VisaQueueMessageProducer {
3
4     @Resource(mappedName = "jms/VisaConnectionFactory")
5     private static ConnectionFactory connectionFactory;
6
7     @Resource(mappedName = "jms/VisaPagosQueue")
8     private static Queue queue;
9
10    // Metodo de prueba
11    public static void browseMessages(Session session)
12    {
13        try
14        {
15            Enumeration messageEnumeration;
16            TextMessage textMessage;
17            QueueBrowser browser = session.createBrowser(queue);
18            messageEnumeration = browser.getEnumeration();
19            if (messageEnumeration != null)
20            {
21                if (!messageEnumeration.hasMoreElements())
22                {
23                    System.out.println("Cola de mensajes vacia!");
24                }
25                else
26                {
27                    System.out.println("Mensajes en cola:");
28                    while (messageEnumeration.hasMoreElements())
29                    {
30                        textMessage =
31                            (TextMessage) messageEnumeration.nextElement();
32                        System.out.println(textMessage.getText());
33                    }
34                }
35            }
36        }
37        catch (Exception e)
38        {
39            e.printStackTrace();
40        }
41    }
42
43    public static void main(String[] args) {
```

```

44      Connection connection = null;
45      Session session = null;
46      MessageProducer messageProducer = null;
47      TextMessage message = null;
48
49      if (args.length != 1) {
50          System.err.println("Uso: VisaQueueMessageProducer [-browse |
51              <msg>]");
52          return;
53      }
54
55      try {
56          //InitialContext jndi = new InitialContext();
57
58          //connectionFactory=(ConnectionFactory)jndi.lookup("jms/
59              VisaConnectionFactory");
60          //queue = (Queue)jndi.lookup("jms/VisaPagosQueue");
61
62          connection = connectionFactory.createConnection();
63          session = connection.createSession(false, Session.
64              AUTO_ACKNOWLEDGE);
65          if (args[0].equals("-browse")) {
66              browseMessages(session);
67          }
68      } catch (Exception e) {
69          e.printStackTrace();
70      }
71
72      //...

```

La ventaja del método estático será la velocidad de ejecución, ya que todos los nombres quedan resueltos en tiempo de compilación; mientras que la ventaja del dinámico es que en tiempo de compilación no tenemos porqué conocer el nombre del recurso, ya que el nombre se resuelve en tiempo de ejecución. Por lo tanto es más adaptable a cambios.



## 15. Ejercicio 13

**Ejercicio 13.** Automatice la creación de los recursos JMS (cola y factoría de conexiones) en el `build.xml` y `jms.xml`. Para ello, indique en `jms.properties` los nombres de ambos y el Physical Destination Name de la cola de acuerdo a los valores asignados en los ejercicios 7 y 8. Recuerde también asignar las direcciones IP adecuadas a las variables `as.host.mdb` (`build.properties`) y `as.host.server` (`jms.properties`). ¿Por qué ha añadido esas IPs?

Borre desde la consola de administración de Glassfish la `connectionFactory` y la cola creadas manualmente y ejecute:

```
cd P1-jms
ant todo
```

Compruebe en la consola de administración del Glassfish que, efectivamente, los recursos se han creado automáticamente. Incluye una captura de pantalla, donde se muestre la consola de administración con los recursos creados. Revise el fichero `jms.xml` y anote en la memoria de prácticas cuál es el comando equivalente para crear una cola JMS usando la herramienta `asadmin`.


### ¿Por qué ha añadido esas IPs?

Hemos puesto el valor 10.1.1.2 a `as.host.mdb` con el objetivo de que los recursos JMS se creen en la segunda máquina virtual. Además, dado que el servidor de aplicaciones sigue corriendo en la 10.1.1.2 desde el ejercicio anterior, hemos tenido que darle el valor 10.1.1.2 a `as.host.server`.

### Comando equivalente usando `asadmin`:

