		Escuela Politécnica Superior Ingeniería Informática Prácticas de Sistemas Informáticos 2			
Grupo	**2401**	**Práctica**	4	**Fecha**	14/03/2017
Alumno/a		Hernando, Bernabé, Amanda			
Alumno/a		Pérez, Manso, Pablo			

Práctica 1B: Arquitectura de JAVA EE (Segunda Parte)

Cuestión número 1:

Editar el archivo VisaDAOLocal.java y comprobar la definición de dicha interfaz. Anote sus comentarios en la memoria.

Un cliente sólo puede acceder a un *EJB session* mediante los métodos definidos en la interfaz de negocio del *EJB*. La interfaz de negocio define la vista del cliente de un *EJB*. Todos los demás aspectos del *EJB* (implementaciones de métodos y configuración de implementación) se ocultan al cliente. En este caso, el tipo de acceso de cliente permitido por los *EJB* es local.

Ejercicio número 1:

Introduzca las siguientes modificaciones en el bean VisaDAOBean para convertirlo en un EJB stateless con interfaz local:

- **Declarar la clase con interfaz local y 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() / realizaPago() a la interfaz definida en VisaDAOLocal.**

```
public PagoBean realizaPago(PagoBean pago);
public PagoBean[] getPagos(String idComercio);
```

Para más detalle, los cambios se encuentran en el código entregado.

Ejercicio número 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:

```
import ssii2.visa.VisaDAOWSService; // Stub generado utomáticamente
import ssii2.visa.VisaDAOWS; // Stub generado automáticamente
```

- **Añadir como atributo de la clase que implementa el servlet 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()
  ```

Importante: Esta operación deberá ser realizada para todos los servlets del proyecto que hagan uso del antiguo VisaDAOWS. Verifique también posibles errores de compilación y ajustes necesarios en el código al cambiar la interfaz del antiguo VisaDAOWS (en particular, el método getPagos()).
Los cambios realizados se encuentran en el código adjunto.

Cuestión número 2:

Editar el archivo application.xml y comprobar 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 en la memoria.

- P1-ejb.jar (módulo EJB)
  Dentro del archivo empaquetado se ha introducido el descriptor del *EJB*, proporcionado en el código de la práctica en el archivo *conf/server/META-INF/sun-ejb*-jar.xml, además de las clases utilizadas por el EJB servidor.

```
e302505@3-12-65-79:~/Desktop/P1-ejb$ jar -tvf dist/server/P1-ejb.jar
     0 Tue Feb 28 17:08:20 CET 2017 META-INF/
   103 Tue Feb 28 17:08:18 CET 2017 META-INF/MANIFEST.MF
     0 Tue Feb 28 17:07:28 CET 2017 ssii2/
     0 Tue Feb 28 17:07:30 CET 2017 ssii2/visa/
     0 Tue Feb 28 17:07:30 CET 2017 ssii2/visa/dao/
   255 Tue Feb 28 17:08:20 CET 2017 META-INF/sun-ejb-jar.xml
  1464 Tue Feb 28 17:07:30 CET 2017 ssii2/visa/PagoBean.class
   856 Tue Feb 28 17:07:30 CET 2017 ssii2/visa/TarjetaBean.class
   593 Tue Feb 28 17:07:30 CET 2017 ssii2/visa/VisaDAOLocal.class
  1723 Tue Feb 28 17:07:28 CET 2017 ssii2/visa/dao/DBTester.class
  7096 Tue Feb 28 17:07:30 CET 2017 ssii2/visa/dao/VisaDAOBean.class
e302505@3-12-65-79:~/Desktop/P1-ejb$
```

- P1-ejb-cliente.war (módulo web)
  Un módulo web contiene archivos de clases de servlets, archivos web (JSP), archivos de clases adicionales (de error), archivos HTML y, opcionalmente, un descriptor de implementación de aplicaciones web que en nuestro caso no se ha introducido.

