

		<b>Escuela Politécnica Superior</b> <b>Ingeniería Informática</b> <b>Prácticas de Sistemas Informáticos 2</b>			
<b>Grupo</b>	2401	<b>Pareja</b>	2	<b>Fecha</b>	13/03/2018
<b>Alumno/a</b>	Gómez, Martínez, Javier				
<b>Alumno/a</b>	Li, Hu, Carlos				

## Práctica 1B: Arquitectura de JAVA EE (Segunda parte)

### Cuestión número 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.

Nos hemos fijado en que esencialmente usa librerías de java.sql para poder realizar las operaciones en la base de datos:

```
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;
```

También podemos encontrar en la interfaz local las definiciones de los métodos que deben implementar cualquier clase que sea VisaDAO (en nuestro caso VisaDAOBean):

```
@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);
}
```

## **Ejercicio número 1:**

Introduzca las siguientes modificaciones en el bean VisaDAOBean para convertirlo en un EJB de sesión stateless con interfaz local: Hacer que la clase implemente la interfaz local y convertirla en un EJB stateless mediante la anotación Stateless

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

```

Eliminar el constructor por defecto de la clase.

Ajustar los métodos `getPagos()` a la interfaz definida en VisaDAOLocal.

El cambio más relevante sería el de el ajuste del método `getPagos`, para ello, respetando esta interfaz:

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

modificamos los módulos donde se llama a estas funciones, para que los retornos sean compatibles.

## **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 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()
...

```

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()`).

Los cambios descritos han sido reflejados en el código proporcionado.

## Cuestión número 2:

Abrir 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.

El contenido de los archivos siguientes es:

- `application.xml`: describe donde se despliega la aplicación, sus componentes y como está configurada la aplicación.

```
<?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>
```

- `P1-ejb.jar`: Contiene un xml que describe el EJB (`sun-ejb-jar.xml`) y las clases java que usará el servidor.

```
e321083@localhost:~/Desktop/si2/P1B/P1-ejb/dist/server$ jar -tvf P1-ejb.jar
0 Tue Feb 27 17:24:50 CET 2018 META-INF/
104 Tue Feb 27 17:24:48 CET 2018 META-INF/MANIFEST.MF
0 Tue Feb 27 17:23:26 CET 2018 ssii2/
0 Tue Feb 27 17:23:26 CET 2018 ssii2/visa/
0 Tue Feb 27 17:23:26 CET 2018 ssii2/visa/dao/
0 Tue Feb 27 17:23:26 CET 2018 ssii2/visa/error/
255 Tue Feb 27 17:24:48 CET 2018 META-INF/sun-ejb-jar.xml
1464 Tue Feb 27 17:23:26 CET 2018 ssii2/visa/PagoBean.class
856 Tue Feb 27 17:23:26 CET 2018 ssii2/visa/TarjetaBean.class
593 Tue Feb 27 17:23:26 CET 2018 ssii2/visa/VisaDAOLocal.class
1723 Tue Feb 27 17:23:26 CET 2018 ssii2/visa/dao/DBTester.class
6976 Tue Feb 27 17:23:26 CET 2018 ssii2/visa/dao/VisaDAOBean.class
616 Tue Feb 27 17:23:26 CET 2018 ssii2/visa/error/ErrorVisa.class
198 Tue Feb 27 17:23:26 CET 2018 ssii2/visa/error/ErrorVisaCVV.class
209 Tue Feb 27 17:23:26 CET 2018 ssii2/visa/error/ErrorVisaFechaCaducidad.class
207 Tue Feb 27 17:23:26 CET 2018 ssii2/visa/error/ErrorVisaFechaEmision.class
201 Tue Feb 27 17:23:26 CET 2018 ssii2/visa/error/ErrorVisaNumero.class
202 Tue Feb 27 17:23:26 CET 2018 ssii2/visa/error/ErrorVisaTitular.class
e321083@localhost:~/Desktop/si2/P1B/P1-ejb/dist/server$
```

