		<b>Escuela Politécnica Superior</b> <b>Ingeniería Informática</b> <b>Prácticas de Sistemas Informáticos 2</b>			
<b>Grupo</b>	<b>2401</b>	<b>Práctica</b>	1B	<b>Fecha</b>	1203/2018
<b>Alumno/a</b>		Marcos Manchón, Pablo			
<b>Alumno/a</b>		Nevado Catalán, David			

## Cuestión 1

Librerías y anotaciones importadas en **VisaDaoLocal.java**:

- **import java.sql.Connection:**

Es la clase encargada de establecer la conexión con la base de datos mediante el método **getConnection()** y a partir de su instanciación podremos hacer invocaciones para crear sentencias sql y ejecutarlas en la base de datos.

- **import java.sql.PreparedStatement:**

Esta clase se encarga de almacenar sentencias preparadas que podrán crearse a partir de una string con una sentencia SQL. Tras ello pueden insertarse los argumentos de la consulta y ejecutarse mediante el método **executeQuery()**.

- **import java.sql.ResultSet:**

Es la clase encargada de almacenar los resultados de la consulta sql devueltos por **executeQuery()**. Se podrá iterar por los resultados mediante el método **next()**.

- **import java.sql.SQLException:**

Es la excepción devuelta por las clases y métodos de la librería de sql.

- **import java.sql.Statement:**

Clase para ejecutar sentencias sin preparar mediante el método **executeQuery(qry)** donde **qry** es un string con la sentencia sql a ejecutar.

- **import javax.ejb.Local:**

La anotación **@Local**, en este caso utilizada en la interfaz **VisaDaoLocal** para marcarla como una "local business interface" y poder compartirla con el lado del cliente.

## Ejercicio 1

Modificamos el código de los archivos **VisaDaoBean.java** como se especifica en el enunciado, cambiando el método **getPagos()** para que devuelva un array de pagos e implemente la interfaz **VisaDaoLocal**. Tras esto ya podemos compilar y empaquetar el servidor.

## Ejercicio 2

Modificamos el servlet **procesaPago** que está situado en el archivo **procesaPago.java** del cliente como se especifica en el enunciado, tras ello y tras compilar el servidor (pues es necesario para poder acceder a las clases por la anotación **Local**) podemos compilar y empaquetar de forma correcta el cliente.

## Cuestión 2

Tras examinar el contenido de **application.xml** situado en **conf/application/META-INF/** vemos que hace referencia a los archivos **P1-ejb.jar** y **P1-ejb-cliente.war**:

(application.xml)

- **P1-ejb-cliente.war**: Este fichero contiene las clases de servlets (.class), páginas dinámicas (.jsp) y HTML. Generado al empaquetar el cliente.

```
david@david-LY ~/Desktop/Pr4/Si2/virtual_machines
david@david-LY ~/Desktop/Pr4/Si2/si2/PracticalB/P1-ejb $ jar -tf dist/client/P1-ejb-
cliente.war
META-INF/
META-INF/MANIFEST.MF
WEB-INF/
WEB-INF/classes/
WEB-INF/classes/ssii2/
WEB-INF/classes/ssii2/controlador/
WEB-INF/classes/ssii2/filtros/
WEB-INF/classes/ssii2/visa/
WEB-INF/classes/ssii2/visa/error/
WEB-INF/lib/
error/
WEB-INF/classes/ssii2/controlador/ComienzaPago.class
WEB-INF/classes/ssii2/controlador/DelPagos.class
WEB-INF/classes/ssii2/controlador/GetPagos.class
WEB-INF/classes/ssii2/controlador/ProcesaPago.class
WEB-INF/classes/ssii2/controlador/ServletRaiz.class
WEB-INF/classes/ssii2/filtros/CompruebaSesion.class
WEB-INF/classes/ssii2/visa/ValidadorTarjeta.class
WEB-INF/classes/ssii2/visa/error/ErrorVisa.class
WEB-INF/classes/ssii2/visa/error/ErrorVisaCVV.class
WEB-INF/classes/ssii2/visa/error/ErrorVisaFechaCaducidad.class
WEB-INF/classes/ssii2/visa/error/ErrorVisaFechaEmision.class
WEB-INF/classes/ssii2/visa/error/ErrorVisaNumero.class
WEB-INF/classes/ssii2/visa/error/ErrorVisaTitular.class
WEB-INF/web.xml
borradoerror.jsp
borradook.jsp
cabecera.jsp
error/muestraerror.jsp
formdatosvisa.jsp
listapagos.jsp
pago.html
pagoexito.jsp
pie.html
testbd.jsp
```