```
e302505@3-12-65-79:~/Desktop/P1-ejb$ jar -tvf dist/client/P1-ejb-cliente.war
     0 Tue Feb 28 17:17:26 CET 2017 META-INF/
   103 Tue Feb 28 17:17:24 CET 2017 META-INF/MANIFEST.MF
     0 Tue Feb 28 17:17:26 CET 2017 WEB-INF/
     0 Tue Feb 28 17:16:28 CET 2017 WEB-INF/classes/
     0 Tue Feb 28 17:17:12 CET 2017 WEB-INF/classes/ssii2/
     0 Tue Feb 28 17:17:12 CET 2017 WEB-INF/classes/ssii2/controlador/
     0 Tue Feb 28 17:17:12 CET 2017 WEB-INF/classes/ssii2/filtros/
     0 Tue Feb 28 17:17:12 CET 2017 WEB-INF/classes/ssii2/visa/
     0 Tue Feb 28 17:17:12 CET 2017 WEB-INF/classes/ssii2/visa/error/
     0 Tue Feb 28 17:06:30 CET 2017 WEB-INF/lib/
     0 Tue Feb 28 17:17:26 CET 2017 error/
  2844 Tue Feb 28 17:16:28 CET 2017 WEB-INF/classes/ssii2/controlador/ComienzaPago.class
  2244 Tue Feb 28 17:16:28 CET 2017 WEB-INF/classes/ssii2/controlador/DelPagos.class
  2142 Tue Feb 28 17:17:12 CET 2017 WEB-INF/classes/ssii2/controlador/GetPagos.class
  5503 Tue Feb 28 17:17:12 CET 2017 WEB-INF/classes/ssii2/controlador/ProcesaPago.class
  1894 Tue Feb 28 17:16:28 CET 2017 WEB-INF/classes/ssii2/controlador/ServletRaiz.class
  2608 Tue Feb 28 17:17:12 CET 2017 WEB-INF/classes/ssii2/filtros/CompruebaSesion.class
  3170 Tue Feb 28 17:17:12 CET 2017 WEB-INF/classes/ssii2/visa/ValidadorTarjeta.class
   616 Tue Feb 28 17:17:12 CET 2017 WEB-INF/classes/ssii2/visa/error/ErrorVisa.class
   198 Tue Feb 28 17:17:12 CET 2017 WEB-INF/classes/ssii2/visa/error/ErrorVisaCVV.class
   209 Tue Feb 28 17:17:12 CET 2017 WEB-INF/classes/ssii2/visa/error/ErrorVisaFechaCaducidad.class
   207 Tue Feb 28 17:17:12 CET 2017 WEB-INF/classes/ssii2/visa/error/ErrorVisaFechaEmision.class
   201 Tue Feb 28 17:17:12 CET 2017 WEB-INF/classes/ssii2/visa/error/ErrorVisaNumero.class
   202 Tue Feb 28 17:17:12 CET 2017 WEB-INF/classes/ssii2/visa/error/ErrorVisaTitular.class
  6170 Tue Feb 28 17:17:26 CET 2017 WEB-INF/web.xml
   455 Tue Feb 28 17:17:26 CET 2017 borradoerror.jsp
   501 Tue Feb 28 17:17:26 CET 2017 borradook.jsp
   509 Tue Feb 28 17:17:26 CET 2017 cabecera.jsp
   283 Tue Feb 28 17:17:26 CET 2017 error/muestraerror.jsp
  2729 Tue Feb 28 17:17:26 CET 2017 formdatosvisa.jsp
  1257 Tue Feb 28 17:17:26 CET 2017 listapagos.jsp
  1178 Tue Feb 28 17:17:26 CET 2017 pago.html
  1142 Tue Feb 28 17:17:26 CET 2017 pagoexito.jsp
   104 Tue Feb 28 17:17:26 CET 2017 pie.html
  5011 Tue Feb 28 17:17:26 CET 2017 testbd.jsp
e302505@3-12-65-79:~/Desktop/P1-ejb$
```

- P1-ejb.ear
  El archivo empaquetado contiene los módulos Java EE anteriormente nombrados y un descriptor de despliegue (application.xml).

```
e302505@3-12-65-79:~/Desktop/P1-ejb$ jar -tvf dist/P1-ejb.ear
     0 Tue Feb 28 17:19:54 CET 2017 META-INF/
   103 Tue Feb 28 17:19:52 CET 2017 META-INF/MANIFEST.MF
   508 Sat Feb 11 23:33:00 CET 2012 META-INF/application.xml
 21920 Tue Feb 28 17:17:26 CET 2017 P1-ejb-cliente.war
  7020 Tue Feb 28 17:08:20 CET 2017 P1-ejb.jar
e302505@3-12-65-79:~/Desktop/P1-ejb$
```

- application.xml: es el descriptor de despliegue y describe la configuración de implementación de la aplicación y define sus componentes.