```
$ asadmin --user admin --passwordfile path/to/passwordfile
--host 10.1.1.2 --port 4848 create-jms-resource --restype javax.jms.Queue --enabled=true --property VisaPagosQueue jms/-
VisaPagosQueue
```

### Evidencias:



Connection Factories (1)				
<input checked="" type="checkbox"/> <input type="checkbox"/>   <input type="button" value="New..."/> <input type="button" value="Delete"/> <input type="button" value="Enable"/> <input type="button" value="Disable"/>				
Select	JNDI Name	Logical JNDI Name	Enabled	Resource Type
<input type="checkbox"/>	jms/_defaultConnectionFactory	java:comp/DefaultJMSConnectionFactory	<input checked="" type="checkbox"/>	javax.jms.ConnectionFactory

Figura 28: Connection Factories iniciales.

Connection Factories (2)				
			New...	Delete Enable Disable
Select	JNDI Name	Logical JNDI Name	Enabled	Resource Type
<input type="checkbox"/>	jms/_defaultConnectionFactory	java:comp/DefaultJMSConnectionFactory	✓	javax.jms.Con
<input type="checkbox"/>	jms/VisaConnectionFactory		✓	javax.jms.Que

Figura 29: Connection Factories tras la creación automática.

Destination Resources (0)					
New...	Delete	Enable	Disable		
Select	JNDI Name	Status	Enabled	Resource Type	Description
No items found.					

Figura 30: Recursos iniciales.

Destination Resources (1)				
			New...	Delete Enable Disable
Select	JNDI Name	Enabled	Resource Type	Description
<input type="checkbox"/>	jms/VisaPagosQueue	✓	javax.jms.Queue	

Figura 31: Recursos tras la creación automática.

```
santi@santi-X550VXK:~/repos/si2/p2/P1-jms$ ant todo
Buildfile: /home/santi/repos/si2/p2/P1-jms/build.xml

GlassFish™ Server Open Source Edition
todo:
  Total # of available updates : 1

setup-jms:
  Common Tasks
create-jms-connection-factory:
  [exec] Connector resource jms/VisaConnectionFactory created.
  [exec] Command create-jms-connection-factory executed successfully.
  Clusters
create-jms-resource:
  [exec] Administered object jms/VisaPagosQueue created.
  [exec] Command create-jms-resource executed successfully.
  Applications
mdb: Lifecycle Modules
  Monitoring Data
montar-jerarquia:
  [mkdir] Created dir: /home/santi/repos/si2/p2/P1-jms/build
  [mkdir] Created dir: /home/santi/repos/si2/p2/P1-jms/build/clientjms
  [mkdir] Created dir: /home/santi/repos/si2/p2/P1-jms/build/mdb
  [mkdir] Created dir: /home/santi/repos/si2/p2/P1-jms/dist
  [mkdir] Created dir: /home/santi/repos/si2/p2/P1-jms/dist/clientjms
  [mkdir] Created dir: /home/santi/repos/si2/p2/P1-jms/dist/mdb
  JMS Resources
  JNDI
compilar-mdb: Sessions
  [javac] Compiling 2 source files to /home/santi/repos/si2/p2/P1-jms/build/mdb
  Configurations
preparar-meta-inf-mdb:
  [copy] Copying 2 files to /home/santi/repos/si2/p2/P1-jms/build/mdb
  server-config
empaquetar-mdb:
  [jar] Building jar: /home/santi/repos/si2/p2/P1-jms/dist/mdb/P1-jms-mdb.jar

desplegar-mdb:
  [exec] Application deployed with name P1-jms-mdb.
  [exec] Command deploy executed successfully.

clientjms:
montar-jerarquia:
compilar-clientjms:
  [javac] Compiling 1 source file to /home/santi/repos/si2/p2/P1-jms/build/clientjms
empaquetar-clientjms:
  [jar] Building jar: /home/santi/repos/si2/p2/P1-jms/dist/clientjms/P1-jms-clientjms.jar

BUILD SUCCESSFUL
Total time: 3 seconds
santi@santi-X550VXK:~/repos/si2/p2/P1-jms$
```

Figura 32: Ejecución de ant todo.

## 16. Ejercicio 14

**Ejercicio 14. Importante:** Detenga la ejecución del MDB con la consola de administración para poder realizar satisfactoriamente el siguiente ejercicio (check de 'Enabled' en Applications/P1-jms-mdb y guardar los cambios).

Modifique el cliente, VisaQueueMessageProducer.java, implementando el envío de args[0] como mensaje de texto (consultar los apéndices). Incluye en la memoria el fragmento de código que ha tenido que modificar.

Ejecute el cliente en el PC del laboratorio mediante el comando:

```
/opt/glassfish-4.1.2/glassfish/bin/appclient -targetserver  
10.X.Y.Z -client dist/clientjms/P1-jms-clientjms.jar idAu-  
torizacion
```

Donde 10.X.Y.Z representa la dirección IP de la máquina virtual en cuyo servidor de aplicaciones se encuentra desplegado el MDB. Para garantizar que el comando funcione correctamente es necesario fijar la variable (web console->Configurations->server-config->Java Message Service->JMS Hosts->default\_JMS\_host) que toma el valor "localhost" por la dirección IP de dicha máquina virtual. El cambio se puede llevar a cabo desde la consola de administración. Será necesario reiniciar el servidor de aplicaciones para que surja efecto.

Verifique el contenido de la cola ejecutando:

```
/opt/glassfish-4.1.2/glassfish/bin/appclient -targetserver  
10.X.Y.Z -client dist/clientjms/P1-jms-clientjms.jar -  
browse
```

Indique la salida del comando e inclúyala en la memoria de prácticas.

A continuación, volver a habilitar la ejecución del MDB y realizar los siguientes pasos:

- Realice un pago con la aplicación web
- Obtenga evidencias de que se ha realizado
- Cancelelo con el cliente
- Obtenga evidencias de que se ha cancelado y de que el saldo se ha rectificado

Al realizar este ejercicio en los laboratorios surge un error indicando que no es posible resolver el nombre del host local a una dirección IP. Esto se debe a que no hay una entrada con dicho nombre en el fichero `/etc/hosts` asociado a una dirección IP. Como dicho fichero no se puede editar, la solución es ejecutar el cliente de colas de mensajes desde la máquina virtual 1, para que se conecte a la máquina virtual 2. Basta con copiar el `.jar` del cliente a la máquina virtual, iniciar sesión de forma remota. Indicar donde se encuentra la versión 8 de java exportando la variable `JAVA_HOME` y ejecutar el cliente de colas con `appclient` desde la máquina virtual 1. Para ello, hay que ejecutar la siguiente secuencia de comandos:

Desde PC1 host:

```
$ scp dist/clientjms/P1-jms-clientjms.jar si2@10.X.Y.1:/tmp
```

Desde la máquina virtual 10.X.Y.1:

```
si2@si2srv01: $ export JAVA_HOME=/usr/lib/jvm/java-8-oracle/
```

```
si2@si2srv01: $ /opt/glassfish4/glassfish/bin/appclient  
-targetserver 10.X.Y.2 -client /tmp/P1-jms-clientjms.jar  
<idAutorizacion>
```

## Código modificado:

```
1 package ssii2;  
2 import javax.jms.ConnectionFactory;  
3 import javax.jms.Destination;  
4 import javax.jms.Queue;  
5 import javax.jms.Connection;  
6 import javax.jms.Session;  
7 import javax.jms.MessageProducer;  
8 import javax.jms.TextMessage;  
9 import javax.jms.JMSEException;  
10 import javax.annotation.Resource;  
11 import javax.jms.QueueBrowser;  
12 import java.util.Enumeration;  
13 import javax.naming.InitialContext;  
14  
15 public class VisaQueueMessageProducer {  
16  
17     @Resource(mappedName = "jms/VisaConnectionFactory")  
18     private static ConnectionFactory connectionFactory;  
19  
20     @Resource(mappedName = "jms/VisaPagosQueue")  
21     private static Queue queue;  
22  
23     // Metodo de prueba  
24     public static void browseMessages(Session session)  
25     {  
26         try  
27         {  
28             Enumeration messageEnumeration;  
29             TextMessage textMessage;
```