(P1-ejb-cliente.war)

- **P1-ejb.jar:** Generado al empaquetar el servidor. Contiene las clases java del servidor y un descriptor del EJB (META-INF/sun-ejb-jar.xml)

```
david@david-LY ~/Desktop/Pr4/Si2/virtual_machines
david@david-LY ~/Desktop/Pr4/Si2/si2/PracticalB/P1-ejb $ jar -tvf dist/server/P1-ejb-
.jar
0 Fri Mar 02 00:35:08 CET 2018 META-INF/
104 Fri Mar 02 00:35:06 CET 2018 META-INF/MANIFEST.MF
0 Fri Mar 02 00:34:10 CET 2018 ssii2/
0 Fri Mar 02 00:34:10 CET 2018 ssii2/visa/
255 Fri Mar 02 00:35:06 CET 2018 META-INF/sun-ejb-jar.xml
1729 Fri Mar 02 00:34:10 CET 2018 ssii2/visa/DBTester.class
1464 Fri Mar 02 00:34:10 CET 2018 ssii2/visa/PagoBean.class
856 Fri Mar 02 00:34:10 CET 2018 ssii2/visa/TarjetaBean.class
7029 Fri Mar 02 00:34:10 CET 2018 ssii2/visa/VisaDAOBean.class
593 Fri Mar 02 00:34:10 CET 2018 ssii2/visa/VisaDAOLocal.class
david@david-LY ~/Desktop/Pr4/Si2/si2/PracticalB/P1-ejb $
```

- **P1-ejb.ear:** Generado al empaquetar la aplicación, contiene los archivos generados anteriormente (.jar, .war) y el fichero de descripción de application.xml

```

david@david-LY ~/Desktop/Pr4/Si2/virtual_machines
david@david-LY ~/Desktop/Pr4/Si2/si2/Practical1B/P1-ejb $ jar -tvf dist/P1-ejb.ear
0  Fri Mar 02 00:53:04 CET 2018 META-INF/
104 Fri Mar 02 00:53:02 CET 2018 META-INF/MANIFEST.MF
508 Wed Feb 28 12:59:32 CET 2018 META-INF/application.xml
20936 Fri Mar 02 00:52:20 CET 2018 P1-ejb-cliente.war
6874 Fri Mar 02 00:52:26 CET 2018 P1-ejb.jar
david@david-LY ~/Desktop/Pr4/Si2/si2/Practical1B/P1-ejb $

```

## Ejercicio 3

Modificamos los archivos build.properties y postgresql.properties con las direcciones del cliente y servidor.

build.properties	postgresql.properties
as.host.client=10.1.9.2	db.host=10.1.9.1
as.host.server=10.1.9.2	db.client.host=10.1.9.2

Tras realizar todos los pasos descritos procedemos a desplegar la aplicación con el comando **ant desplegar**. Podemos ver que efectivamente el despliegue ha sido correcto accediendo a la consola de la aplicación, donde podemos ver los servlets desplegados.

The screenshot shows the GlassFish Server Open Source Edition console. The left sidebar lists various tasks and resources, with 'P1-ejb' selected under 'Applications'. The main panel shows the configuration for the application, including the 'Java Web Start' section which is 'Enabled'. Below this, the 'Location' is set to '\$(com.sun.aas.instanceRootURI)/applications/P1-ejb/' and the 'Deployment Order' is 100. The 'Libraries' section is empty. The 'Description' field is also empty. At the bottom, a table titled 'Modules and Components (9)' lists the deployed modules and their components.