```
<?xml version="1.0" encoding="UTF-8"?>
<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">
  <display-name>P1-ejb</display-name>
  <module>
    <ejb>P1-ejb.jar</ejb>
  </module>
  <module>
    <web>
      <web-uri>P1-ejb-cliente.war</web-uri>
      <context-root>/P1-ejb-cliente</context-root>
    </web>
  </module>
</application>
```

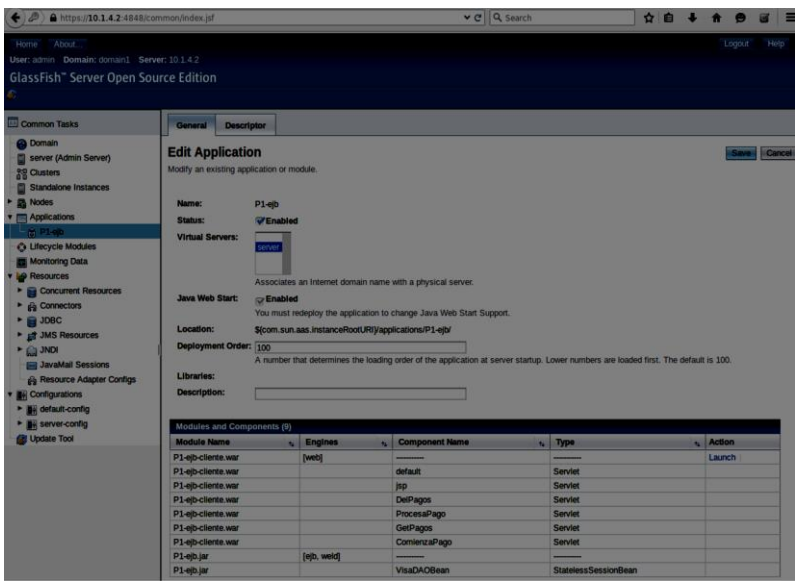
Ejercicio número 3:

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.**
  Como ambos el cliente y el servidor se encuentran en el PC2VM:
  `as.host.client=10.1.4.2`
  `as.host.server=10.1.4.2`
- **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.**
  Como la base de datos reside en el PC1VM:
  `db.host=10.1.4.1`
  `db.client.host=10.1.4.2`

Desplegar la aplicación de empresa.

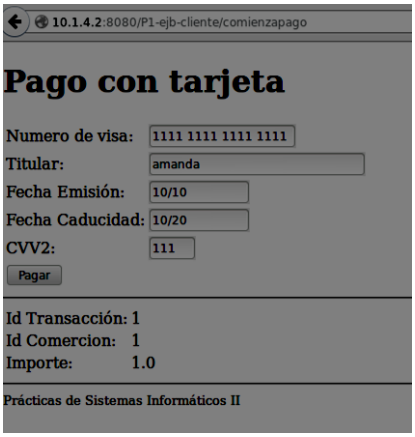
En la siguiente imagen observamos el correcto despliegue de la aplicación en la dirección 10.1.4.2:

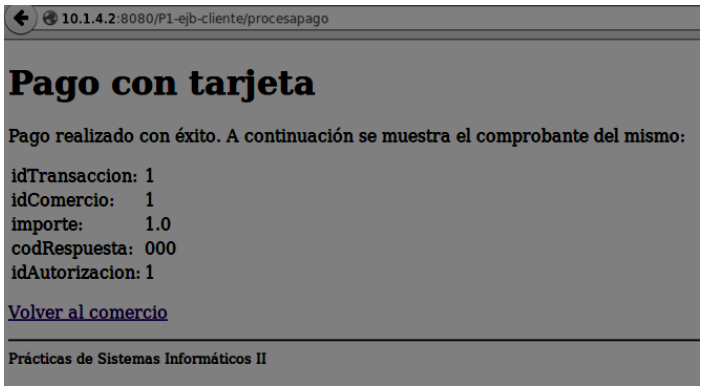


Ejercicio número 4:

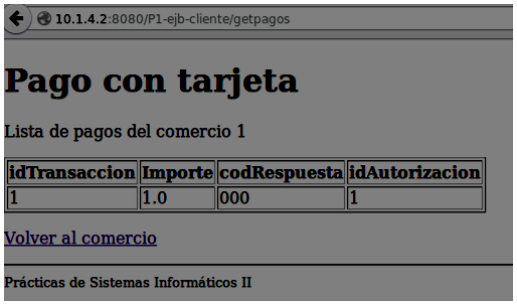
Comprobar el correcto funcionamiento de la aplicación mediante llamadas directas a través de las páginas *pago.html* y *testbd.jsp* (sin *directconnection*). Realice un pago. Lístelo. Elimínelo. Téngase en cuenta que la aplicación se habrá desplegado bajo la ruta /P1-ejb-cliente.

1. Realizamos el pago:





2. Comprobamos que se ha realizado correctamente:



3. Eliminamos el pago:

Ejercicio número 5:

Realizar los cambios indicados en P1-ejb-servidor-remoto y preparar los PCs con el esquema de máquinas virtuales indicado y 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.

Para implementar el servidor remoto hemos definido primero de todo, la interfaz remota del EJB (con los mismos métodos que la local), hemos hecho que VisaDAOBean implemente ambas interfaces, la local y la remota y por último, hemos marcado como serializables PagoBean y TarjetaBean.

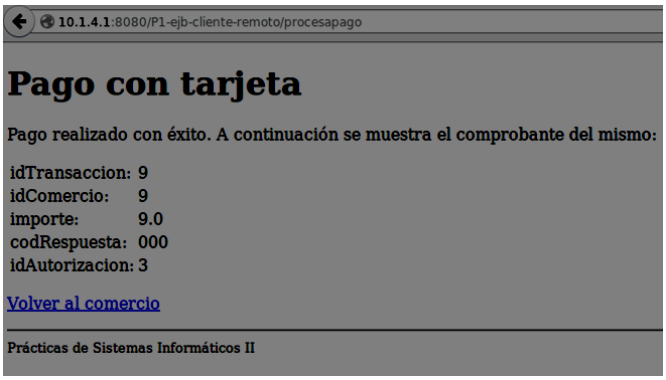
No hemos modificado ninguna dirección IP ya que el esquema de máquinas virtuales no ha variado. Hemos compilado, empaquetado y desplegado para comprobar el correcto despliegue de la aplicación servidor.

