PRÁCTICA 1B

Sistemas Informáticos 2

Adrián San Felipe y Luis Miguel Durán

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? Comparar esta interfaz con el fichero de configuración del web service implementado en la práctica P1A.

```java
package ssii2.visa;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import javax.ejb.Local;

@Local
public interface VisaDAOLocal {
    public boolean compruebaTarjeta(TarjetaBean tarjeta);
    public PagoBean realizaPago(PagoBean pago);
    public PagoBean[] getPagos(String idComercio);
    public int delPagos(String idComercio);
    public boolean isDebug();
    public boolean isPrepared();
    public void setPrepared(boolean prepared);
    public void setDebug(boolean debug);
    public int getDirectConnectionCount();
    public int getDSNConnectionCount();
    public boolean isDirectConnection();
    public void setDirectConnection(boolean directConnection);
}
```

En el archivo VisaDAOLocal.java podemos observar que se importa la librería "*javax.ejb.Local*". Esta sirve para designar como Local una interfaz y definir tanto los métodos como el Bean. @Local se utiliza para especificar que dicha interfaz es local, aunque esto no es necesario que se especifique si la interfaz por defecto ya es local.

El fichero de configuración del WS implementado en la P1A se encuentra en lenguaje xml.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app version="2.5" 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/web-app_2_5.xsd">
<!-- Descripción general de la aplicación web-->
    <display-name>Sistemas Informáticos II - Práctica 1</display-name>
    <description>Esta segunda versión de la práctica 1 implementa la funcionalidad de pago con VISA</description>

    <!-- Inicialización de parámetros de contexto que definen constantes
       de tipo String que se usan en la aplicación, lo que se puede
       particularizar por el administrador del sistema que instala la
       aplicación. Los valores que se asignen a estos parámetros se
       pueden obtener en un servlet o página JSP llamando a:

        String value =
              getServletContext().getInitParameter("nombre");

       donde "nombre se corresponde con el elemento <param-name>
       de uno de estos parámetros de inicialización.

       Se puede definir cualquier número de parámetros de inicialización,
       incluyendo cero parámetros.
    -->

    <context-param>
        <param-name>webmaster</param-name>
        <param-value>http://10.5.9.1:8080/P1-ws-ws/VisaDAOWSService</param-value>
    </context-param>


    <!-- Definición de cada uno de los Servlet que componen la aplicación
       incluyendo parámetros de inicialización de cada servlet.

       Los parámetros de inicialización de servlets se pueden obtener en
       un servlet o página JSP llamando a:

             String value =
               getServletConfig().getInitParameter("nombre");

       donde "nombre se corresponde con el elemento <param-name>
       de uno de estos parámetros de inicialización.

        Se puede definir cualquier número de servlets, incluso cero
    -->
    <!-- Definición de filtros que dispone la aplicación. Los filtro se definen
    de manera similar a los servlet. Requieren un nombre y la clase que lo
    implementa.

    Los parámetros de inicialización de un filtro se pueden obtener en llamando a:
```

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 el método getPagos() a la interfaz definida en VisaDAOLocal
- Incluye en la memoria cada fragmento de código donde se han ido añadiendo las modificaciones solicitadas.

Añadimos el import necesario al comienzo del fichero:

```
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import javax.jws.WebMethod;
import javax.jws.WebParam;
import javax.jws.WebService;
import javax.ejb.Stateless;
```

Eliminamos el constructor que estaba anteriormente y lo modificamos por el que se nos indica en el enunciado:

```
@Stateless(mappedName="VisaDAOBean")
public class VisaDAOBean extends DBTester implements VisaDAOLocal {
```

La imagen siguiente es el constructor por defecto de la clase. Lo eliminamos:

```
    /**
     * Constructor de la clase
     */
    public VisaDAO() {
        return;
    }
```

Cambiamos el método getPagos() a como se encuentra en VisaDAOLocal:

```
    /**
     * Buscar los pagos asociados a un comercio
     * @param idComercio
     * @return
     */
    public PagoBean[] getPagos(String idComercio) {
```

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

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.

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()).

Se han añadido los import indicados en el enunciado y borrado los que han dejado de existir en ProcesaPago.java, DelPagos.java y GetPagos.java :

```
import java.io.IOException;
import java.net.InetAddress;
import java.net.NetworkInterface;
import java.net.SocketException;
import java.net.UnknownHostException;
import java.util.Collections;
import java.util.Enumeration;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import ssii2.visa.*;
import javax.xml.ws.WebServiceRef;
import javax.xml.ws.*;
import javax.ejb.EJB; // AÑADIDO EJ2
import ssii2.visa.VisaDAOLocal; // AÑADIDO EJ2
```

Añadimos como atributo de la clase el objeto proxy que permite acceder al EJB local, con su correspondiente anotación que lo declara como tal en ProcesaPago.java, DelPagos.java y GetPagos.java:

```
/**
 * AÑADIDO EL OBJETO PROXY PARA ACCEDER AL EJB LOCAL
 */
@EJB(name="VisaDAOBean", beanInterface=VisaDAOLocal.class)
private VisaDAOLocal dao;
```

Eliminamos las siguientes líneas del archivo ProcesaPago.java, DelPagos.java y GetPagos.java

```
VisaDAOWSService service = new VisaDAOWSService();
VisaDAOWS dao = service. getVisaDAOWSPort ();
BindingProvider bp = (BindingProvider) dao;
bp.getRequestContext().put(BindingProvider.ENDPOINT_ADDRESS_PROPERTY, getServletContext().getInitParameter("webmaster"));
```

En específico en GetPagos.java, cambiamos esta línea:

```
/* Petici&oacute;n de los pagos para el comercio */
PagoBean[] pagos = (PagoBean[] ) dao.getPagos(idComercio).toArray(new PagoBean[dao.getPagos(idComercio).size()]);
```

Por esta, para que se adapte a la nueva interfaz:

```
/* Petici&oacute;n de los pagos para el comercio */
PagoBean[] pagos =  dao.getPagos(idComercio);
```

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.

El contenido del archivo application.xml es el siguiente:

```
<?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/j
  <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>
```

<application>: funciona como esquema de la aplicación, indicando su versión.

<ejb>: indica la ruta donde se escuentra ejb-jar

<web>: encontramos tanto la etiqueta <web-uri> como <context-root>, que son la URI y el contexto raíz del módulo de la aplicación web, ya que podemos observar que estas tres etiquetas se encuentran dentro de <module>

Contenido del archivo .war:

```
eps@labvirteps:~/Escritorio/P1-ejb$ jar -tvf dist/client/P1-ejb-cliente.war
     0 Wed Mar 17 14:29:18 CET 2021 META-INF/
   125 Wed Mar 17 14:29:16 CET 2021 META-INF/MANIFEST.MF
     0 Wed Mar 17 14:29:18 CET 2021 WEB-INF/
     0 Wed Mar 17 14:29:04 CET 2021 WEB-INF/classes/
     0 Wed Mar 17 14:29:04 CET 2021 WEB-INF/classes/ssii2/
     0 Wed Mar 17 14:29:04 CET 2021 WEB-INF/classes/ssii2/controlador/
     0 Wed Mar 17 14:29:04 CET 2021 WEB-INF/classes/ssii2/filtros/
     0 Wed Mar 17 14:29:04 CET 2021 WEB-INF/classes/ssii2/visa/
     0 Wed Mar 17 14:29:04 CET 2021 WEB-INF/classes/ssii2/visa/error/
     0 Wed Mar 17 13:23:58 CET 2021 WEB-INF/lib/
     0 Wed Mar 17 14:29:18 CET 2021 error/
  2844 Wed Mar 17 14:29:04 CET 2021 WEB-INF/classes/ssii2/controlador/ComienzaPago.class
  1513 Wed Mar 17 14:29:04 CET 2021 WEB-INF/classes/ssii2/controlador/DelPagos.class
  1365 Wed Mar 17 14:29:04 CET 2021 WEB-INF/classes/ssii2/controlador/GetPagos.class
  4915 Wed Mar 17 14:29:04 CET 2021 WEB-INF/classes/ssii2/controlador/ProcesaPago.class
  1894 Wed Mar 17 14:29:04 CET 2021 WEB-INF/classes/ssii2/controlador/ServletRaiz.class
  2608 Wed Mar 17 14:29:04 CET 2021 WEB-INF/classes/ssii2/filtros/CompruebaSesion.class
  3170 Wed Mar 17 14:29:04 CET 2021 WEB-INF/classes/ssii2/visa/ValidadorTarjeta.class
   616 Wed Mar 17 14:29:04 CET 2021 WEB-INF/classes/ssii2/visa/error/ErrorVisa.class
   198 Wed Mar 17 14:29:04 CET 2021 WEB-INF/classes/ssii2/visa/error/ErrorVisaCVV.class
   209 Wed Mar 17 14:29:04 CET 2021 WEB-INF/classes/ssii2/visa/error/ErrorVisaFechaCaducidad.class
   207 Wed Mar 17 14:29:04 CET 2021 WEB-INF/classes/ssii2/visa/error/ErrorVisaFechaEmision.class
   201 Wed Mar 17 14:29:04 CET 2021 WEB-INF/classes/ssii2/visa/error/ErrorVisaNumero.class
   202 Wed Mar 17 14:29:04 CET 2021 WEB-INF/classes/ssii2/visa/error/ErrorVisaTitular.class
  6025 Wed Mar 17 14:29:18 CET 2021 WEB-INF/web.xml
   455 Wed Mar 17 14:29:18 CET 2021 borradoerror.jsp
   501 Wed Mar 17 14:29:18 CET 2021 borradook.jsp
   509 Wed Mar 17 14:29:18 CET 2021 cabecera.jsp
   283 Wed Mar 17 14:29:18 CET 2021 error/muestraerror.jsp
  2729 Wed Mar 17 14:29:18 CET 2021 formdatosvisa.jsp
  1257 Wed Mar 17 14:29:18 CET 2021 listapagos.jsp
  1178 Wed Mar 17 14:29:18 CET 2021 pago.html
  1142 Wed Mar 17 14:29:18 CET 2021 pagoexito.jsp
   104 Wed Mar 17 14:29:18 CET 2021 pie.html
  5011 Wed Mar 17 14:29:18 CET 2021 testbd.jsp
```

Contenido del archivo .ear:

```
eps@labvirteps:~/Escritorio/P1-ejb$ jar -tvf dist/P1-ejb.ear
     0 Wed Mar 17 14:51:10 CET 2021 META-INF/
   125 Wed Mar 17 14:51:08 CET 2021 META-INF/MANIFEST.MF
   508 Sat Feb 11 23:33:00 CET 2012 META-INF/application.xml
 20951 Wed Mar 17 14:29:18 CET 2021 P1-ejb-cliente.war
  6924 Wed Mar 17 13:42:10 CET 2021 P1-ejb.jar
```

Contenido del archivo .jar:

```
eps@labvirteps:~/Escritorio/P1-ejb$ jar -tvf dist/server/P1-ejb.jar
     0 Wed Mar 17 13:42:12 CET 2021 META-INF/
   125 Wed Mar 17 13:42:10 CET 2021 META-INF/MANIFEST.MF
     0 Wed Mar 17 13:36:54 CET 2021 ssii2/
     0 Wed Mar 17 13:36:54 CET 2021 ssii2/visa/
   255 Wed Mar 17 13:42:10 CET 2021 META-INF/sun-ejb-jar.xml
  1719 Wed Mar 17 13:36:54 CET 2021 ssii2/visa/DBTester.class
  1464 Wed Mar 17 13:36:54 CET 2021 ssii2/visa/PagoBean.class
   856 Wed Mar 17 13:36:54 CET 2021 ssii2/visa/TarjetaBean.class
  7089 Wed Mar 17 13:36:54 CET 2021 ssii2/visa/VisaDAOBean.class
   593 Wed Mar 17 13:36:54 CET 2021 ssii2/visa/VisaDAOLocal.class
```

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
```

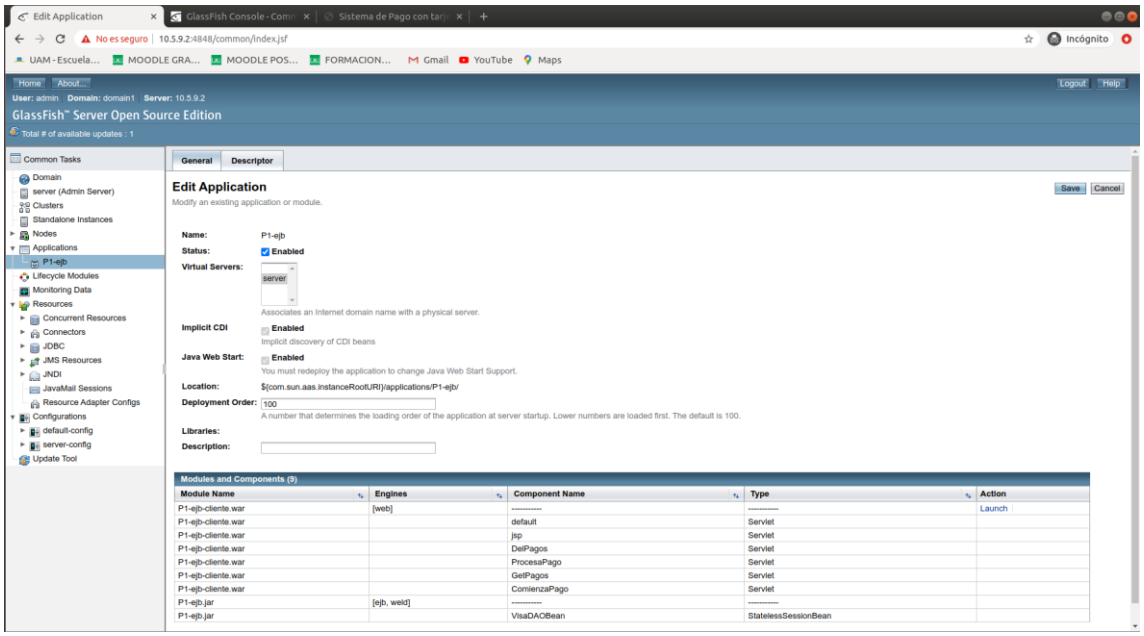
En postgresql.properties debemos cambiar el host de la base de datos a la ip 10.5.9.1 y db.client.host a la ip 10.5.9.2

```
# Propiedades de la BD postgresql

# Parametros propios de postgresql
db.name=visa
db.user=alumnodb
db.password=****
db.port=5432
db.host=10.5.9.1
# Recursos y pools asociados
db.pool.name=VisaPool
db.jdbc.resource.name=jdbc/VisaDB
db.url=jdbc:postgresql://${db.host}:${db.port}/${db.name}
db.client.host=10.5.9.2
db.client.port=4848

db.delimiter=;
db.driver=org.postgresql.Driver
db.datasource=org.postgresql.ds.PGConnectionPoolDataSource
db.vendorname=SQL92

# Herramientas
db.createdb=/usr/bin/createdb
db.dropdb=/usr/bin/dropdb

# Scripts de creacion / borrado
db.create.src=./sql/create.sql
db.insert.src=./sql/insert.sql
db.delete.src=./sql/drop.sql
```

En build.properties tenemos que poner las dos variables que se nos indican con la ip 10.5.9.2, ya que ahora en la misma máquina se encuentran el cliente y el servidor:

```
nombre=P1-ejb
build=${basedir}/build
build.client=${build}/client
build.server=${build}/server
dist=${basedir}/dist
dist.client=${dist}/client
dist.server=${dist}/server
src=${basedir}/src
src.client=${src}/client
src.server=${src}/server
web=${basedir}/web
conf=${basedir}/conf
conf.server=${conf}/server
conf.application=${conf}/application
paquete=ssii2
war=${nombre}-cliente.war
jar=${nombre}.jar
ear=${nombre}.ear
asadmin=${as.home}/bin/asadmin
as.home=${env.J2EE_HOME}
as.lib=${as.home}/lib
as.user=admin
as.host.client=10.5.9.2
as.host.server=10.5.9.2
as.port=4848
as.passwordfile=${basedir}/passwordfile
as.target=server
```