Module Name	Engines	Component Name	Type	Action
P1-ejb-cliente.war	[web]	default	Servlet	Launch
P1-ejb-cliente.war		jsp	Servlet	
P1-ejb-cliente.war		DelPagos	Servlet	
P1-ejb-cliente.war		ProcesaPago	Servlet	
P1-ejb-cliente.war		GetPagos	Servlet	
P1-ejb-cliente.war		ComienzaPago	Servlet	
P1-ejb.jar	[ejb, weld]	VisaDAOBean	StatelessSessionBean	

## Ejercicio 4

1. Realizamos un pago a través de la página **testbd.jsp**.

**Pago con tarjeta**

**Proceso de un pago**

Id Transacción: 12

Id Comercio: 34

Importe: 56

Numero de visa: 1899 8476 1549 9754

Titular: Luis Mojamuto Vallejo

Fecha Emisión: 05/09

Fecha Caducidad: 01/20

CVV2: 370

Modo debug: ☒ True ☐ False

Direct Connection: ☐ True ☒ False

Use Prepared: ☒ True ☐ False

Pagar

2. Recibimos confirmación de que el pago se ha realizado.

**Pago con tarjeta**

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

idTransaccion: 12

idComercio: 34

importe: 56.0

codRespuesta: 000

idAutorizacion: 1

[Volver al comercio](#)

Prácticas de Sistemas Informáticos II

3. Buscamos el pago y comprobamos que se ha guardado.

**Pago con tarjeta**

Lista de pagos del comercio 34

idTransaccion	Importe	codRespuesta	idAutorizacion
12	56.0	000	1

[Volver al comercio](#)

Prácticas de Sistemas Informáticos II

4. Borramos el pago.

**Pago con tarjeta**

Se han borrado 1 pagos correctamente para el comercio 34

[Volver al comercio](#)

Prácticas de Sistemas Informáticos II

Realizamos el mismo procedimiento desde la página **pago.html** para comprobar el correcto funcionamiento.

A screenshot of a web browser showing a form titled 'pago.html'. The form contains three input fields: 'Id Transacción:' with the value '1233', 'Id Comercio:' with the value '3443', and 'Importe:' with the value '56'. Below these fields is a button labeled 'Envia Datos Pago'.

A screenshot of a web browser showing a page titled 'Pago con tarjeta'. The page contains a form for card payment details. Fields include: 'Numero de visa:' (1899 8476 1549 9754), 'Titular:' (Luis Mojamoto Vallejo), 'Fecha Emisión:' (05/09), 'Fecha Caducidad:' (01/20), and 'CVV2:' (370). There is a 'Pagar' button. Below the form, it displays transaction details: 'Id Transacción: 1233', 'Id Comercio: 3443', and 'Importe: 56.0'. The footer text is 'Prácticas de Sistemas Informáticos II'.

A screenshot of a web browser showing a confirmation page titled 'Pago con tarjeta'. The page states 'Pago realizado con éxito. A continuación se muestra el comprobante del mismo:'. It lists transaction details: 'idTransaccion: 1233', 'idComercio: 3443', 'importe: 56.0', 'codRespuesta: 000', and 'idAutorizacion: 1'. There is a link 'Volver al comercio'. The footer text is 'Prácticas de Sistemas Informáticos II'.

## Ejercicio 5

Seguimos los pasos enumerados en el enunciado:

- Crear la interfaz **VisaDAORemote** igual que **VisaDAOLocal** cambiando simplemente la etiqueta **@Local** por **@Remote**.
- Hacer que **VisaDAOBean** implemente ahora también la interfaz remota.
- Hacer **PagoBean** y **TarjetaBean** serializables.

Volvemos a repetir los pasos para desplegar la aplicación y comprobamos el despliegue correcto.

## Ejercicio 6

Realizamos los cambios especificados en el enunciado para la creación de el cliente remoto y su despliegue en la dirección **10.1.9.1**. Comprobamos que se efectúa de forma correcta un pago.

- Se realiza un pago desde el cliente remoto y se comprueba que se ha realizado con éxito:



The screenshot shows a web browser window with the address bar displaying "10.1.9.1:8080/P1/procesapago". The page title is "Pago con tarjeta". The main content area displays the message "Pago realizado con éxito. A continuación se muestra el comprobante del mismo:" followed by transaction details: "idTransaccion: 12", "idComercio: 34", "importe: 56.0", "codRespuesta: 000", and "idAutorizacion: 1". A link "Volver al comercio" is provided below the details. The footer of the page reads "Prácticas de Sistemas Informáticos II".