- `P1-ejb-cliente.war`: contiene las clases generadas por los controladores (servlets) y las vistas (JSP) junto a los `.html` que requiera para su página web.

```
e321083@localhost:~/Desktop/si2/P1B/P1-ejb/dist/client$ jar -tvf P1-ejb-cliente.war
0 Tue Feb 27 17:38:22 CET 2018 META-INF/
104 Tue Feb 27 17:38:20 CET 2018 META-INF/MANIFEST.MF
0 Tue Feb 27 17:38:20 CET 2018 WEB-INF/
0 Tue Feb 27 17:36:08 CET 2018 WEB-INF/classes/
0 Tue Feb 27 17:36:08 CET 2018 WEB-INF/classes/ssii2/
0 Tue Feb 27 17:36:08 CET 2018 WEB-INF/classes/ssii2/controlador/
0 Tue Feb 27 17:36:08 CET 2018 WEB-INF/classes/ssii2/filtros/
0 Tue Feb 27 17:36:08 CET 2018 WEB-INF/classes/ssii2/visa/
0 Tue Feb 27 17:36:08 CET 2018 WEB-INF/classes/ssii2/visa/error/
0 Tue Feb 27 17:15:46 CET 2018 WEB-INF/lib/
0 Tue Feb 27 17:38:20 CET 2018 error/
2844 Tue Feb 27 17:36:08 CET 2018 WEB-INF/classes/ssii2/controlador/ComienzaPago.class
1513 Tue Feb 27 17:36:08 CET 2018 WEB-INF/classes/ssii2/controlador/DelPagos.class
1365 Tue Feb 27 17:36:08 CET 2018 WEB-INF/classes/ssii2/controlador/GetPagos.class
4919 Tue Feb 27 17:36:08 CET 2018 WEB-INF/classes/ssii2/controlador/ProcesaPago.class
1894 Tue Feb 27 17:36:08 CET 2018 WEB-INF/classes/ssii2/controlador/ServletRaiz.class
2608 Tue Feb 27 17:36:08 CET 2018 WEB-INF/classes/ssii2/filtros/CompruebaSesion.class
3170 Tue Feb 27 17:36:08 CET 2018 WEB-INF/classes/ssii2/visa/ValidadorTarjeta.class
616 Tue Feb 27 17:36:08 CET 2018 WEB-INF/classes/ssii2/visa/error/ErrorVisa.class
198 Tue Feb 27 17:36:08 CET 2018 WEB-INF/classes/ssii2/visa/error/ErrorVisaCVV.class
209 Tue Feb 27 17:36:08 CET 2018 WEB-INF/classes/ssii2/visa/error/ErrorVisaFechaCaducidad.class
207 Tue Feb 27 17:36:08 CET 2018 WEB-INF/classes/ssii2/visa/error/ErrorVisaFechaEmision.class
201 Tue Feb 27 17:36:08 CET 2018 WEB-INF/classes/ssii2/visa/error/ErrorVisaNumero.class
202 Tue Feb 27 17:36:08 CET 2018 WEB-INF/classes/ssii2/visa/error/ErrorVisaTitular.class
6020 Tue Feb 27 17:38:20 CET 2018 WEB-INF/web.xml
455 Tue Feb 27 17:38:20 CET 2018 borradoerror.jsp
501 Tue Feb 27 17:38:20 CET 2018 borradook.jsp
509 Tue Feb 27 17:38:20 CET 2018 cabecera.jsp
283 Tue Feb 27 17:38:20 CET 2018 error/muestraerror.jsp
2729 Tue Feb 27 17:38:20 CET 2018 formdatosvisa.jsp
1257 Tue Feb 27 17:38:20 CET 2018 listapagos.jsp
1178 Tue Feb 27 17:38:20 CET 2018 pago.html
1142 Tue Feb 27 17:38:20 CET 2018 pagoexito.jsp
104 Tue Feb 27 17:38:20 CET 2018 pie.html
5011 Tue Feb 27 17:38:20 CET 2018 testbd.jsp
e321083@localhost:~/Desktop/si2/P1B/P1-ejb/dist/client$
```

- P1-ejb.ear: contiene los 2 paquetes (.jar y .war) anteriores junto al descriptor application.xml

```
e321083@localhost:~/Desktop/si2/P1B/P1-ejb/dist$ jar -tvf P1-ejb.ear
0 Tue Feb 27 17:39:10 CET 2018 META-INF/
104 Tue Feb 27 17:39:08 CET 2018 META-INF/MANIFEST.MF
508 Sat Feb 11 23:33:00 CET 2012 META-INF/application.xml
20937 Tue Feb 27 17:38:20 CET 2018 P1-ejb-cliente.war
9093 Tue Feb 27 17:24:48 CET 2018 P1-ejb.jar
e321083@localhost:~/Desktop/si2/P1B/P1-ejb/dist$
```

### 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.**

La configuración actual es esta:

*as.host.client=10.1.2.2*

*as.host.server=10.1.2.2*

**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.**

La configuración actual es esta:

*db.client.host=10.1.2.2*

*db.host=10.1.2.1*

**Desplegar la aplicación de empresa ant desplegar.**

Tras el despliegue, comprobamos en la aplicación de administración de Glassfish que en este caso, la aplicación no se ha desplegado, como en las prácticas anteriores, como Web Application, sino que se encuentra bajo el apartado de Enterprise Applications

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]			
P1-eib.iar		VisaDAOBean	StatelessSessionBean	

## 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.

Procedemos a probar el funcionamiento en ambos módulos:

- Pago.html

Rellenamos el formulario con estos datos:

## Pago con tarjeta

Numero de visa:	<input type="text" value="2347 4840 5058 7931"/>
Titular:	<input type="text" value="Gabriel Avila Locke"/>
Fecha Emisión:	<input type="text" value="11/09"/>
Fecha Caducidad:	<input type="text" value="01/20"/>
CVV2:	<input type="text" value="207"/>
<input type="button" value="Pagar"/>	

Id Transacción: 10  
Id Comercion: 10  
Importe: 99.0

Prácticas de Sistemas Informáticos II

Y realizamos el pago con éxito:



# Pago con tarjeta

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

idTransaccion: 10  
idComercio: 10  
importe: 99.0  
codRespuesta: 000  
idAutorizacion: 1

[Volver al comercio](#)

---

Prácticas de Sistemas Informáticos II

Listamos los pagos con el id de comercio anterior y lo comprobamos:

# Pago con tarjeta

Lista de pagos del comercio 10

idTransaccion	Importe	codRespuesta	idAutorizacion
10	99.0	000	1

[Volver al comercio](#)

---

Prácticas de Sistemas Informáticos II

Posteriormente lo borramos:

# Pago con tarjeta

Se han borrado 1 pagos correctamente para el comercio 10

[Volver al comercio](#)

---

Prácticas de Sistemas Informáticos II

Y si tratamos de volver a listarlo, veremos que ha sido borrado correctamente:

# Pago con tarjeta

Lista de pagos del comercio 10

idTransaccion	Importe	codRespuesta	idAutorizacion
---------------	---------	--------------	----------------

[Volver al comercio](#)

---

Prácticas de Sistemas Informáticos II

- **testbd.jsp**

Procedemos rellenando el formulario con estos parámetros

## Pago con tarjeta

### Proceso de un pago

Id Transacción:	<input type="text" value="11"/>
Id Comercio:	<input type="text" value="11"/>
Importe:	<input type="text" value="11"/>
Numero de visa:	<input type="text" value="2347 4840 5058 7931"/>
Titular:	<input type="text" value="Gabriel Avila Locke"/>
Fecha Emisión:	<input type="text" value="11/09"/>
Fecha Caducidad:	<input type="text" value="01/20"/>
CVV2:	<input type="text" value="207"/>
Modo debug:	<input checked="" type="radio"/> True <input type="radio"/> False
Direct Connection:	<input type="radio"/> True <input checked="" type="radio"/> False
Use Prepared:	<input checked="" type="radio"/> True <input type="radio"/> False
<input type="button" value="Pagar"/>	

Lo pagamos correctamente:

## Pago con tarjeta

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

idTransaccion: 11  
idComercio: 11  
importe: 11.0  
codRespuesta: 000  
idAutorizacion: 1

[Volver al comercio](#)

---

Prácticas de Sistemas Informáticos II

Listamos el pagos con el id de comercio anterior:

## Pago con tarjeta

Lista de pagos del comercio 11

idTransaccion	Importe	codRespuesta	idAutorizacion
11	11.0	000	1

[Volver al comercio](#)

A continuación lo borramos:

## Pago con tarjeta

Se han borrado 1 pagos correctamente para el comercio 11

[Volver al comercio](#)

Y lo tratamos de volver a listar, pero vemos que ha sido borrado correctamente:

## Pago con tarjeta

Lista de pagos del comercio 11

idTransaccion	Importe	codRespuesta	idAutorizacion
---------------	---------	--------------	----------------

[Volver al comercio](#)



### ***Ejercicio número 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.

Los cambios esenciales que hemos tenido que realizar en el código del servidor remoto para que se despliegue correctamente son:

- Cambiar, en VisaDAORemote.java, el nombre de la interfaz a VisaDAORemote y cambiar la anotación @Local por @Remote.
- Añadir import javax.ejb.Remote; y quitar import javax.ejb.Local
- Hacer que VisaDAOBean implemente ambas interfaces, la local y la remota.
- marcar como serializables PagoBean y TarjetaBean

### ***Ejercicio número 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.

Los cambios en el código detallados previamente son bastante concisos y completos, así que vamos a omitirlos. Y por tanto, tras compilar, empaquetar y desplegar correctamente, probamos la funcionalidad:

Empezamos rellenando el formulario de pago en testbd.jsp con estos parámetros:

## Pago con tarjeta

### Proceso de un pago

Id Transacción:	<input type="text" value="12"/>
Id Comercio:	<input type="text" value="12"/>
Importe:	<input type="text" value="12"/>
Numero de visa:	<input type="text" value="2347 4840 5058 7931"/>
Titular:	<input type="text" value="Gabriel Avila Locke"/>
Fecha Emisión:	<input type="text" value="11/09"/>
Fecha Caducidad:	<input type="text" value="01/20"/>
CVV2:	<input type="text" value="207"/>
Modo debug:	<input checked="" type="radio"/> True <input type="radio"/> False
Direct Connection:	<input type="radio"/> True <input checked="" type="radio"/> False
Use Prepared:	<input checked="" type="radio"/> True <input type="radio"/> False
<input type="button" value="Pagar"/>	

A continuación realizamos el pago correctamente:

## Pago con tarjeta

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

idTransaccion: 12  
idComercio: 12  
importe: 12.0  
codRespuesta: 000  
idAutorizacion: 2

[Volver al comercio](#)

Y listamos el pago, que vemos que ha sido registrado correctamente:

## Pago con tarjeta

Lista de pagos del comercio 12

idTransaccion	Importe	codRespuesta	idAutorizacion
12	12.0	000	2

[Volver al comercio](#)

---

Prácticas de Sistemas Informáticos II

### 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;
```

Archivo VisaDAOBean.java: Importar la definición de la excepción EJBException que debe lanzar el servlet para indicar que se debe realizar un rollback:

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

Declarar un prepared statement para recuperar el saldo de una tarjeta de la base de datos.

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

Modificar el método realizaPago con las siguientes acciones:

- Recuperar el saldo de la tarjeta a través del prepared statement declarado anteriormente.
- Comprobar si el saldo es mayor o igual que el importe de la operación. Si no lo es, retornar denegando el pago (idAutorizacion= null y pago retornado=null)
- Si el saldo es suficiente, decrementarlo en el valor del importe del pago y actualizar el registro de la tarjeta para reflejar el nuevo saldo mediante el prepared statement declarado anteriormente.
- Si lo anterior es correcto, ejecutar el proceso de inserción del pago y obtención del idAutorizacion, tal como se realizaba en la práctica anterior (este código ya debe estar programado y no es necesario modificarlo).
- En caso de producirse cualquier error a lo largo del proceso (por ejemplo, si no se obtiene el idAutorizacion porque la transacción está duplicada), lanzar una excepción EJBException para retornar al cliente.

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

Hemos realizado los import y declarado los atributos privados en los ficheros correspondientes.

Los prepared statement que hemos declarado son:

Para recuperar el saldo de una tarjeta de la base de datos:

```
select saldo from tarjeta where numeroTarjeta=?
```

Para insertar el nuevo saldo calculado en la base de datos:

```
update tarjeta set saldo = ? where numeroTarjeta=?
```

A continuación hemos añadido líneas de código en el método realizaPago para implementar esta nueva funcionalidad de restar saldo a las tarjetas mediante la ejecución de los prepared statement anteriores.

Dado que no consideramos la opción de que no se ejecuten prepared statement, si se intentara hacer de esta otra forma, el método lanzaría una excepción.

Otros lugares en los que podría lanzar una excepción son:

- En el caso en el que hubiera algún tipo de error en las consultas a la base de datos
- En el caso en el que el saldo de la tarjeta fuera inferior al importe del pago

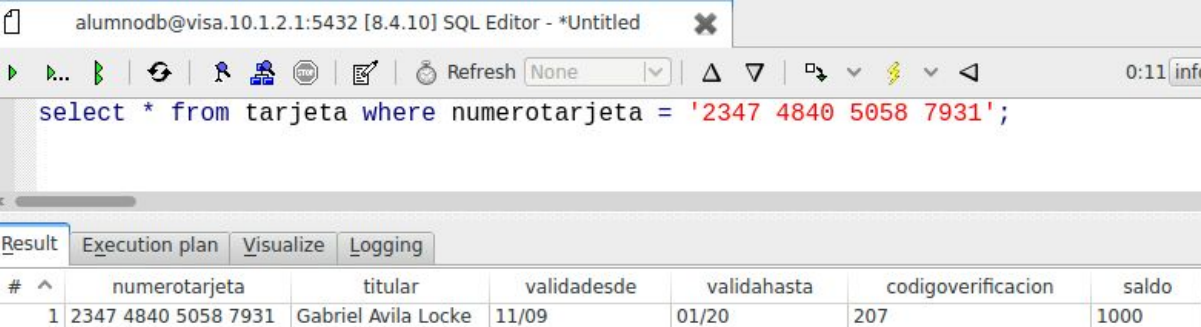
De esta forma, si sucediera algún tipo de error durante la transacción, no se efectuaría ningún cambio en la base de datos, pues lanzando la excepción *EJBException* se llevaría a cabo un rollback automático.

Finalmente, en el servlet hemos añadido un try catch alrededor de la invocación al método realizaPago para que, en caso de que se lanzara la excepción antes mencionada, se pudiera redireccionar al usuario a una página de error mediante la llamada al método *enviaError*.

## 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.**

Haremos las pruebas con esta tarjeta, que vemos que tiene un saldo inicial de 500



The screenshot shows a SQL Editor window with the title 'alumnodb@visa.10.1.2.1:5432 [8.4.10] SQL Editor - \*Untitled'. The editor contains the SQL query: `select * from tarjeta where numerotarjeta = '2347 4840 5058 7931';`. Below the editor, the 'Result' tab is active, displaying a table with the following data:

#	numerotarjeta	titular	validadesde	validahasta	codigoverificacion	saldo
1	2347 4840 5058 7931	Gabriel Avila Locke	11/09	01/20	207	1000

A continuación realizamos el pago con sus datos y un importe de 500:

← → ↻ 10.1.2.2:8080/P1-ejb-cliente/testbd.jsp

## Pago con tarjeta

### Proceso de un pago

Id Transacción:	<input type="text" value="3"/>
Id Comercio:	<input type="text" value="3"/>
Importe:	<input type="text" value="500"/>
Numero de visa:	<input type="text" value="2347 4840 5058 7931"/>
Titular:	<input type="text" value="Gabriel Avila Locke"/>
Fecha Emisión:	<input type="text" value="11/09"/>
Fecha Caducidad:	<input type="text" value="01/20"/>
CVV2:	<input type="text" value="207"/>
Modo debug:	<input checked="" type="radio"/> True <input type="radio"/> False
Direct Connection:	<input type="radio"/> True <input checked="" type="radio"/> False
Use Prepared:	<input checked="" type="radio"/> True <input type="radio"/> False
<input type="button" value="Pagar"/>	

El pago se realiza correctamente:

← → ↻ 10.1.2.2:8080/P1-ejb-cliente/procesapago

## Pago con tarjeta

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

idTransaccion: 3  
idComercio: 3  
importe: 500.0  
codRespuesta: 000  
idAutorizacion: 3

[Volver al comercio](#)

A continuación comprobamos que se ha sustraído el valor correctamente



alumnodb@visa.10.1.2.1:5432 [8.4.10] SQL Editor - \*Untitled

```

1 select * from tarjeta where numerotarjeta = '2347 4840 5058 7931';
2
3

```

Result Execution plan Visualize Logging

#	numerotarjeta	titular	validadesde	validahasta	codigoverificacion	saldo
1	2347 4840 5058 7931	Gabriel Avila Locke	11/09	01/20	207	500

Y vemos que el pago se ha registrado correctamente:

alumnodb@visa.10.1.2.1:5432 [8.4.10] SQL Editor - \*Untitled

```

8
9
10 select * from pago where idautorizacion=3

```

Result Execution plan Visualize Logging

#	idautorizacion	idtransaccion	codrespuesta	importe	idcomercio	numerotarjeta	fecha
1	3	3	000	500	3	2347 4840 5058 7931	02/03/18 03:17

**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.**

A continuación tratamos de realizar el pago, pero nos da incorrecto:

← → ↻ 10.1.2.2:8080/P1-ejb-cliente/procesapago

## Pago con tarjeta

### Pago incorrecto

#### Prácticas de Sistemas Informáticos II

Además el pago no ha sido sustraído de la tarjeta, como vemos aquí:

alumnodb@visa.10.1.2.1:5432 [8.4.10] SQL Editor - \*Untitled

```

8
9
10 select * from pago where idautorizacion=3

```

Result Execution plan Visualize Logging

#	idautorizacion	idtransaccion	codrespuesta	importe	idcomercio	numerotarjeta	fecha
1	3	3	000	500	3	2347 4840 5058 7931	02/03/18 03:17

### 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.

Rellenamos los campos del formulario tal y como aparece en la figura 4 para crear la factoría de conexiones.

### **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

Rellenamos los campos del formulario tal y como aparece en la figura 5 para crear la cola de mensajes.

### **Ejercicio número 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.**

Para empezar, hemos modificado el xml tal y como se nos indicaba en el enunciado de la práctica, quedando el contenido de la etiqueta <ejb> de la siguiente manera:

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

En concreto, la etiqueta <mdb-conneciton-factory> es la que indicará al sistema el nombre de la factoría de conexiones (con la etiqueta <jndi-name> que hay dentro)

Continuando con el ejercicio, hemos realizado los siguientes dos prepared statements dentro del fichero VisaCancelacionJMSBean.java:

- Para actualizar el código de respuesta a 999:  
*update pago set codRespuesta = 999 where idAutorizacion=?*
- Para devolver el importe del pago a la tarjeta:  
*update tarjeta as t set saldo = saldo + p.importe from pago as p where p.idAutorizacion = ? and p.numeroTarjeta = t.numeroTarjeta*

Después, en el método onMessage, hemos ejecutado estas consultas a la base de datos (realizando una conexión, y ejecución análoga a la realizada en ejercicios anteriores) siempre rodeadas por un try catch que, en caso de que se produjera algún error, haría rollback de esta operación (gracias a la excepción *EJBException*)

### **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 método de acceso estático presenta el inconveniente de que se debe conocer el nombre del recurso en tiempo de compilación, lo cual implica que si se quisiera cambiar este recurso, se tendría que compilar el código de nuevo.

Por otra parte, el método de acceso dinámico permite, modificando un archivo, cambiar el recurso al que se accede sin necesidad de compilar el código. De hecho podría incluso hacerse de forma remota con ssh. Esto permite mayor flexibilidad a la aplicación para, por ejemplo, en caso de que la cola estuviera funcionando incorrectamente o algo similar, arreglar el problema realizando una tarea sencilla.

Sin embargo, el hecho de tener que realizar la conexión de forma dinámica puede estar sujeto a errores que hay que controlar mediante el uso de excepciones en el código. Además, la conexión en sí puede resultar costosa (en cuanto a tiempo) si se realiza de forma dinámica.

En cambio, si se realiza en tiempo de compilación, gracias a las anotaciones, Java podría realizar optimizaciones oportunas para agilizar este proceso.

### ***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.**

El comando equivalente para crear una cola jms usando asadmin es:

```
asadmin --user admin --passwordfile passwordfile --host 10.1.2.2 --port 4848 create-jms-resource
--restype javax.jms.QueueConnectionFactory --enabled=true --property Name=Visa
jms/VisaPagosQueue
```

Tras eliminar manualmente la connectionFactory y las colas creadas desde la consola de administración de glassfish y cambiar el jms.xml para que en las reglas *create-jms-resource* y *create-jms-connection-factory* para esto hemos añadido las siguientes etiquetas <exec>

Para la creación de colas: <exec executable="{asadmin}">

```
<arg line="--user ${as.user}" />
<arg line="--passwordfile ${as.passwordfile}" />
<arg line="--host ${as.host.server}" />
<arg line="--port ${as.port}" />
<arg line="create-jms-resource"/>
<arg line="--restype ${jms.restype}" />
<arg line="--enabled=true" />
<arg line="--property ${jms.resource.property}" />
<arg line="${jms.resource.name}" />
```

</exec>

Para la creación de factorías:

```
<exec executable="{asadmin}">
  <arg line="--user ${as.user}" />
  <arg line="--passwordfile ${as.passwordfile}" />
  <arg line="--host ${as.host.server}" />
  <arg line="--port ${as.port}" />
  <arg line="create-jms-resource"/>
  <arg line="--restype ${jms.restype}" />
  <arg line="--enabled=true" />
  <arg line="${jms.resource.name}" />
</exec>

<exec executable="{asadmin}">
  <arg line="--user ${as.user}" />
  <arg line="--passwordfile ${as.passwordfile}" />
  <arg line="--host ${as.host.server}" />
  <arg line="--port ${as.port}" />
  <arg line="create-jms-resource"/>
  <arg line="--restype ${jms.restype}" />
  <arg line="--enabled=true" />
  <arg line="--property ${jms.resource.property}" />
  <arg line="${jms.resource.name}" />
</exec>
```

Para la creación de factorías:

```
<exec executable="{asadmin}">
  <arg line="--user ${as.user}" />
  <arg line="--passwordfile ${as.passwordfile}" />
  <arg line="--host ${as.host.server}" />
  <arg line="--port ${as.port}" />
  <arg line="create-jms-resource"/>
  <arg line="--restype ${jms.restype}" />
  <arg line="--enabled=true" />
  <arg line="${jms.resource.name}" />
</exec>
```

(nótese que la etiqueta exec para la creación de colas expuesta es equivalente al comando ya escrito)

Tras realizar estos cambios, hemos ejecutado la regla *todo* y comprobado, en la terminal de administración de glassfish, que efectivamente los recursos se han creado correctamente y con los nombres pertinentes (que hemos especificado en el fichero jms.properties)

## 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-4.1.1/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-4.1.1/glassfish/bin/appclient -targetserver 10.X.Y.Z -client  
dist/clientjms/P1-jms-clientjms.jar -browse
```

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

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

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

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

Desde PC1 host:

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

Desde la máquina virtual 10.X.Y.1:

```
si2@si2srv01:~$ export JAVA_HOME=/usr/lib/jvm/java-8-oracle/  
si2@si2srv01:~$ /opt/glassfish4/glassfish/bin/appclient -targetserver 10.X.Y.2 -client  
/tmp/P1-jms-clientjms.jar
```

Hemos ejecutado ambos comandos desde la máquina virtual del cliente y hemos obtenido la siguiente salida (con el enabled de P1-jms-mdb a false):

```
si2@si2srv01:~$ /opt/glassfish4/glassfish/bin/appclient -targetserver 10.1.2.2 -client /tmp/P1-jms-clientjms.jar 1  
Mar 06, 2018 9:24:50 AM org.hibernate.validator.internal.util.Version <clinit>  
INFO: HV000001: Hibernate Validator 5.0.0.Final  
Mar 06, 2018 9:24:50 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 06, 2018 9:24:50 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 06, 2018 9:24:50 AM com.sun.messaging.jms.ra.ResourceAdapter start  
INFO: MQJMSRA RA1101: GlassFish MQ JMS Resource Adapter Started:REMOTE  
mensaje enviado : 1  
si2@si2srv01:~$ /opt/glassfish4/glassfish/bin/appclient -targetserver 10.1.2.2 -client /tmp/P1-jms-clientjms.jar -browse  
Mar 06, 2018 9:25:04 AM org.hibernate.validator.internal.util.Version <clinit>  
INFO: HV000001: Hibernate Validator 5.0.0.Final  
Mar 06, 2018 9:25:04 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 06, 2018 9:25:04 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 06, 2018 9:25:04 AM com.sun.messaging.jms.ra.ResourceAdapter start  
INFO: MQJMSRA RA1101: GlassFish MQ JMS Resource Adapter Started:REMOTE  
Mensajes en cola:  
1  
-
```

El primer comando envía un 1 a la cola de mensajes.

El segundo comando mira el contenido de la cola y encuentra dicho 1 enviado (pues al estar deshabilitado el enabled anterior, no se procesa el mensaje de la cola).

A continuación, realizamos un pago desde la web:



← → ↻ 10.1.2.2:8080/P1-ejb-cliente/comienzapago

## Pago con tarjeta

Numero de visa:	2347 4840 5058 7931
Titular:	Gabriel Avila Locke
Fecha Emisión:	11/09
Fecha Caducidad:	01/20
CVV2:	207
<input type="button" value="Pagar"/>	

Id Transacción: 1  
Id Comercion: 1  
Importe: 100.0

Con resultado:

← → ↻ 10.1.2.2:8080/P1-ejb-cliente/procesapago

## Pago con tarjeta

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

idTransaccion: 1  
idComercio: 1  
importe: 100.0  
codRespuesta: 000  
idAutorizacion: 1

[Volver al comercio](#)

Comprobamos en Tora que el pago ha surtido efecto:

```
3 select idautorizacion, saldo, codrespuesta, importe from tarjeta natural join pago
```

Result Execution plan Visualize Logging					
#	^	idautorizacion	saldo	codrespuesta	importe
1	1		900	000	100

Y habilitamos el enable en P1-jms-mdb de manera que el mensaje de la cola se procesará. Volvemos a ver el contenido de la cola ejecutando de nuevo el comando de antes y obtenemos el siguiente resultado: <exec executable="{asadmin}">

<arg line="--user \${as.user}" />

<arg line="--passwordfile \${as.passwordfile}" />

```

<arg line="--host ${as.host.server}" />
<arg line="--port ${as.port}" />
<arg line="create-jms-resource"/>
<arg line="--restype ${jms.restype}" />
<arg line="--enabled=true" />
<arg line="--property ${jms.resource.property}" />
<arg line="${jms.resource.name}" />
</exec>

```

Para la creación de factorías:

```

<exec executable="${asadmin}">
  <arg line="--user ${as.user}" />
  <arg line="--passwordfile ${as.passwordfile}" />
  <arg line="--host ${as.host.server}" />
  <arg line="--port ${as.port}" />
  <arg line="create-jms-resource"/>
  <arg line="--restype ${jms.restype}" />
  <arg line="--enabled=true" />
  <arg line="${jms.resource.name}" />
</exec>

```

```

si2@si2srv01:~$ /opt/glassfish4/glassfish/bin/appclient -targetserver 10.1.2.2 -client /tmp/P1-jms-clientjms.jar -browse
Mar 06, 2018 9:31:57 AM org.hibernate.validator.internal.util.Version <clinit>
INFO: HV000001: Hibernate Validator 5.0.0.Final
Mar 06, 2018 9:31:57 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 06, 2018 9:31:57 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 06, 2018 9:31:57 AM com.sun.messaging.jms.ra.ResourceAdapter start
INFO: MQJMSRA RA1101: GlassFish MQ JMS Resource Adapter Started:REMOTE
Cola de mensajes vac?a!

```

Como se ha procesado el mensaje de la cola. Comprobamos que la cancelación del pago ha surtido efecto:

3 `select idautorizacion, saldo, codrespuesta, importe from tarjeta natural join pago`

#	^	idautorizacion	saldo	codrespuesta	importe
1	1		1000	999	100

Como podemos comprobar, se ha devuelto el importe al saldo de la tarjeta y se ha escrito un código 999 en el campo codRespuesta.