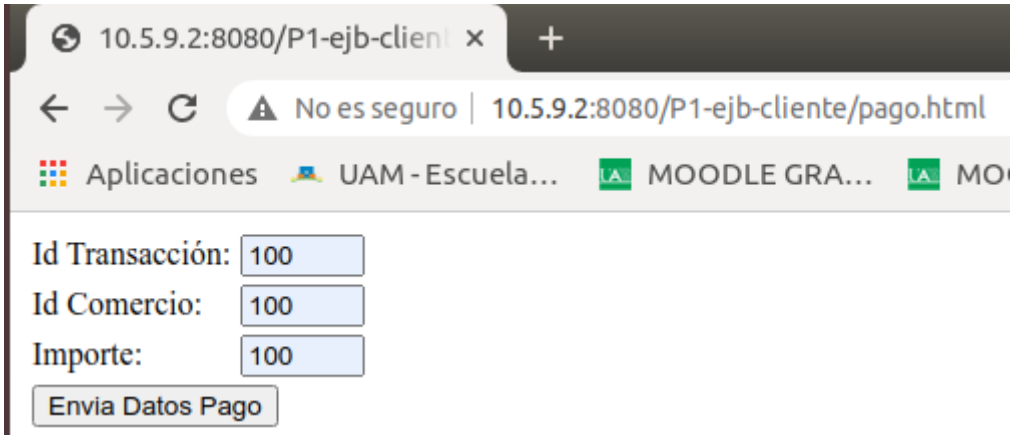
```
eps@labvirteps:~/Escritorio/P1-ejb$ ant desplegar
Buildfile: /home/eps/Escritorio/P1-ejb/build.xml

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

BUILD SUCCESSFUL
Total time: 25 seconds
```

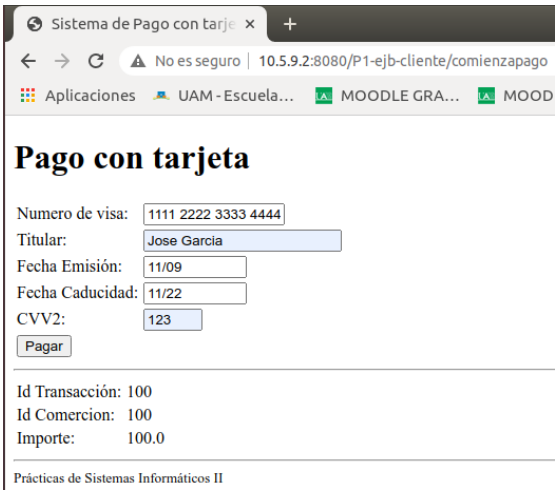


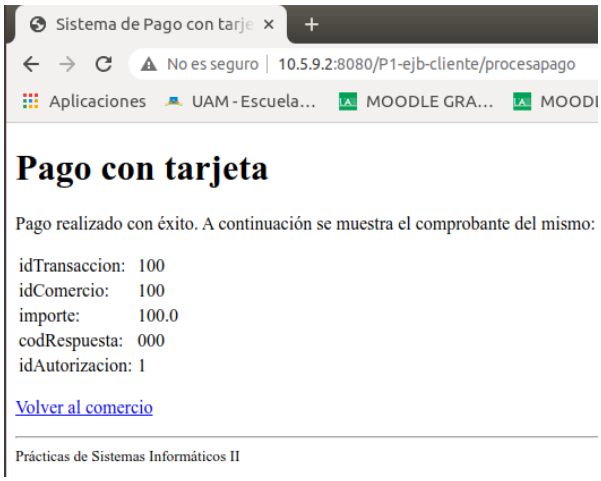
Ejercicio 4: Comprobar el correcto funcionamiento de la aplicación mediante llamadas directas a través de las páginas *pago.html* y *testbd.jsp (sin directconnection)*. Realice un pago. Lístelo. Elimínelo. Téngase en cuenta que la aplicación se habrá desplegado bajo la ruta /P1-ejb-cliente.

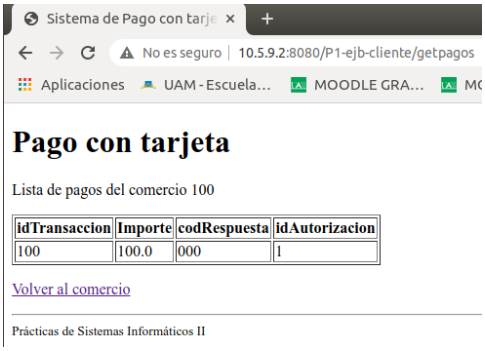
Incluya en la memoria de prácticas todos los pasos necesarios para resolver este ejercicio así como las evidencias obtenidas. Se pueden incluir por ejemplo capturas de pantalla.

Pruebas en pago.html:

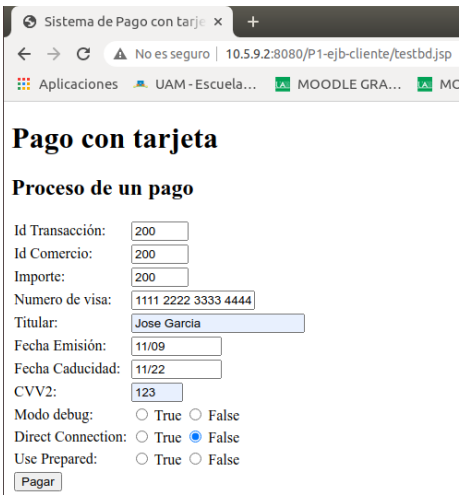


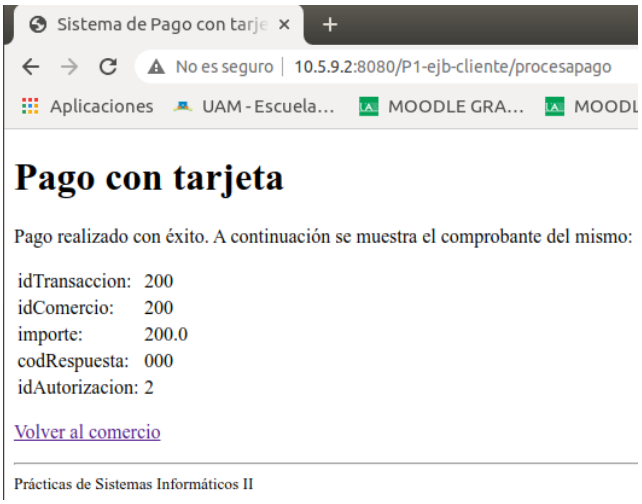






Pruebas en testbd.jsp:





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.

Añadimos import en PagoBean.java:

```
package ssii2.visa;
import java.io.Serializable;
```

Añadimos que sea serializable:

```
public class PagoBean implements Serializable {
```

Añadimos import en TarjetaBean.java:

```
package ssii2.visa;
import java.io.Serializable;
```

Añadimos que sea serializable:

```
public class TarjetaBean implements Serializable {
```

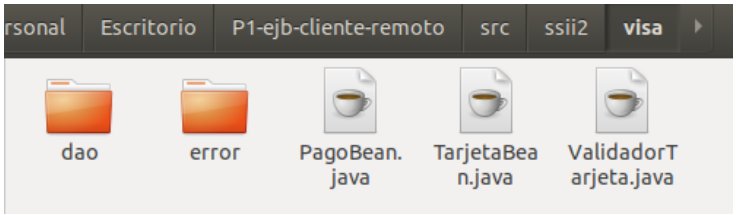
Ejercicio 6: Realizar los cambios comentados en la aplicación P1-base para convertirla en P1-ejb-cliente-remoto. Compilar, empaquetar y desplegar de nuevo la aplicación en otra máquina virtual distinta a la de la aplicación servidor, es decir, esta aplicación cliente estará desplegada en la MV del PC1 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.