- Se comprueba que aparece listado el pago



The screenshot shows a web browser window with the address bar displaying "10.1.9.1:8080/P1/getpagos". The page title is "Pago con tarjeta". The main content area displays the message "Lista de pagos del comercio 34" above a table. The table has four columns: "idTransaccion", "Importe", "codRespuesta", and "idAutorizacion". The first row of data shows values: "12", "56.0", "000", and "1". A link "Volver al comercio" is provided below the table. The footer of the page reads "Prácticas de Sistemas Informáticos II".

idTransaccion	Importe	codRespuesta	idAutorizacion
12	56.0	000	1

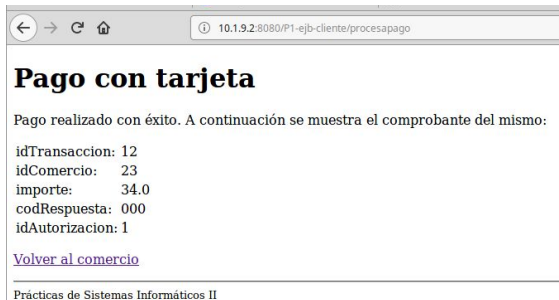
## Ejercicio 7

Como se especifica en el enunciado añadimos la variable **private double saldo** a la clase **TarjetaBean** y en **VisaDaoBean** modificamos los métodos **realizaPago** para que se tenga en cuenta el nuevo atributo saldo de la tarjeta y se comprueben los casos de error y se gestionen las excepciones y se realice rollback en tales casos. También modificamos **procesaPago** para que gestione la excepción **EJBException** de forma adecuada.



## Ejercicio 8

- Realizamos un pago y comprobamos que se modifica el saldo en la BD.



10.1.9.2:8080/P1-ejb-cliente/procesapago

### Pago con tarjeta

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

idTransaccion: 12  
idComercio: 23  
importe: 34.0  
codRespuesta: 000  
idAutorizacion: 1

[Volver al comercio](#)

Prácticas de Sistemas Informáticos II

```
david@david-LY ~/Desktop/Pr4/Si2/si2/PracticalB/P1-ejb-transaccional $ psql -U alumnodb -h 10.1.9.1
-p 5432 -d visa
psql (9.5.12, server 8.4.10)
SSL connection (protocol: TLSv1, cipher: DHE-RSA-AES256-SHA, bits: 256, compression: off)
Type "help" for help.

visa=# select * from tarjeta where numeroTarjeta='8816 2190 2967 8500';
 numeroTarjeta | titular | validadesde | validahasta | codigoverificacion | saldo
-----
 8816 2190 2967 8500 | Camilo Moss Lopez | 07/09 | 01/20 | 649 | 966
(1 row)

visa=#
```

- Tratamos de realizar un pago incorrecto (con identificador de transacciones y comercio duplicados) y comprobamos que no afecta a la BD (se hace un rollback).



### Pago con tarjeta

Pago incorrecto

Prácticas de Sistemas Informáticos II

```
david@david-LY ~/Desktop/Pr4/Si2/si2/PracticalB/P1-ejb-transaccional $ psql -U alumnodb -h 10.1.9.1
-p 5432 -d visa
psql (9.5.12, server 8.4.10)
SSL connection (protocol: TLSv1, cipher: DHE-RSA-AES256-SHA, bits: 256, compression: off)
Type "help" for help.

visa=# select * from tarjeta where numeroTarjeta='8816 2190 2967 8500';
 numeroTarjeta | titular | validadesde | validahasta | codigoverificacion | saldo
-----
 8816 2190 2967 8500 | Camilo Moss Lopez | 07/09 | 01/20 | 649 | 966
(1 row)

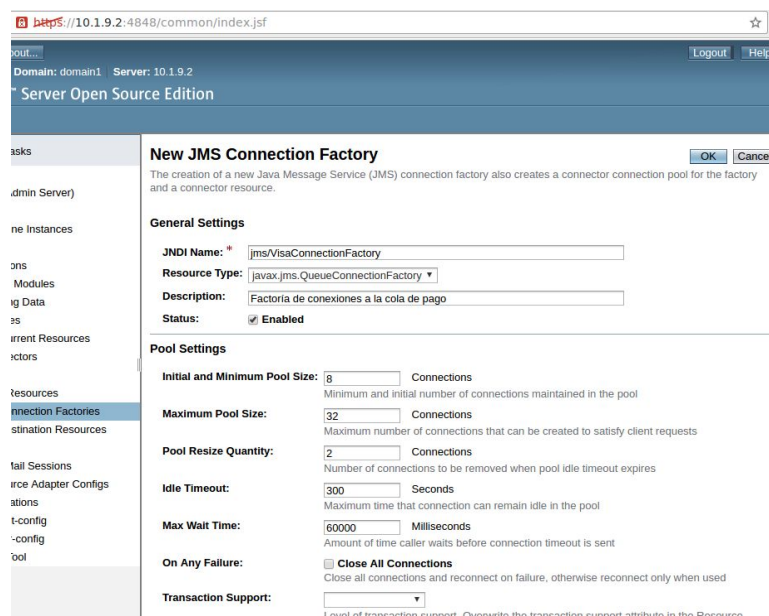
visa=# select * from tarjeta where numeroTarjeta='8816 2190 2967 8500';
 numeroTarjeta | titular | validadesde | validahasta | codigoverificacion | saldo
-----
 8816 2190 2967 8500 | Camilo Moss Lopez | 07/09 | 01/20 | 649 | 966
(1 row)

visa=#
```

- Lo mismo ocurre si se trata de realizar un pago que excede el saldo de la tarjeta.

## Ejercicio 9

Creamos manualmente la connection factory.



https://10.1.9.2:4848/common/index.jsf

Logout Help

Domain: domain1 Server: 10.1.9.2

Server Open Source Edition

Tasks

Admin Server

One Instance

One Modules

Log Data

Resources

Connection Factories

Destination Resources

Full Sessions

Source Adapter Configs

Test-config

Tool

### New JMS Connection Factory

The creation of a new Java Message Service (JMS) connection factory also creates a connector connection pool for the factory and a connector resource.

General Settings

JNDI Name: \* jms/VisaConnectionFactory

Resource Type: javax.jms.QueueConnectionFactory

Description: Factoría de conexiones a la cola de pago

Status: ☒ Enabled

Pool Settings

Initial and Minimum Pool Size: 8 Connections  
Minimum and initial number of connections maintained in the pool

Maximum Pool Size: 32 Connections  
Maximum number of connections that can be created to satisfy client requests

Pool Resize Quantity: 2 Connections  
Number of connections to be removed when pool idle timeout expires

Idle Timeout: 300 Seconds  
Maximum time that connection can remain idle in the pool

Max Wait Time: 60000 Milliseconds  
Amount of time caller waits before connection timeout is sent

On Any Failure: ☐ Close All Connections  
Close all connections and reconnect on failure, otherwise reconnect only when used

Transaction Support:   
Level of transaction support. Overwrite the transaction support attribute in the Resource



## Ejercicio 10

Creamos manualmente la cola de mensajes.

The screenshot shows the JBoss Administration Console interface. The top navigation bar includes the URL `/10.1.9.2:4848/common/index.jsf`, a star icon, and a menu icon. Below the navigation bar, the page title is "Open Source Edition". The main content area is titled "New JMS Destination Resource" and includes a sub-header "The creation of a new Java Message Service (JMS) destination resource also creates an admin object resource." The form contains the following fields:

- JNDI Name:** `jms/VisaPagosQueue`
- Physical Destination Name:** `VisaPagosQueue`
- Resource Type:** `javax.jms.Queue`
- Description:** `Cola de pagos visa`
- Status:** ☒ **Enabled**

Below the form is a section titled "Additional Properties (0)" with buttons for "Add Property" and "Delete Properties". A table with columns "Select", "Name", "Value", and "Description" is shown, with the message "No items found." below it.

## Ejercicio 11

Modificamos el fichero `sun-ejb-jar.xml` para que el mdb pueda conectarse a su connection factory.

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

Añadimos las dos sentencias preparadas en `VisaCancelacionJMSBean` para poder cancelar el pago:

```
private static final String UPDATE_CANCELA_QRY = "UPDATE pago SET  
codRespuesta=999 WHERE idAutorizacion=?";
```

```
private static final String RECTIFICA_PAGO_QRY = "UPDATE tarjeta AS t1 " +  
"SET saldo = saldo + importe " +  
"FROM pago WHERE pago.idAutorizacion=? " +  
"AND pago.numeroTarjeta=t1.numeroTarjeta";
```

Y implementamos en `onMessage()` las llamadas necesarias para que se ejecuten las queries al recibir un mensaje.

## Ejercicio 12

En el primer método simplemente anotamos las variables **queue** y **connectionFactory**.

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

En el segundo hacemos una búsqueda explícita mediante JNDI, inicializando las dos variables de la clase de forma dinámica en el main con un lookup.

```
InitialContext jndi = new InitialContext();  
connectionFactory = (ConnectionFactory)jndi.lookup("jms/VisaConnectionFactory");  
queue = (Queue)jndi.lookup("jms/VisaPagosQueue");
```

En el método de acceso mediante **JNDI** se proporciona transparencia de ubicación, pues no es necesario en tiempo de compilación conocer el nombre de los recursos, evitando recompilar la clase en caso de cambiar el nombre de estos y pudiendo compilarse antes de conocerse.

## Ejercicio 13

Modificamos los archivos build.properties y jms.properties

build.properties	jms.properties
as.host.mdb=10.1.9.2	jms.name=jms/VisaPagosQueue
	jms.factoryname=jms/VisaConnectionFactory
	jms.physname=Visa
	as.host.server=10.1.9.2

Efectivamente, tras ejecutar el comando **ant todo**, al entrar en la consola de administración del servidor que estaba activo (**10.1.9.2**) vemos que se ha creado la cola de mensajes.

**JMS Connection Factories**

Java Message Service (JMS) connection factories are objects that allow an application to create other JMS objects programmatically. Click New to create a new connection factory to modify its properties.

Select	JNDI Name	Logical JNDI Name	Enabled	Resource Type
<input type="checkbox"/>	jms/_defaultConnectionFactory	java:comp/DefaultJMSConnectionFactory	✓	javax.jms.ConnectionFactory
<input type="checkbox"/>	jms/VisaConnectionFactory		✓	javax.jms.QueueConnectionFactory

**JMS Destination Resources**

JMS destinations serve as the repositories for messages. Click New to create a new destination resource. Click the name of a destination resource to modify its properties.

Select	JNDI Name	Enabled	Resource Type
<input type="checkbox"/>	jms/VisaPagosQueue	✓	javax.jms.Queue

Del fichero **jms.xml** obtenemos el siguiente comando:

Sustituyendo los nombres de las variables el comando sería:

```
asadmin --user admin --passwordfile passwordfile --host 10.1.9.2 --port 4848 create-jms-resource --restype javax.jms.QueueConnectionFactory --enabled=true --property Name=VisaPagosQueue jms/VisaPagosQueue
```

## Ejercicio 14

En el main de VisaQueueMessageProducer.java añadimos el código necesario para que los mensajes recibidos por argumento se envíen a la cola de mensajes.

```
message = session.createTextMessage();
message.setText(args[0]);
messageProducer.send(message);
```

- Detenemos la aplicación P1-jms-mdb

✓ New values successfully saved.

### Edit Application

Modify an existing application or module.

**Name:** P1-jms-mdb  
**Status:** ☐ Enabled  
**Location:** \${com.sun.aas.instanceRootURI}/applications/P1-jms-mdb/

- Modificamos la variable con la ip de la máquina virtual y reiniciamos el servidor.

### Edit JMS Host

The Java Message Service (JMS) host specifies the system where the JMS service is running.

[Load Defaults](#)

**Configuration Name:** server-config

**Name:** default\_JMS\_host  
**Host:**   
Name or IP address; if name, must contain only alphanumeric, underscore, dash, or dot characters

- Salida tras ejecución de los comandos para añadir un mensaje a la cola y ver el número de mensajes en ella.

```
e321079@localhost:~/Desktop/si2/PracticalB/P1-jms$ /usr/local/glassfish-4.1.1/glassfish/bin/appclient -targetserver 10.1.9.2 -client dist/cli
entjms/P1-jms-clientjms.jar 1
mar 12, 2018 4:58:34 PM org.hibernate.validator.internal.util.Version <clinit>
INFO: HV000001: Hibernate Validator 5.1.2.Final
mar 12, 2018 4:58:34 PM com.sun.messaging.jms.ra.ResourceAdapter start
INFORMACIÓN: MQJMSRA_RAI101: GlassFish MQ JMS Resource Adapter: Version: 5.1.1 (Build 2-c) Compile: March 17 2015 1045
mar 12, 2018 4:58:34 PM com.sun.messaging.jms.ra.ResourceAdapter start
INFORMACIÓN: MQJMSRA_RAI101: GlassFish MQ JMS Resource Adapter starting: broker is REMOTE, connection mode is TCP
mar 12, 2018 4:58:34 PM com.sun.messaging.jms.ra.ResourceAdapter start
INFORMACIÓN: MQJMSRA_RAI101: GlassFish MQ JMS Resource Adapter Started:REMOTE
e321079@localhost:~/Desktop/si2/PracticalB/P1-jms$
```

```
e321079@localhost:~/Desktop/si2/PracticalB/P1-jms$ /usr/local/glassfish-4.1.1/glassfish/bin/appclient -targetserver 10.1.9.2 -client dist/cl
entjms/P1-jms-clientjms.jar -browse
mar 12, 2018 5:00:46 PM org.hibernate.validator.internal.util.Version <clinit>
INFO: HV000001: Hibernate Validator 5.1.2.Final
mar 12, 2018 5:00:46 PM com.sun.messaging.jms.ra.ResourceAdapter start
INFORMACIÓN: MQJMSRA_RAI101: GlassFish MQ JMS Resource Adapter: Version: 5.1.1 (Build 2-c) Compile: March 17 2015 1045
mar 12, 2018 5:00:46 PM com.sun.messaging.jms.ra.ResourceAdapter start
INFORMACIÓN: MQJMSRA_RAI101: GlassFish MQ JMS Resource Adapter starting: broker is REMOTE, connection mode is TCP
mar 12, 2018 5:00:47 PM com.sun.messaging.jms.ra.ResourceAdapter start
INFORMACIÓN: MQJMSRA_RAI101: GlassFish MQ JMS Resource Adapter Started:REMOTE
Mensajes en cola:
1
1
```

Ahora comprobamos que se las cancelaciones se gestionan correctamente, para ello volvemos a habilitar p1-jms-mdb y realizamos un pago con P1-ejb-transaccional

← → ↻ 10.1.9.2:8080/P1-ejb-cliente/procesapago

## Pago con tarjeta

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

idTransaccion: 2  
idComercio: 2  
importe: 23.0  
codRespuesta: 000  
idAutorizacion: 1

[Volver al comercio](#)

Comprobamos en la base de datos que efectivamente se han realizado los cambios en la tabla pagos y tarjeta.

```
visa=# select * from pago;
 idautorizacion | idtransaccion | codrespuesta | importe | idcomercio | numerotarjeta | fecha
-----+-----+-----+-----+-----+-----+-----
          1 | 2            | 000         | 23      | 2          | 1111 2222 3333 4444 | 2018-03-12 11:24:45.620744
(1 row)
```

```
visa=# select * from tarjeta where numerotarjeta='1111 2222 3333 4444';
 numerotarjeta | titular | validadesde | validahasta | codigoverificacion | saldo
-----+-----+-----+-----+-----+-----
1111 2222 3333 4444 | Jose Garcia | 11/09      | 11/20      | 123                | 977
```

A continuación procedemos a cancelar el pago con el cliente P1-jms como se indica en el enunciado. Tras compilar P1-jms

- Desde PC1 host:

- scp dist/clientjms/P1-jms-clientjms.jar si2@10.X.Y.1:/tmp
- Desde la máquina virtual **10.1.9.1** mediante ssh:
  - export JAVA\_HOME=/usr/lib/jvm/java-8-oracle/
  - /opt/glassfish4/glassfish/bin/appclient -targetserver 10.X.Y.2 -client /tmp/P1-jms-clientjms.jar 1

Vemos que el código de respuesta se ha modificado y que el saldo de la tarjeta se ha restaurado.

idautorizacion	idtransaccion	codrespuesta	importe	idcomercio	numerotarjeta	fecha
1	2	999	23	2	1111 2222 3333 4444	2018-03-12 11:24:45.620744

```
visa=# select * from tarjeta where numerotarjeta='1111 2222 3333 4444';
```

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

(1 row)