Ejercicio número 6:

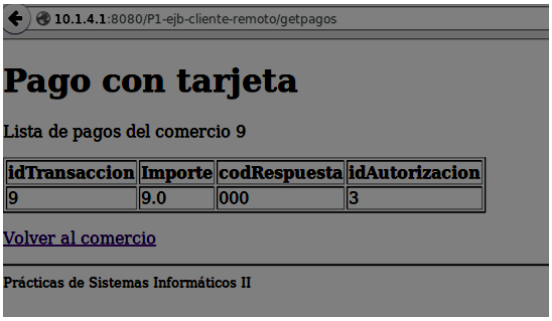
Realizar los cambios comentados en la aplicación P1-base para convertirla en P1-EJB-clienteremoto y compilar, empaquetar y desplegar de nuevo la aplicación en otra máquina virtual distinta a la de la aplicación servidor, es decir, esta aplicación cliente estará desplegada en la MV del PC1 tal y como se muestra en 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.

Para implementar el cliente remoto hemos seguido los pasos descritos en el enunciado.
Primero hemos eliminado el directorio *ssii2/visa/dao* ya que ahora toda la lógica se invoca de forma remota. En los *servlets* que invocan la lógica de negocio hemos eliminado la declaración de *VisaDAO dao* para insertar una referencia al objeto remoto obtenida por inyección. Además, hemos marcado como serializables *PagoBean* y *TarjetaBean* y hemos declarado la interfaz remota *VisaDAORemote.java*.
Por último, hemos creado un fichero *glassfish-web.xml* que especifica cómo se resuelven las referencias a los *EJBs* remotos y cómo se invocan los métodos remotos.
Como el cliente se despliega en la MVPC1 *as.host* debe ser 10.1.4.1.
Para comprobar el correcto despliegue de ambas aplicaciones seguimos los siguientes pasos:

1. Realizamos el pago.



2. Listamos el pago.

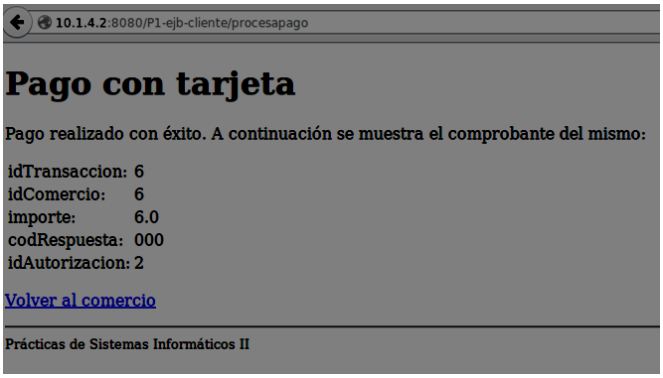


3. Eliminamos el pago.

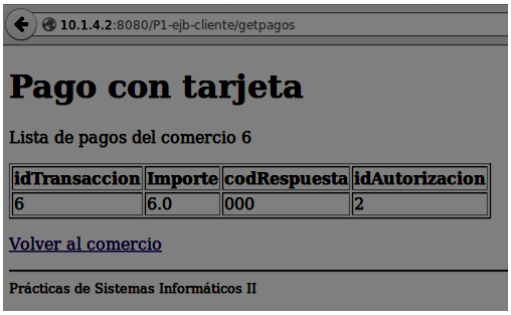
4. Comprobamos la correcta eliminación.

Además, hemos comprobado que el cliente local sigue funcionando correctamente. Para ello, hemos seguido los mismos pasos que antes, pero ahora conectándonos a la dirección 10.1.4.2.

1. Realizamos el pago.



2. Listamos el pago.



3. Eliminamos el pago.

4. Comprobamos la correcta eliminación.

Ejercicio número 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;

public double getSaldo() {
        return saldo;  }

public void setSaldo(double saldo) {
        this.saldo = 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 la tarjeta de la base de datos.**