```

30 QueueBrowser browser = session.createBrowser(queue);
31 messageEnumeration = browser.getEnumeration();
32 if (messageEnumeration != null)
33 {
34     if (!messageEnumeration.hasMoreElements())
35     {
36         System.out.println("Cola de mensajes vacia!");
37     }
38     else
39     {
40         System.out.println("Mensajes en cola:");
41         while (messageEnumeration.hasMoreElements())
42         {
43             textMessage =
44                 (TextMessage) messageEnumeration.nextElement();
45             System.out.println(textMessage.getText());
46         }
47     }
48 }
49 }
50 catch (Exception e)
51 {
52     e.printStackTrace();
53 }
54 }
55
56 public static void main(String[] args) {
57     Connection connection = null;
58     Session session = null;
59     MessageProducer messageProducer = null;
60     TextMessage message = null;
61
62     if (args.length != 1) {
63         System.err.println("Uso: VisaQueueMessageProducer [-browse |
64             <msg>]");
65         return;
66     }
67     try {
68         //InitialContext jndi = new InitialContext();
69
70         //connectionFactory=(ConnectionFactory)jndi.lookup("jms/
71             VisaConnectionFactory");
72         //queue = (Queue)jndi.lookup("jms/VisaPagosQueue");
73
74         connection = connectionFactory.createConnection();
75         session = connection.createSession(false, Session.
76             AUTOACKNOWLEDGE);
77         if (args[0].equals("-browse")) {
78             browseMessages(session);
79         } else {
80             messageProducer=session.createProducer(queue);
81             message = session.createTextMessage();
82
83             message.setText(args[0]);
84             messageProducer.send(message);
85             messageProducer.close();
86             session.close();

```

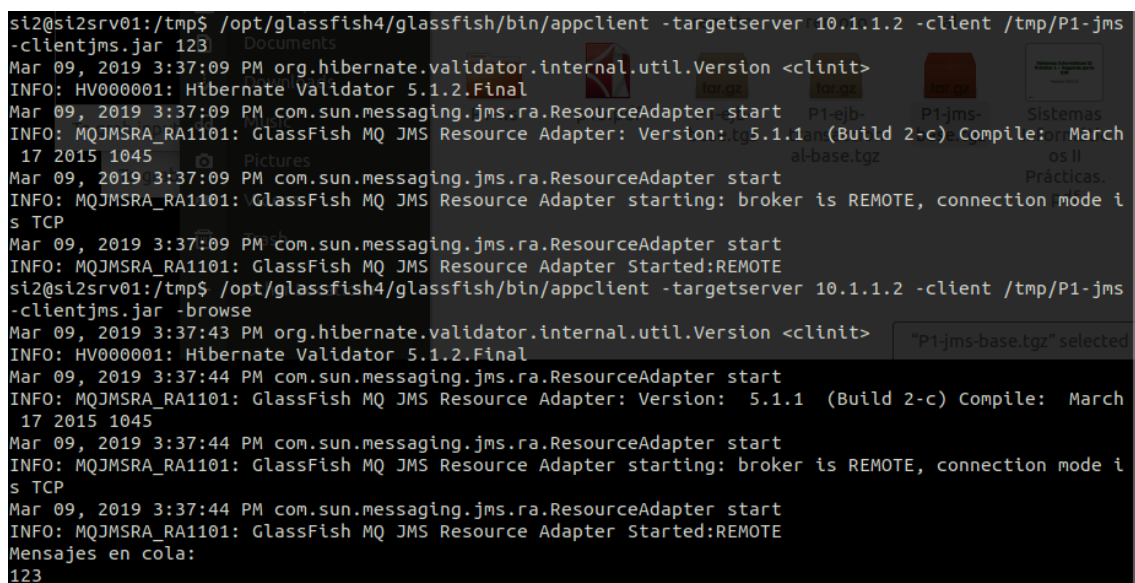
```

85     }
86   } catch (Exception e) {
87     System.out.println("Excepcion : " + e.toString());
88   } finally {
89     if (connection != null) {
90       try {
91         connection.close();
92       } catch (JMSEException e) {
93       }
94     } // if
95
96     System.exit(0);
97   } // finally
98 } // main
99 } // class

```

## Evidencias:

Comentar que hemos ejecutado el cliente desde la MV1, en vez de desde nuestra propia máquina, por la mayor facilidad de ejecutar comandos de asadmin, no obstante, es equivalente. Es decir, desde nuestro ordenador el resultado sería el mismo ya que lo único importante es que el cliente sea ejecutado desde una máquina distinta a la que corren los servidores JMS y de aplicaciones (en nuestro caso la 10.1.1.2) para que se ejecute en un escenario real (cliente y servicio en distintas máquinas). También, comentar que el valor que le hemos tenido que dar a la variable default\_JMS\_host es 10.1.1.2 para que la aplicación cliente ejecutada desde la MV1 no use la cola de manera local (el valor por defecto es localhost), sino que use el recurso creado en la 10.1.1.2 (host MDB) de manera que al mandar el mensaje el servidor ejecute el método onMessage().