Cambiamos el nombre del build.properties:

build.properties

```
# Propiedades de despliegue de
nombre=P1-ejb-cliente-remoto
build=${basedir}/build
```

Eliminamos la carpeta “dao”:



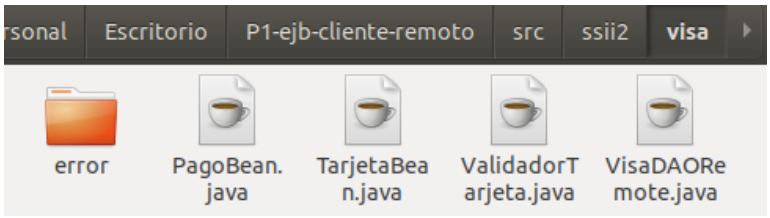
Cambios en PagoBean.java:

```java
import java.io.Serializable;

/**
 *
 * @author jaime
 */
public class PagoBean implements Serializable{
```

Cambios en TarjetaBean.java:

```java
import java.io.Serializable;

public class TarjetaBean implements Serializable{
```

Copiamos VisaDAORemote.java a la ruta src/ssii2/visa:



Añadidos import en ProcesaPago.java, GetPagos.java y DelPagos.java:

```java
import javax.ejb.EJB;
import ssii2.visa.VisaDAORemote;
```

Añadida anotación en ProcesaPago.java, GetPagos.java y DelPagos.java:

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

Archivo glassfish-web.xml:

glassfish-web.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE glassfish-web-app PUBLIC "-//GlassFish.org//DTD GlassFish Application Server 3.1 Servlet 3.0//EN" "http://glassfish.org/dtds/glassfish-web-app_3_0-1.dtd">
<glassfish-web-app>
<ejb-ref>
 <ejb-ref-name>VisaDAOBean</ejb-ref-name>
 <jndi-name>corbaname:iiop:10.5.9.2:3700#java:global/P1-ejb/P1-ejb/VisaDAOBean!ssii2.visa.VisaDAORemote</jndi-name>
</ejb-ref>
</glassfish-web-app>
```

Archivos build.properties y postgresql.properties:

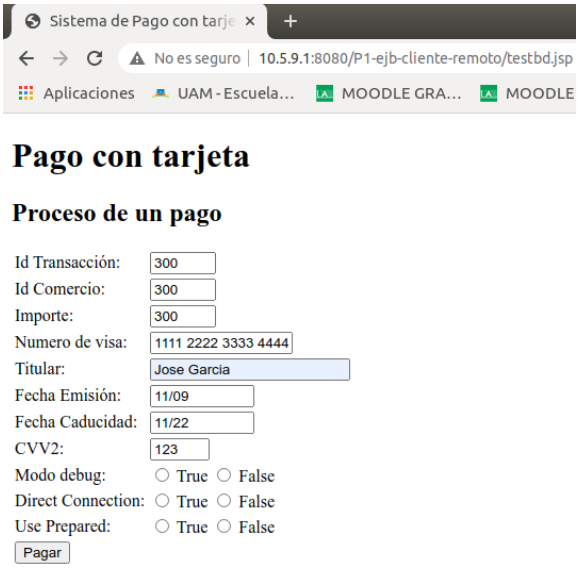
build.properties

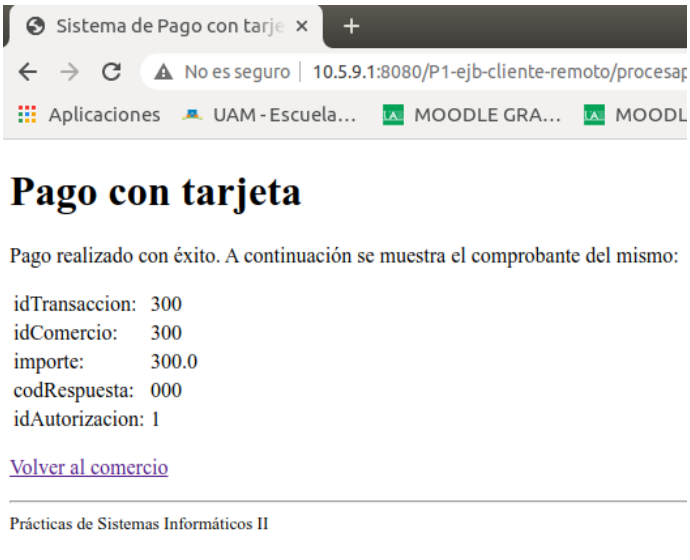
```
# Propiedades de despliegue de aplicacion de Visa
nombre=P1-ejb-cliente-remoto
build=${basedir}/build


dist=${basedir}/dist


src=${basedir}/src


web=${basedir}/web


paquete=ssii2
war=${nombre}.war



asadmin=${as.home}/bin/asadmin
as.home=${env.J2EE_HOME}
as.lib=${as.home}/lib
as.user=admin
as.host=10.5.9.1

as.port=4848
as.passwordfile=${basedir}/passwordfile
as.target=server
```

postgresql.properties

```
# Propiedades de la BD postgresql

# Parametros propios de postgresql
db.name=visa
db.user=alumnodb
db.password=****
db.port=5432
db.host=10.5.9.1
# Recursos y pools asociados
db.pool.name=VisaPool
db.jdbc.resource.name=jdbc/VisaDB
db.url=jdbc:postgresql://${db.host}:${db.port}/${db.name}
db.client.host=10.5.9.2
db.client.port=4848

db.delimiter=;
db.driver=org.postgresql.Driver
db.datasource=org.postgresql.ds.PGConnectionPoolDataSource
db.vendorname=SQL92

# Herramientas
db.createdb=/usr/bin/createdb
db.dropdb=/usr/bin/dropdb

# Scripts de creacion / borrado
db.create.src=./sql/create.sql
db.insert.src=./sql/insert.sql
db.delete.src=./sql/drop.sql
```

Prueba de pago:





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 actualizar 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.

Añadir el atributo saldo y sus métodos de acceso a TarjetaBean.java:

```
private double saldo;
```

```
public double getSaldo() {
return saldo;
}

public void setSaldo(double saldo) {
this.saldo = saldo;
}
```

Añadidos en archivo VisaDAOBean.java:

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

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

```
errorLog(SELECT_SALDO_QRY);
pstmt = con.prepareStatement(SELECT_SALDO_QRY);
pstmt.setString(1, pago.getTarjeta().getNumero());
ResultSet rs_saldo = pstmt.executeQuery();

if (rs_saldo.next()) {
    if (rs_saldo.getDouble("saldo") < pago.getImporte()) {
        pago.setIdAutorizacion("null");
        return null;
    }
} else {
    throw new EJBException("Error en la tarjeta.");
}


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

```
        throw new EJBException("Pago erróneo: id de transacción duplicada.");
    }
 } else {
     throw new EJBException("Pago erróneo.");
```

Añadidos en archivo ProcesaPago.java:

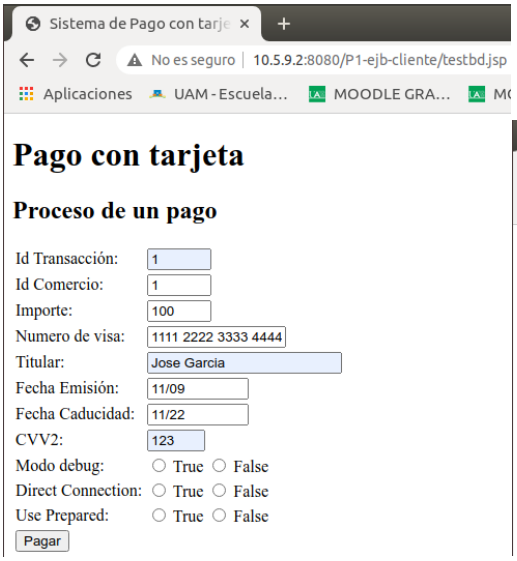
```
import javax.ejb.EJBException;
```

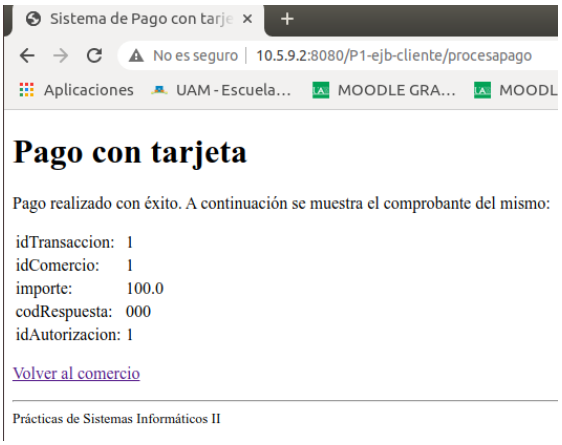
```
try{
    pago = dao.realizaPago(pago);
}catch (EJBException e){
    if(sesion != null)
        sesion.invalidate();
    enviaError(new Exception("Pago incorrecto."), request, response);
    return;
}
```

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.