```
private static final String SELECT_SALDO_QRY =
                "select saldo from tarjeta " +
                "where numeroTarjeta=? ";
```

- **Declarar un *prepared statement* para insertar el nuevo saldo calculado en la base de datos.**

```
private static final String UPDATE_SALDO_QRY =
                "update tarjeta " +
                "set saldo=? " +
                "where numeroTarjeta=? ";
```

- **Modificar el método realizaPago con las siguientes acciones:**
  - **Recuperar el saldo de la tarjeta a través del *prepared statement* declarado anteriormente.**

```
String select = SELECT_SALDO_QRY;
double saldo;
errorLog(select);
pstmt = con.prepareStatement(select);
pstmt.setString(1, pago.getTarjeta().getNumero());
ResultSet rs0 = pstmt.executeQuery();
```

  - **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).**

```
if (rs0.next()) {
     saldo = rs0.getDouble("saldo");
     if (saldo < pago.getImporte()) {
         return null;
     }
}else {
     throw new EJBException("No se ha encontrado la tarjeta.");
}
```

  - **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.**

```
String update = UPDATE_SALDO_QRY;
errorLog(update);
pstmt = con.prepareStatement(update);
pstmt.setDouble(1, saldo - pago.getImporte());
pstmt.setString(2, pago.getTarjeta().getNumero());
pstmt.execute();
```

  - **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.**

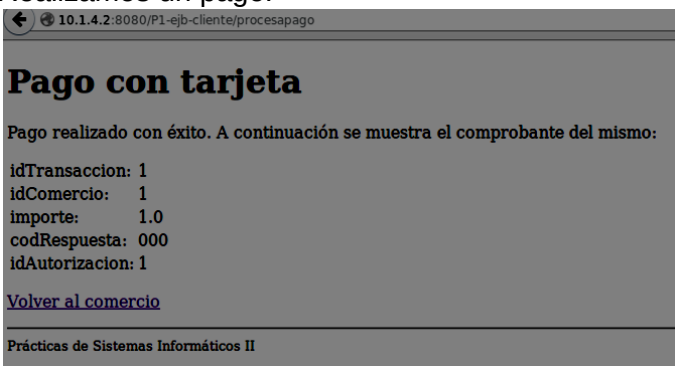
- **Modificar el *servlet* ProcesaPago para que capture la posible interrupción EJBException lanzada por realizaPago, y, en caso de que se haya lanzado, devuelva la página de error mediante el método enviaError (recordar antes de retornar que se debe invalidar la sesión, si es que existe):**