```

si2@si2srv01:/tmp$ /opt/glassfish4/glassfish/bin/appclient -targetserver 10.1.1.2 -client /tmp/P1-jms
-clientjms.jar 123
Mar 09, 2019 3:37:09 PM org.hibernate.validator.internal.util.Version <clinit>
INFO: HV000001: Hibernate Validator 5.1.2.Final
Mar 09, 2019 3:37:09 PM com.sun.messaging.jms.ra.ResourceAdapter start
INFO: MQJMSRA_RA1101: GlassFish MQ JMS Resource Adapter: Version: 5.1.1 (Build 2-c) Compile: March
17 2015 1045
Mar 09, 2019 3:37:09 PM com.sun.messaging.jms.ra.ResourceAdapter start
INFO: MQJMSRA_RA1101: GlassFish MQ JMS Resource Adapter starting: broker is REMOTE, connection mode i
s TCP
Mar 09, 2019 3:37:09 PM com.sun.messaging.jms.ra.ResourceAdapter start
INFO: MQJMSRA_RA1101: GlassFish MQ JMS Resource Adapter Started:REMOTE
si2@si2srv01:/tmp$ /opt/glassfish4/glassfish/bin/appclient -targetserver 10.1.1.2 -client /tmp/P1-jms
-clientjms.jar -browse
Mar 09, 2019 3:37:43 PM org.hibernate.validator.internal.util.Version <clinit>
INFO: HV000001: Hibernate Validator 5.1.2.Final
Mar 09, 2019 3:37:44 PM com.sun.messaging.jms.ra.ResourceAdapter start
INFO: MQJMSRA_RA1101: GlassFish MQ JMS Resource Adapter: Version: 5.1.1 (Build 2-c) Compile: March
17 2015 1045
Mar 09, 2019 3:37:44 PM com.sun.messaging.jms.ra.ResourceAdapter start
INFO: MQJMSRA_RA1101: GlassFish MQ JMS Resource Adapter starting: broker is REMOTE, connection mode i
s TCP
Mar 09, 2019 3:37:44 PM com.sun.messaging.jms.ra.ResourceAdapter start
INFO: MQJMSRA_RA1101: GlassFish MQ JMS Resource Adapter Started:REMOTE
Mensajes en cola:
123

```

Figura 33: Cola funcionando desde la MV1.

Sistema de Pago con tarjeta × +

← → ↻ 🏠 ⓘ 10.1.1.2:8080/P1-ejb-cliente/comienzapago

## Pago con tarjeta

Numero de visa:

Titular:

Fecha Emisión:

Fecha Caducidad:

CVV2:

---

Id Transacción: 76  
Id Comercion: 72  
Importe: 23.0

---

Prácticas de Sistemas Informáticos II

Figura 34: Formulario de pago.

Sistema de Pago con tarjeta × +

← → ↻ 🏠 ⓘ 10.1.1.2:8080/P1-ejb-cliente/procesapago

## Pago con tarjeta

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

idTransaccion:  
idComercio: 72  
importe: 23.0  
codRespuesta: 000  
idAutorizacion: 3

[Volver al comercio](#)

---

Prácticas de Sistemas Informáticos II

Figura 35: Confirmación del pago.

1001	1111 2222 3333 4444	Jose Garcia	11/09	11/20	123	878
------	---------------------	-------------	-------	-------	-----	-----

Figura 36: Saldo de la tarjeta.



2	3	76	999	23	72	1111 2222 3333 4444	09/03/19 15:46
---	---	----	-----	----	----	---------------------	----------------

Figura 37: Pago denegado por el cliente (desde la MV1 como el que veremos en las Figuras 39 y 40 de manera más visual).

1001	1111 2222 3333 4444	Jose Garcia	11/09	11/20	123	901
------	---------------------	-------------	-------	-------	-----	-----

Figura 38: Saldo de la tarjeta restablecido.

	idautorizacion	idtransaccion	codrespuesta	importe	idcomercio	numerotarjeta	fecha
1	1	123	000	99	2	1111 2222 3333 4444	08/03/19 17:45
2	3	76	999	23	72	1111 2222 3333 4444	09/03/19 15:46
3	4	453	999	113	12	1111 2222 3333 4444	09/03/19 19:03

```
santi@santi-X550VXK: ~/repos/si2/p2/P1-jms
File Edit View Search Terminal Help
LC_MEASUREMENT = "es_ES.UTF-8",
LC_IDENTIFICATION = "es_ES.UTF-8",
LC_NUMERIC = "es_ES.UTF-8",
LC_PAPER = "es_ES.UTF-8",
LANG = "en_US.UTF-8"
are supported and installed on your system.
perl: warning: Falling back to the standard locale ("C").
si2@si2srv01:~$ ls
sockets
si2@si2srv01:~$ export JAVA_HOME=/usr/lib/jvm/java-8-oracle/
si2@si2srv01:~$ /opt/glassfish4/glassfish/bin/appclient -targetserver 10.1.1.2 -
client /tmp/P1-jms-clientjms.jar -browse
Mar 10, 2019 11:07:41 AM org.hibernate.validator.internal.util.Version <clinit>
INFO: HV000001: Hibernate Validator 5.1.2.Final
Mar 10, 2019 11:07:41 AM com.sun.messaging.jms.ra.ResourceAdapter start
INFO: MQJMSRA_RA1101: GlassFish MQ JMS Resource Adapter: Version: 5.1.1 (Build
(2-c) Compile: March 17 2015 1045
Mar 10, 2019 11:07:41 AM com.sun.messaging.jms.ra.ResourceAdapter start
INFO: MQJMSRA_RA1101: GlassFish MQ JMS Resource Adapter starting: broker is REMO
TE, connection mode is TCP
Mar 10, 2019 11:07:41 AM com.sun.messaging.jms.ra.ResourceAdapter start
INFO: MQJMSRA_RA1101: GlassFish MQ JMS Resource Adapter Started:REMOTE
Cola de mensajes vac?a!
si2@si2srv01:~$ /opt/glassfish4/glassfish/bin/appclient -targetserver 10.1.1.2 -
client /tmp/P1-jms-clientjms.jar 1
```

Figura 39: Ejecución del cliente y base de datos desde la MV1.

	idautorizacion	idtransaccion	codrespuesta	importe	idcomercio	numerotarjeta	fecha
1	3	76	999	23	72	1111 2222 3333 4444	09/03/19 15:46
2	4	453	999	113	12	1111 2222 3333 4444	09/03/19 19:03
3	1	123	999	99	2	1111 2222 3333 4444	08/03/19 17:45

```
santi@santi-X550VXK: ~/repos/si2/p2/P1-jms
File Edit View Search Terminal Help
si2@si2srv01:~$ /opt/glassfish4/glassfish/bin/appclient -targetserver 10.1.1.2 -
client /tmp/P1-jms-clientjms.jar -browse
Mar 10, 2019 11:07:41 AM org.hibernate.validator.internal.util.Version <clinit>
INFO: HV000001: Hibernate Validator 5.1.2.Final
Mar 10, 2019 11:07:41 AM com.sun.messaging.jms.ra.ResourceAdapter start
INFO: MQJMSRA_RA1101: GlassFish MQ JMS Resource Adapter: Version: 5.1.1 (Build
2-c) Compile: March 17 2015 1045
Mar 10, 2019 11:07:41 AM com.sun.messaging.jms.ra.ResourceAdapter start
INFO: MQJMSRA_RA1101: GlassFish MQ JMS Resource Adapter starting: broker is REMO
TE, connection mode is TCP
Mar 10, 2019 11:07:41 AM com.sun.messaging.jms.ra.ResourceAdapter start
INFO: MQJMSRA_RA1101: GlassFish MQ JMS Resource Adapter Started:REMOTE
Cola de mensajes vac?a!
si2@si2srv01:~$ /opt/glassfish4/glassfish/bin/appclient -targetserver 10.1.1.2 -
client /tmp/P1-jms-clientjms.jar 1
Mar 10, 2019 11:09:21 AM org.hibernate.validator.internal.util.Version <clinit>
INFO: HV000001: Hibernate Validator 5.1.2.Final
Mar 10, 2019 11:09:21 AM com.sun.messaging.jms.ra.ResourceAdapter start
INFO: MQJMSRA_RA1101: GlassFish MQ JMS Resource Adapter: Version: 5.1.1 (Build
2-c) Compile: March 17 2015 1045
Mar 10, 2019 11:09:21 AM com.sun.messaging.jms.ra.ResourceAdapter start
INFO: MQJMSRA_RA1101: GlassFish MQ JMS Resource Adapter starting: broker is REMO
TE, connection mode is TCP
Mar 10, 2019 11:09:21 AM com.sun.messaging.jms.ra.ResourceAdapter start
INFO: MQJMSRA_RA1101: GlassFish MQ JMS Resource Adapter Started:REMOTE
```

Figura 40: Pago denegado correctamente.