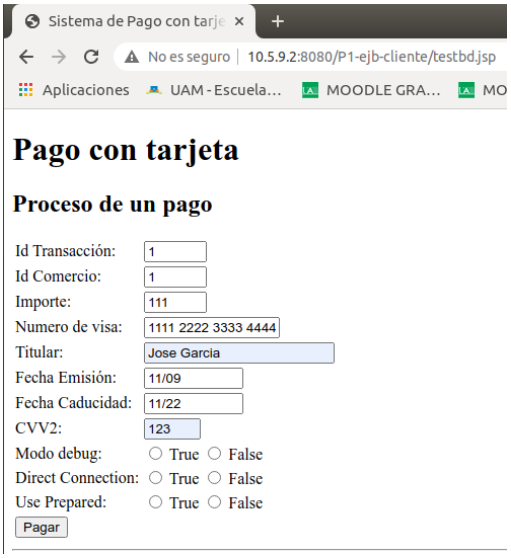
Pruebas de pago con saldo:

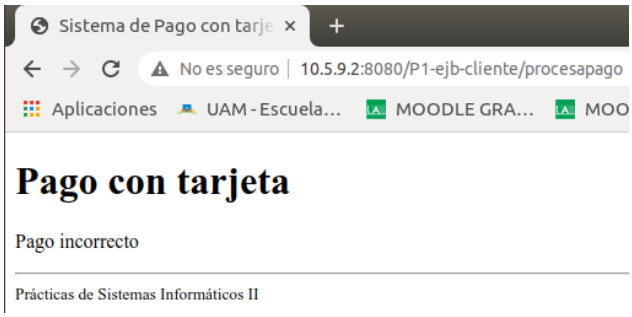




	numerotarjeta	titular	validadesde	validahasta	codigoverificacion	saldo
1	1111 2222 3333 4444	Jose Garcia	11/09	11/22	123	900

Operación de pago incorrecto:



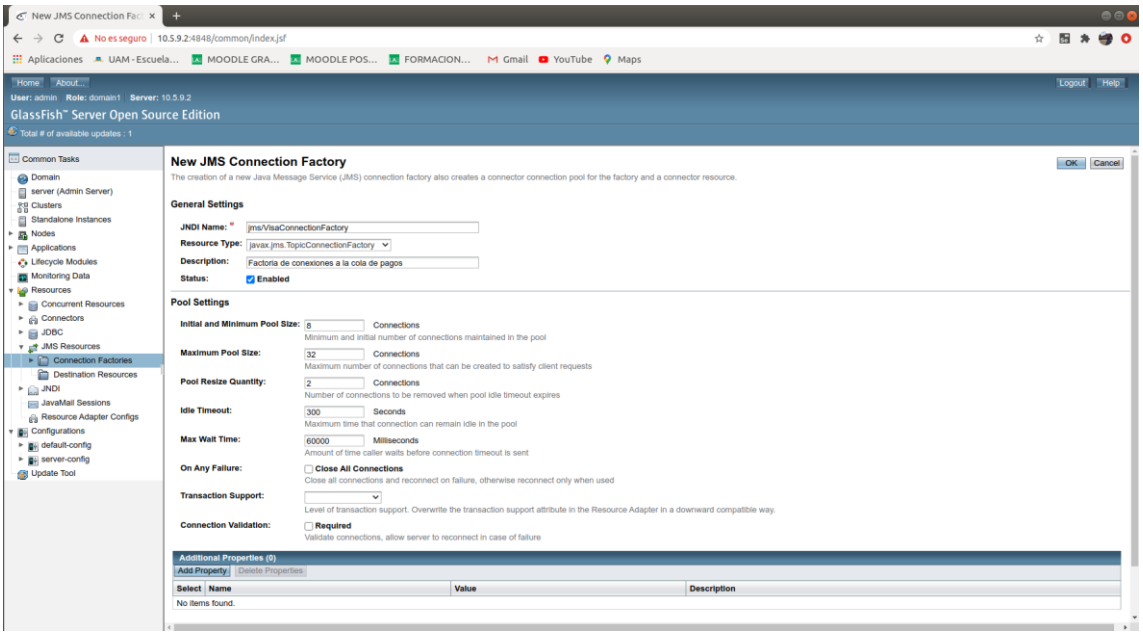


86	1111 2222 3333 4444	Jose Garcia	11/09	11/22	123	900

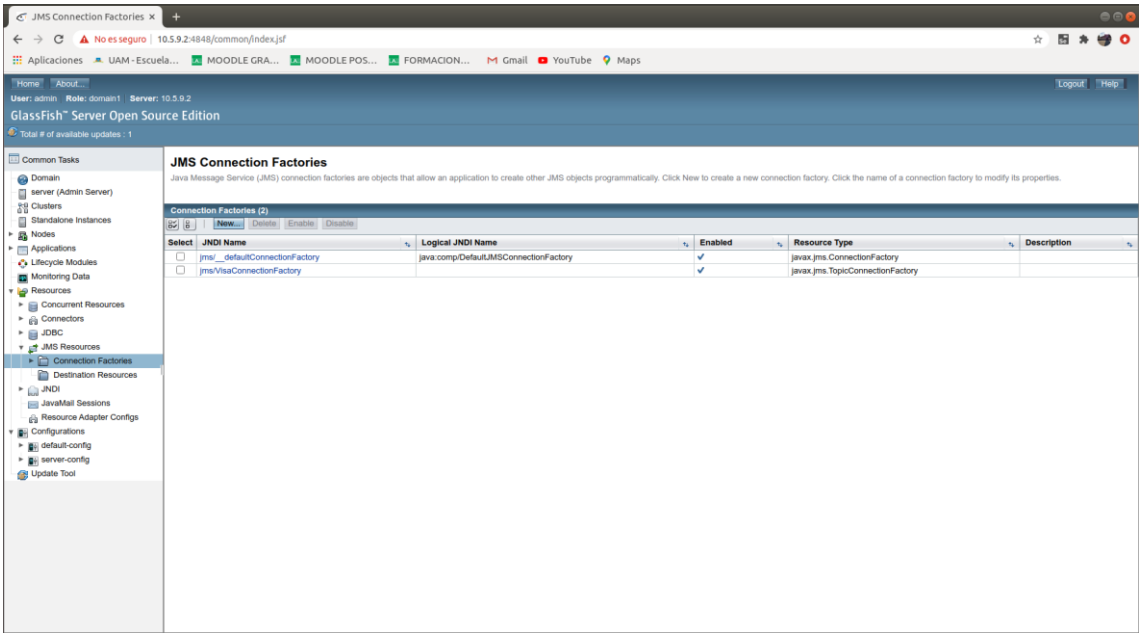
Ejercicio 9: En la máquina virtual donde se encuentra el servidor de aplicaciones (10.X.Y.Z2), declare manualmente la factoría de conexiones empleando la consola de administración, tal y como se adjunta en la Figura 4.

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

Creamos la factoría de conexiones como se nos indica en el enunciado:

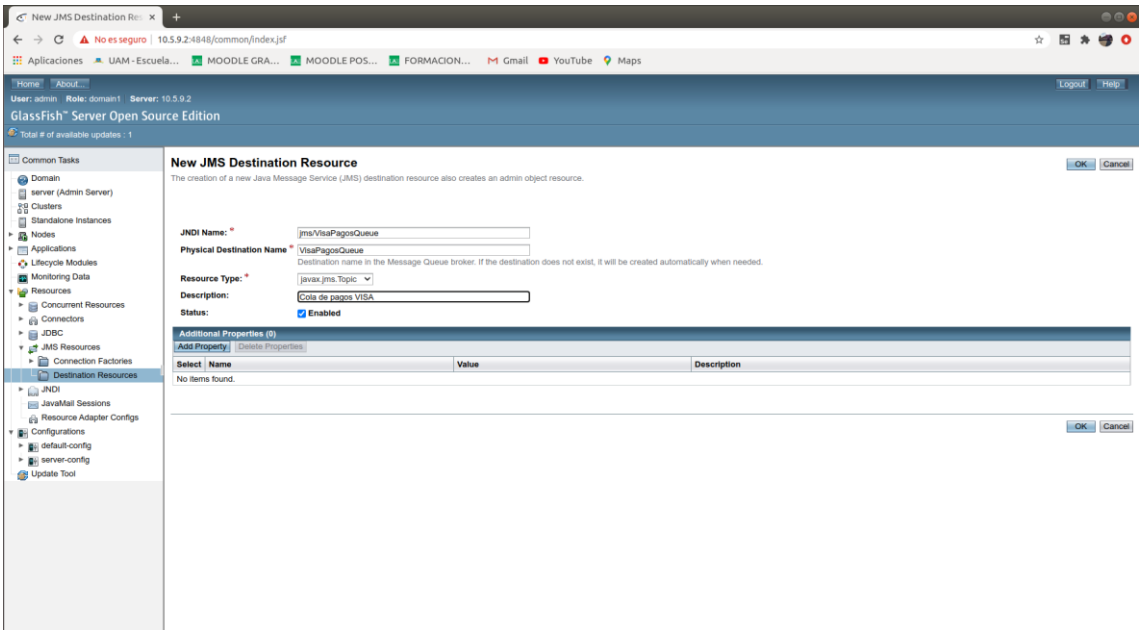


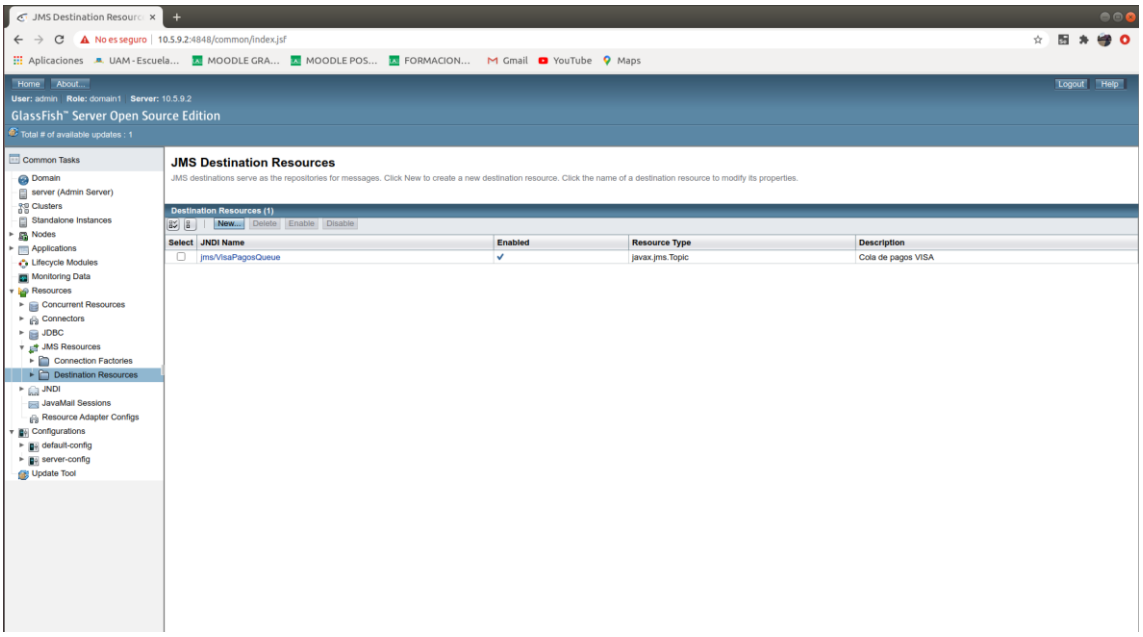
Comprobamos que la factoría de conexiones se ha creado correctamente.



Ejercicio 10: En la máquina virtual donde se encuentra el servidor de aplicaciones (10.X.Y.Z2), 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.





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 y cierre de conexiones.

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

Modificado el fichero sun-ejb-jar.xml:

sun-ejb-jar.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!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">
<sun-ejb-jar>
  <enterprise-beans>
    <ejb>
      <ejb-name>VisaCancelacionJMSBean</ejb-name>
        <mdb-connection-factory>
          <jndi-name>jms/VisaConnectionFactory</jndi-name>
        </mdb-connection-factory>
    </ejb>
  </enterprise-beans>
</sun-ejb-jar>
```

Nuevas consultas SQL:

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

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

Método onMessage():

```
public void onMessage(Message inMessage) {
    TextMessage msg = null;
    Connection con = null;
    PreparedStatement pstmt = null;
    try {
        if (inMessage instanceof TextMessage) {
          msg = (TextMessage) inMessage;
          logger.info("MESSAGE BEAN: Message received: " + msg.getText());
          int idAutorizacion = Integer.parseInt(msg.getText());
          con = getConnection();

          pstmt = con.prepareStatement(UPDATE_CODRESPUESTA_QRY);
          pstmt.setInt(1,idAutorizacion);
          logger.info("UPDATE COD RESPUESTA: " + UPDATE_CODRESPUESTA_QRY);
          pstmt.executeQuery();

          pstmt = con.prepareStatement(UPDATE_SALDO_QRY);
          pstmt.setInt(1,idAutorizacion);
          logger.info("UPDATE SALDO: " + UPDATE_SALDO_QRY);
          pstmt.executeQuery();

        } else {
            logger.warning(
                    "Message of wrong type: "
                    + inMessage.getClass().getName());
        }
    } catch (JMSException e) {
        e.printStackTrace();
        mdc.setRollbackOnly();
    } catch (Throwable te) {
        te.printStackTrace();
    }
}
```

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.

Añadidas anotaciones estáticas:

```
@Resource(mappedName = "jms/VisaConnectionFactory")
private static ConnectionFactory connectionFactory;
@Resource(mappedName = "jms/VisaPagosQueue")
private static Queue queue;
```

Añadidas anotaciones dinámicas:

```
/*************************************************
InitialContext jndi = new InitialContext();
connectionFactory = (ConnectionFactory)jndi.lookup("jms/NombreDeLaConnectionFactory");
queue = (Queue)jndi.lookup("jms/NombreDeLaCola");
*************************************************/
```

Si usamos las anotaciones estáticas, el sistema no tiene transparencia de ubicación ya que se indica donde se encuentra el recurso. Si usamos las anotaciones dinámicas, ofrecen transparencia de ubicación ya que el nombre de los recursos se conocen al ejecutarse el código.

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 9 y 10. 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.

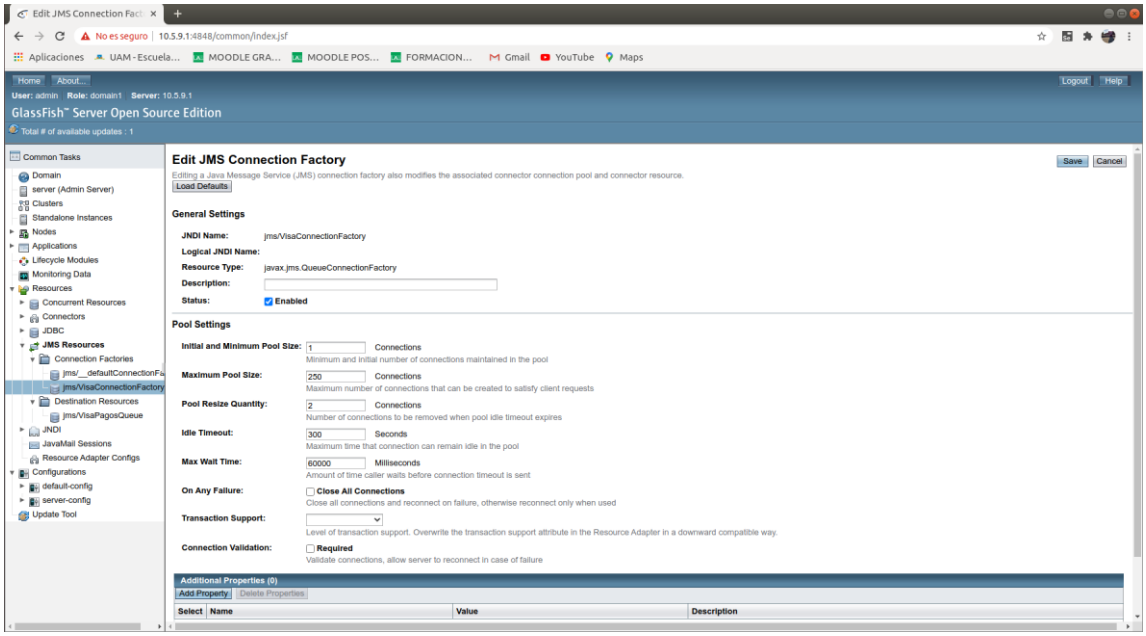
jms.properties

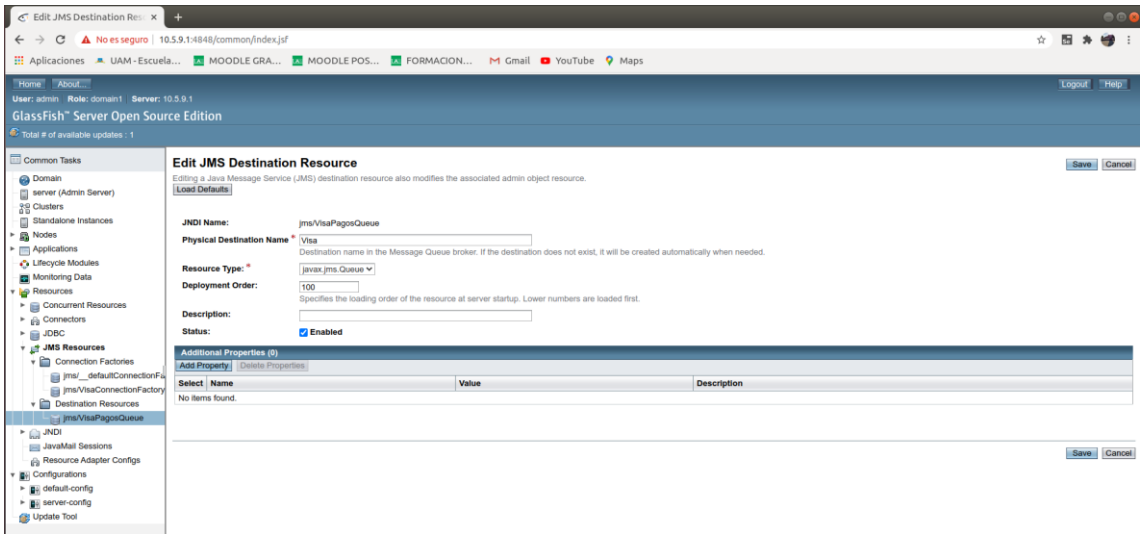
```
as.home=${env.J2EE_HOME}
as.lib=${as.home}/lib
as.user=admin
as.host.client=10.5.9.2
as.host.server=10.5.9.2
as.port=4848
as.passwordfile=${basedir}/passwordfile
as.target=server
jms.factoryname=jms/VisaConnectionFactory
jms.name=jms/VisaPagosQueue
jms.physname=Visa
```

build.properties