```
try{
    pago =  dao.realizaPago(pago);
}catch (EJBException e){
    if (sesion != null) sesion.invalidate();
     enviaError(new Exception("Pago incorrecto"), request, response);
     return;  }
```

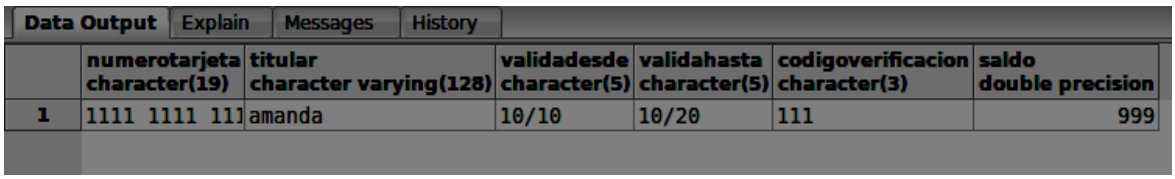
Ejercicio número 8:

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.**
  1. Realizamos un pago.



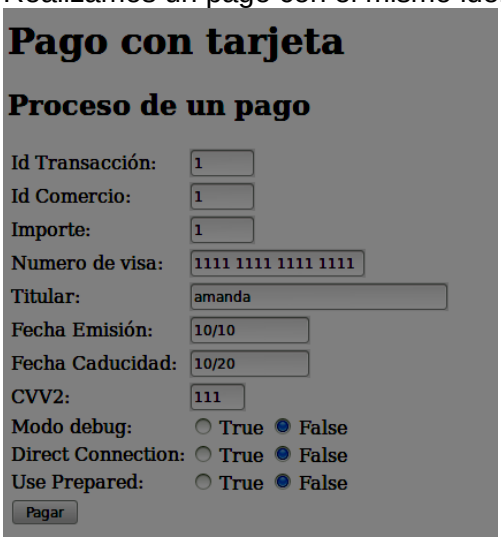
  2. Comprobamos que se ha realizado correctamente. Efectivamente, el saldo ha sido decrementado en 1 euro, quedando así 999 euros de saldo.



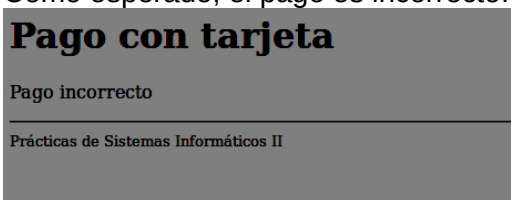
Data Output | Explain | Messages | History

	numerotarjeta character(19)	titular character varying(128)	validadesde character(5)	validahasta character(5)	codigoverificacion character(3)	saldo double precision
1	1111 1111 111	amanda	10/10	10/20	111	999

- **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.**
  1. Realizamos un pago con el mismo identificador de transacción y de comercio:



2. Como esperado, el pago es incorrecto.



Ejercicio número 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.

Ejercicio número 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.

Ejercicio número 11:

Modifique el fichero sun-ejb-jar.xml para que el MDB conecte adecuadamente a su connection Factory.

```
<ejb>
      <ejb-name>VisaCancelacionJMSBean</ejb-name>
      <mdb-connection-factory>
            <jndi-name>jms/VisaConnectionFactory</jndi-name>
      </mdb-connection-factory>
</ejb>
```

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.**

```
private static final String UPDATE_PAGO_QRY = "update pago "+
                                              "set codRespuesta=999 "+
                                              "where idAutorizacion=?";
```

- **Consulta SQL necesaria para rectificar el saldo de la tarjeta que realizó el pago.**

```
private static final String UPDATE_TARJETA_QRY = "update tarjeta as t "+
                                    "set saldo = saldo + pago.importe "+
                                                 "from pago "+
                                       "where pago.idAutorizacion=? "+
                              "and pago.numeroTarjeta = t.numeroTarjeta";
```

- **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.**

```
String update_pago= UPDATE_PAGO_QRY;
pstmt = con.prepareStatement(update_pago);
pstmt.setInt(1,numero);
pstmt.execute();

String update_tarjeta= UPDATE_TARJETA_QRY;
pstmt = con.prepareStatement(update_tarjeta);
pstmt.setInt(1,numero);
pstmt.execute();
```

Ejercicio número 12:

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

El JNDI provee de transparencia de ubicación ya que l código de los clientes no depende de nombres o ubicaciones específicas de equipos. Además, si el recurso cambia de lugar, no es necesario modificar el código del cliente, el servicio de nombre actualizará el cambio.

Ejercicio número 13:

Automatice la creación de los recursos JMS (cola y connection factory) 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).

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. 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.

```
asadmin --user admin --passwordfile passwordfile --host 10.1.4.1 --port
4848" create-jms-resource --restype javax.jms.QueueConnectionFactory --
enabled=true --property Name=Visa jms/VisaPagosQueue
```

Ejercicio número 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). Ejecute el cliente en el PC del laboratorio mediante el comando:

```
/usr/local/glassfish-4.0/glassfish/bin/appclient –targetserver 10.X.Y.Z -
client dist/clientjms/P1-jms-clientjms.jar idAutorizacion
```

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:

```
/usr/local/glassfish-4.0/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**
- **Cancélelo con el cliente**
- **Obtenga evidencias de que se ha cancelado y de que el saldo se ha rectificado**

Hay que ejecutar la siguiente secuencia de comandos:

Desde PC1 host:

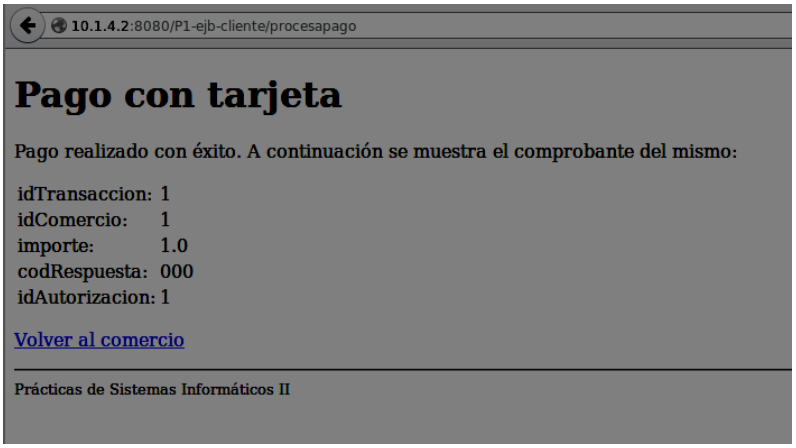
```
$ scp dist/clientjms/P1-jms-clientjms.jar si2@10.X.Y.1:/tmp
```

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>
```

Vamos a proceder con la ejecución del ejercicio:

1. Replegamos todas las aplicaciones previas.
2. Compilamos y desplegamos la parte C de esta práctica.
3. Hacemos un pago para poder luego continuar.



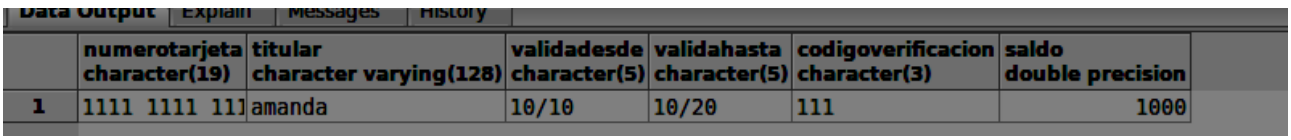
4. Compilamos y desplegamos la versión P1-jms.
5. Anulamos el pago que habiamos hecho en el punto 3.

```
e250858@15-21-67-27:~/Downloads/P1-jms$ /usr/local/glassfish-4.1.1/glassfish/bin
 1
^C
e250858@15-21-67-27:~/Downloads/P1-jms$ /usr/local/glassfish-4.1.1/glassfish/bin
/appclient -targetserver 10.1.4.1 -client dist/clientjms/P1-jms-clientjms.jar 1
mar 13, 2017 6:09:25 PM org.hibernate.validator.internal.util.Version <clinit>
INFO: HV000001: Hibernate Validator 5.1.2.Final
mar 13, 2017 6:09:25 PM com.sun.messaging.jms.ra.ResourceAdapter start
INFORMACIÓN: MQJMSRA_RA1101: GlassFish MQ JMS Resource Adapter: Version:  5.1.1
 (Build 2-c) Compile:  March 17 2015 1045
mar 13, 2017 6:09:25 PM com.sun.messaging.jms.ra.ResourceAdapter start
INFORMACIÓN: MQJMSRA_RA1101: GlassFish MQ JMS Resource Adapter starting: broker
is REMOTE, connection mode is TCP
mar 13, 2017 6:09:25 PM com.sun.messaging.jms.ra.ResourceAdapter start
INFORMACIÓN: MQJMSRA_RA1101: GlassFish MQ JMS Resource Adapter Started:REMOTE
Mesaje: 1
enviado

e250858@15-21-67-27:~/Downloads/P1-jms$
```

6. Comprobamos que el pago se ha anulado y se ha modificado su codRespuesta.

7. Y que le han devuelto a su propietario el dinero en su tarjeta.



Data Output | Explain | Messages | History

	numerotarjeta character(19)	titular character varying(128)	validadesde character(5)	validahasta character(5)	codigoverificacion character(3)	saldo double precision
1	1111 1111 11]	amanda	10/10	10/20	111	1000

8. Comprobamos que no se haya quedado ningún mensaje en la cola de mensajes

```
e250858@15-21-67-27:~/Downloads/P1-jms$ /usr/local/glassfish-4.1.1/glassfish/bin
/appclient -targetserver 10.1.4.1 -client dist/clientjms/P1-jms-clientjms.jar -b
rowse
mar 13, 2017 6:10:26 PM org.hibernate.validator.internal.util.Version <clinit>
INFO: HV000001: Hibernate Validator 5.1.2.Final
mar 13, 2017 6:10:27 PM com.sun.messaging.jms.ra.ResourceAdapter start
INFORMACIÓN: MQJMSRA_RA1101: GlassFish MQ JMS Resource Adapter: Version:  5.1.1
 (Build 2-c) Compile:  March 17 2015 1045
mar 13, 2017 6:10:27 PM com.sun.messaging.jms.ra.ResourceAdapter start
INFORMACIÓN: MQJMSRA_RA1101: GlassFish MQ JMS Resource Adapter starting: broker
is REMOTE, connection mode is TCP
mar 13, 2017 6:10:27 PM com.sun.messaging.jms.ra.ResourceAdapter start
INFORMACIÓN: MQJMSRA_RA1101: GlassFish MQ JMS Resource Adapter Started:REMOTE
Cola de mensajes vacía!
e250858@15-21-67-27:~/Downloads/P1-jms$
```

También probaremos que se puede ejecutar desde la máquina virtual

1. Realizamos el pago con identificador 3 para probar el código

Pago con tarjeta

Lista de pagos del comercio 1

idTransaccion	Importe	codRespuesta	idAutorizacion
1	1.0	999	1
2	1.0	999	2
3	1.0	000	3

Volver al comercio

Prácticas de Sistemas Informáticos II

2. Ejecutamos el programa desde la máquina virtuales

```
si2@si2srv01:~$ /opt/glassfish4/glassfish/bin/appclient -targetserver 10.1.4.1 -client /tmp/P1-jms-clientjms.jar 3
Mar 13, 2017 10:24:32 AM org.hibernate.validator.internal.util.Version <clinit>
INFO: HV000001: Hibernate Validator 5.0.0.Final
Mar 13, 2017 10:24:32 AM com.sun.messaging.jms.ra.ResourceAdapter start
INFO: MQJMSRA_RA1101: GlassFish MQ JMS Resource Adapter: Version:  5.0  (Build 14-e) Compile:  April 12 2013 0104
Mar 13, 2017 10:24:32 AM com.sun.messaging.jms.ra.ResourceAdapter start
INFO: MQJMSRA_RA1101: GlassFish MQ JMS Resource Adapter starting: broker is REMOTE, connection mode is TCP
Mar 13, 2017 10:24:32 AM com.sun.messaging.jms.ra.ResourceAdapter start
INFO: MQJMSRA_RA1101: GlassFish MQ JMS Resource Adapter Started:REMOTE
Mesaje: 3
enviado

si2@si2srv01:~$
```

3. Comprobamos que se anula el pago

Pago con tarjeta

Lista de pagos del comercio 1

idTransaccion	Importe	codRespuesta	idAutorizacion
1	1.0	999	1
2	1.0	999	2
3	1.0	999	3

Volver al comercio

Prácticas de Sistemas Informáticos II