```
# Propiedades de despliegue de aplicacion de Visa
nombre=P1-jms
build=${basedir}/build
build.clientjms=${build}/clientjms
build.mdb=${build}/mdb
dist=${basedir}/dist
dist.clientjms=${dist}/clientjms
dist.mdb=${dist}/mdb
src=${basedir}/src
src.clientjms=${src}/clientjms
src.mdb=${src}/mdb
web=${basedir}/web
conf=${basedir}/conf
conf.mdb=${conf}/mdb
paquete=ssii2
clientjms-jar=${nombre}-clientjms.jar
mdb-jar=${nombre}-mdb.jar
asadmin=${as.home}/bin/asadmin
as.home=${env.J2EE_HOME}
as.lib=${as.home}/lib
as.user=admin
as.host.mdb=10.5.9.2
as.port=4848
as.passwordfile=${basedir}/passwordfile
as.target=server
```

```
eps@labvirteps:~/Escritorio/P1-jms$ ant todo
Buildfile: /home/eps/Escritorio/P1-jms/build.xml

todo:

setup-jms:

create-jms-connection-factory:
     [exec] Connector resource jms/VisaConnectionFactory created.
     [exec] Command create-jms-resource executed successfully.

create-jms-resource:
     [exec] Administered object jms/VisaPagosQueue created.
     [exec] Command create-jms-resource executed successfully.

mdb:

montar-jerarquia:
    [mkdir] Created dir: /home/eps/Escritorio/P1-jms/build
    [mkdir] Created dir: /home/eps/Escritorio/P1-jms/build/clientjms
    [mkdir] Created dir: /home/eps/Escritorio/P1-jms/build/mdb
    [mkdir] Created dir: /home/eps/Escritorio/P1-jms/dist
    [mkdir] Created dir: /home/eps/Escritorio/P1-jms/dist/clientjms
    [mkdir] Created dir: /home/eps/Escritorio/P1-jms/dist/mdb

compilar-mdb:
    [javac] Compiling 2 source files to /home/eps/Escritorio/P1-jms/build/mdb

preparar-meta-inf-mdb:
     [copy] Copying 2 files to /home/eps/Escritorio/P1-jms/build/mdb

empaquetar-mdb:
      [jar] Building jar: /home/eps/Escritorio/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/eps/Escritorio/P1-jms/build/clientjms

empaquetar-clientjms:
      [jar] Building jar: /home/eps/Escritorio/P1-jms/dist/clientjms/P1-jms-clientjms.jar

BUILD SUCCESSFUL
Total time: 1 minute 25 seconds
```





El comando con asadmin sería:

asadmin --user admin --passwordfile passwordfile --host 10.5.9.1 --port 4848 create-jms-resource --restype javax.jms.QueueConnectionFactory --enabled=true --property Name=Visa jms/VisaPagosQueue

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 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:

```
/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
- Cancélelo 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.Z1:/tmp
```

Desde la máquina virtual 10.X.Y.Z1:

```
si2@si2srv01:~$  export JAVA_HOME=/usr/lib/jvm/java-8-oracle/
si2@si2srv01:~$      /opt/glassfish4/glassfish/bin/appclient      -targetserver
10.X.Y.Z2 -client /tmp/P1-jms-clientjms.jar <idAutorizacion>
```

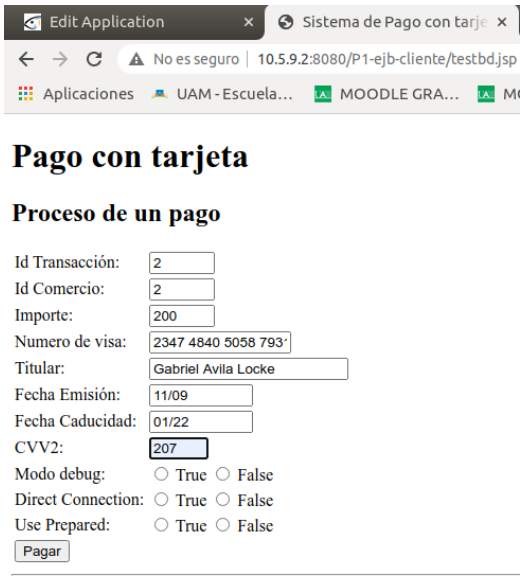
Envío del mensaje de texto:

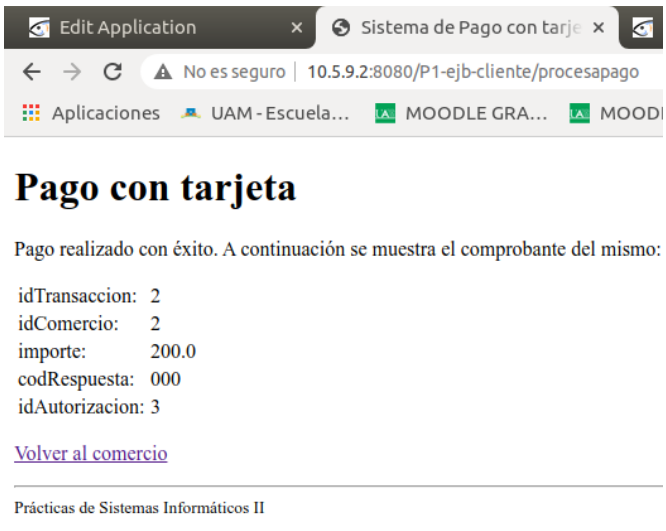
```
// TODO: Enviar argv[0] como mensaje de texto
messageProducer = session.createProducer(queue);
textMessage = session.createTextMessage();
textMessage.setText(args[0]);
messageProducer.send(textMessage);
messageProducer.close();
session.close();
```

Prueba de mensaje a la cola y verificación de que se encuentra a la espera:

```
eps@labvirteps:~/Escritorio/P1-jms$ /opt/glassfish4/glassfish/bin/appclient -targetserver 10.5.9.2  -client dist/clientjms/P1-jms-
clientjms.jar idAutorizacion
mar 19, 2021 12:08:59 AM org.hibernate.validator.internal.util.Version <clinit>
INFO: HV000001: Hibernate Validator 5.1.2.Final
mar 19, 2021 12:08:59 AM 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 19, 2021 12:08:59 AM com.sun.messaging.jms.ra.ResourceAdapter start
INFORMACIÓN: MQJMSRA_RA1101: GlassFish MQ JMS Resource Adapter starting: broker is REMOTE, connection mode is TCP
mar 19, 2021 12:08:59 AM com.sun.messaging.jms.ra.ResourceAdapter start
INFORMACIÓN: MQJMSRA_RA1101: GlassFish MQ JMS Resource Adapter Started:REMOTE
eps@labvirteps:~/Escritorio/P1-jms$ /opt/glassfish4/glassfish/bin/appclient -targetserver 10.5.9.2  -client dist/clientjms/P1-jms-
clientjms.jar -browse
mar 19, 2021 12:10:09 AM org.hibernate.validator.internal.util.Version <clinit>
INFO: HV000001: Hibernate Validator 5.1.2.Final
mar 19, 2021 12:10:10 AM 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 19, 2021 12:10:10 AM com.sun.messaging.jms.ra.ResourceAdapter start
INFORMACIÓN: MQJMSRA_RA1101: GlassFish MQ JMS Resource Adapter starting: broker is REMOTE, connection mode is TCP
mar 19, 2021 12:10:10 AM com.sun.messaging.jms.ra.ResourceAdapter start
INFORMACIÓN: MQJMSRA_RA1101: GlassFish MQ JMS Resource Adapter Started:REMOTE
Mensajes en cola:
idAutorizacion
```

Prueba de pago:

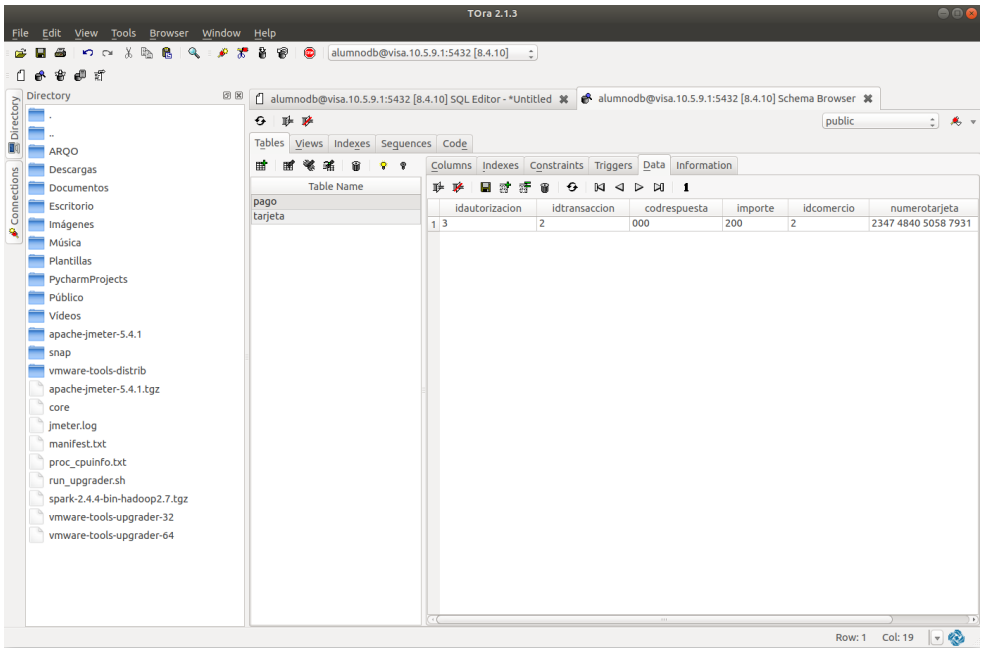




Vemos que ahora la tarjeta tiene 200€ menos:

1001	2347 4840 5058 7931	Gabriel Avila Locke	11/09	01/22	207	800

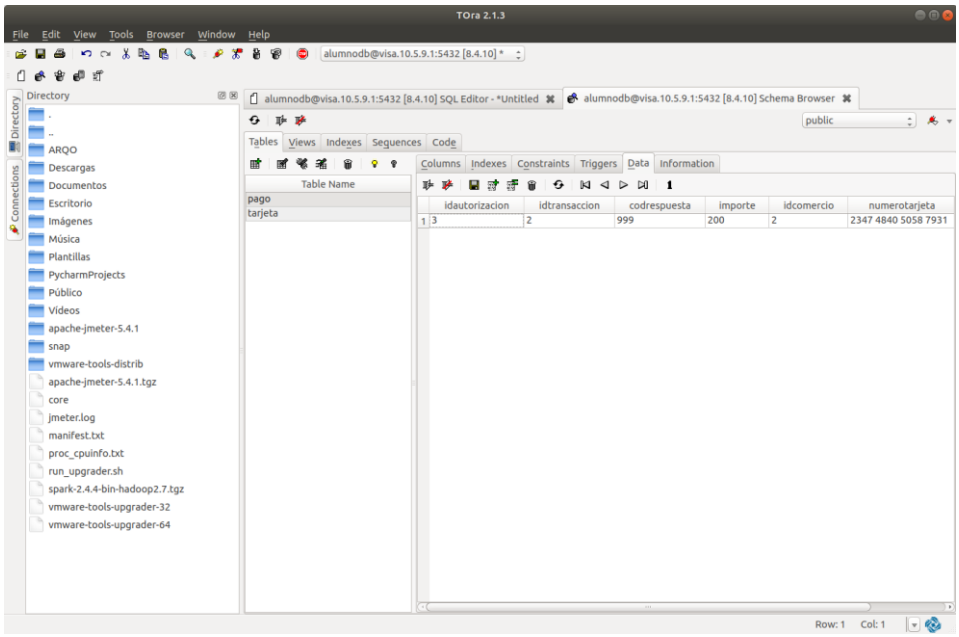
Y que el pago ha quedado reflejado:



Ahora, mandamos un mensaje para eliminar dicho pago:

```
eps@labvirteps:~/Escritorio/P1-jms$ /opt/glassfish4/glassfish/bin/appclient -targetserver 10.5.9.2  -client dist/clientjms/P1-jms-
clientjms.jar 3
mar 19, 2021 12:35:14 AM org.hibernate.validator.internal.util.Version <clinit>
INFO: HV000001: Hibernate Validator 5.1.2.Final
mar 19, 2021 12:35:14 AM 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 19, 2021 12:35:14 AM com.sun.messaging.jms.ra.ResourceAdapter start
INFORMACIÓN: MQJMSRA_RA1101: GlassFish MQ JMS Resource Adapter starting: broker is REMOTE, connection mode is TCP
mar 19, 2021 12:35:14 AM com.sun.messaging.jms.ra.ResourceAdapter start
INFORMACIÓN: MQJMSRA_RA1101: GlassFish MQ JMS Resource Adapter Started:REMOTE
eps@labvirteps:~/Escritorio/P1-jms$ /opt/glassfish4/glassfish/bin/appclient -targetserver 10.5.9.2  -client dist/clientjms/P1-jms-
clientjms.jar -browse
mar 19, 2021 12:36:54 AM org.hibernate.validator.internal.util.Version <clinit>
INFO: HV000001: Hibernate Validator 5.1.2.Final
mar 19, 2021 12:36:54 AM 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 19, 2021 12:36:54 AM com.sun.messaging.jms.ra.ResourceAdapter start
INFORMACIÓN: MQJMSRA_RA1101: GlassFish MQ JMS Resource Adapter starting: broker is REMOTE, connection mode is TCP
mar 19, 2021 12:36:54 AM com.sun.messaging.jms.ra.ResourceAdapter start
INFORMACIÓN: MQJMSRA_RA1101: GlassFish MQ JMS Resource Adapter Started:REMOTE
Cola de mensajes vacía!
```

Comprobamos en Tora como el código de respuesta se ha puesto a 999:



Y que el saldo ha vuelto a su estado original: