

	Escuela Politécnica Superior Ingeniería Informática Prácticas de Sistemas Informáticos 2			
Grupo	2362	Práctica	1B	Fecha
Alumno/a	Ramos Pedroviejo, Alba			
Alumno/a	Serrano Salas, Nicolás			

Práctica 1A: Arquitectura de Java EE

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. ¿Para qué se utilizan?

La librería más llamativa que se importa es javax.ejb.Local, que hace accesible el uso de la anotación @Local. Dicha anotación significa que el cliente y el servidor se van a ejecutar en el mismo servidor, que es lo que se va a realizar en esta primera parte de la práctica. En definitiva, que la interfaz se va a implementar localmente.

Ejercicio 1:

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

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

...

```
import javax.ejb.Stateless;
```

...

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

...

- Eliminar el constructor por defecto de la clase.

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

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

Hemos importado javax.ejb.Stateless y hemos anotado la clase como Stateless, haciendo que implemente la interfaz de la cuestión anterior. Además, se ha eliminado el constructor por defecto.

```

28 import javax.ejb.Stateless;
29
30 /**
31 * @author jaime
32 */
33 @Stateless(mappedName="VisaDAOBean")
34 public class VisaDAOBean extends DBTester implements VisaDAOLocal{
35     private boolean debug = false;

```

Se ha modificado la función getPagos para que devuelva un array de PagoBean en vez de un ArrayList, como se hacía originalmente en P1-base.

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.

En el servlet ProcesaPago se han añadido los imports especificados y se ha añadido el objeto proxy con atributo de la clase, anotando como se nos pide. Finalmente, hemos eliminado la declaración antigua del servicio web antiguo como se nos especifica y hemos eliminado también todos los usos de BindingProvider (se usaba en la función processRequest), así como los imports del mismo.

```

protected void processRequest(HttpServletRequest request, HttpServletResponse response
throws ServletException, IOException {
    TarjetaBean tarjeta = creaTarjeta(request);
    ValidadorTarjeta val = new ValidadorTarjeta();
    PagoBean pago = null;

    // printAddresses(request,response);
    if (!val.esValida(tarjeta)) {
        request.setAttribute(val.getErrorName(), val.getErrorVisa());
        reenvia("/formdatosvisa.jsp", request, response);
        return;
    }

    HttpSession sesion = request.getSession(false);
    if (sesion != null) {
        pago = (PagoBean) sesion.getAttribute(ComienzaPago.ATTR_PAGO);
    }
    if (pago == null) {
        pago = creaPago(request);
        boolean isdebug = Boolean.valueOf(request.getParameter("debug"));
        dao.setDebug(isdebug);
        boolean isdirectConnection = Boolean.valueOf(request.getParameter("directConnection"));
        dao.setDirectConnection(isdirectConnection);
        boolean usePrepared = Boolean.valueOf(request.getParameter("usePrepared"));
        dao.setPrepared(usePrepared);
    }

    // Almacenamos la tarjeta en el pago
    pago.setTarjeta(tarjeta);

    if (!dao.compruebaTarjeta(tarjeta)) {
        enviaError(new Exception("Tarjeta no autorizada:"), request, response);
        return;
    }

    if (dao.realizaPago(pago) == null) {
        enviaError(new Exception("Pago incorrecto"), request, response);
        return;
    }
}

```

En el servlet GetPagos se han añadido los imports especificados y se ha añadido el objeto proxy con atributo de la clase, anotando como se nos pide. Finalmente, hemos modificado el método processRequest para que, cuando llame a getPagos, obtenga un array de PagoBean, en vez del ArrayList de la práctica anterior.

```

protected void processRequest(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    PagoBean[] pagos;

    /* Se recoge de la petici&on el par&aacute;metro idComercio*/
    String idComercio = request.getParameter(PARAM_ID_COMERCIO);

    /* Petici&on de los pagos para el comercio */

    pagos = dao.getPagos(idComercio);

    request.setAttribute(ATTR_PAGOS, pagos);
    reenvia("/listapagos.jsp", request, response);
    return;
}

```

En el servlet DelPagos se han añadido los imports especificados y se ha añadido el objeto proxy con atributo de la clase, anotando como se nos pide. Finalmente, hemos modificado el método processRequest para eliminar las referencias al servicio antiguo y a BindingProvider.

```

protected void processRequest(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
    /* Se recoge de la petici&on el par&aacute;metro idComercio*/
    String idComercio = request.getParameter(PARAM_ID_COMERCIO);

    /* Petici&on de los pagos para el comercio */
    int ret = dao.delPagos(idComercio);

    if (ret != 0) {
        request.setAttribute(ATTR_BORRADOS, ret);
        reenvia("/borradook.jsp", request, response);
    } else {
        reenvia("/borradoerror.jsp", request, response);
    }
    return;
}

```

Cuestión número 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 fichero application.xml contiene los nombres de las direcciones de los ficheros que hemos generado al empaquetar tanto el cliente como el servidor, describiendo así cada uno de los módulos de 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>
```

Si observamos el contenido de los .jar, .war y .ear, observamos que se muestran los ficheros que contienen cada uno, con su tamaño, última modificación y nombre del fichero. Observamos que cada uno contiene todos los módulos necesarios para el empaquetado de cada uno de los componentes.

```
e380143@11-26-11-26:~/S12/practica1b/P1-ejb$ jar -tvf dist/P1-ejb.ear
 0 Wed Feb 26 19:50:40 CET 2020 META-INF/
105 Wed Feb 26 19:50:38 CET 2020 META-INF/MANIFEST.MF
508 Sat Feb 11 23:33:00 CET 2012 META-INF/application.xml
20942 Wed Feb 26 19:49:46 CET 2020 P1-ejb-cliente.war
7004 Wed Feb 26 19:25:34 CET 2020 P1-ejb.jar
e380143@11-26-11-26:~/S12/practica1b/P1-ejb$ jar -tvf dist/client/P1-ejb-cliente.war
 0 Wed Feb 26 19:49:46 CET 2020 META-INF/
105 Wed Feb 26 19:49:44 CET 2020 META-INF/MANIFEST.MF
 0 Wed Feb 26 19:49:46 CET 2020 WEB-INF/
 0 Wed Feb 26 19:48:44 CET 2020 WEB-INF/classes/
 0 Wed Feb 26 19:48:44 CET 2020 WEB-INF/classes/ssii2/
 0 Wed Feb 26 19:48:44 CET 2020 WEB-INF/classes/ssii2/controlador/
 0 Wed Feb 26 19:48:44 CET 2020 WEB-INF/classes/ssii2/filtros/
 0 Wed Feb 26 19:48:44 CET 2020 WEB-INF/classes/ssii2/visa/
 0 Wed Feb 26 19:48:44 CET 2020 WEB-INF/classes/ssii2/visa/error/
 0 Wed Feb 26 19:22:08 CET 2020 WEB-INF/lib/
 0 Wed Feb 26 19:49:46 CET 2020 error/
2844 Wed Feb 26 19:48:44 CET 2020 WEB-INF/classes/ssii2/controlador/ComienzaPago.class
1513 Wed Feb 26 19:48:44 CET 2020 WEB-INF/classes/ssii2/controlador/IDPagos.class
1365 Wed Feb 26 19:48:44 CET 2020 WEB-INF/classes/ssii2/controlador/GetPagos.class
4915 Wed Feb 26 19:48:44 CET 2020 WEB-INF/classes/ssii2/controlador/ProcesaPago.class
1894 Wed Feb 26 19:48:44 CET 2020 WEB-INF/classes/ssii2/controlador/ServLetRaiz.class
2608 Wed Feb 26 19:48:44 CET 2020 WEB-INF/classes/ssii2/filtros/CompruebaSesion.class
3170 Wed Feb 26 19:48:44 CET 2020 WEB-INF/classes/ssii2/visa/ValidadorTarjeta.class
616 Wed Feb 26 19:48:44 CET 2020 WEB-INF/classes/ssii2/visa/error/ErrorVisa.class
198 Wed Feb 26 19:48:44 CET 2020 WEB-INF/classes/ssii2/visa/error/ErrorVisaCV.class
209 Wed Feb 26 19:48:44 CET 2020 WEB-INF/classes/ssii2/visa/error/ErrorVisaFechaAdicidad.class
207 Wed Feb 26 19:48:44 CET 2020 WEB-INF/classes/ssii2/visa/error/ErrorVisaFechaEmision.class
201 Wed Feb 26 19:48:44 CET 2020 WEB-INF/classes/ssii2/visa/error/ErrorVisaNumero.class
202 Wed Feb 26 19:48:44 CET 2020 WEB-INF/classes/ssii2/visa/error/ErrorVisaTitular.class
6032 Wed Feb 26 19:49:46 CET 2020 WEB-INF/web.xml
455 Wed Feb 26 19:49:46 CET 2020 borradoerror.jsp
501 Wed Feb 26 19:49:46 CET 2020 borrado.jsp
509 Wed Feb 26 19:49:46 CET 2020 cabecera.jsp
283 Wed Feb 26 19:49:46 CET 2020 error/mestraerror.jsp
2729 Wed Feb 26 19:49:46 CET 2020 formdatosvisa.jsp
1259 Wed Feb 26 19:49:46 CET 2020 listapagos.jsp
1178 Wed Feb 26 19:49:46 CET 2020 pago.html
1142 Wed Feb 26 19:49:46 CET 2020 pagoxito.jsp
104 Wed Feb 26 19:49:46 CET 2020 pie.html
5011 Wed Feb 26 19:49:46 CET 2020 testbd.jsp
e380143@11-26-11-26:~/S12/practica1b/P1-ejb$ jar -tvf dist/server/P1-ejb.jar
 0 Wed Feb 26 19:25:34 CET 2020 META-INF/
105 Wed Feb 26 19:25:32 CET 2020 META-INF/MANIFEST.MF
 0 Wed Feb 26 19:24:48 CET 2020 ssii2/
 0 Wed Feb 26 19:24:48 CET 2020 ssii2/visa/
 0 Wed Feb 26 19:24:48 CET 2020 ssii2/visa/dao/
255 Wed Feb 26 19:25:34 CET 2020 META-INF/sun-ejb-jar.xml
1464 Wed Feb 26 19:24:48 CET 2020 ssii2/visa/PagoBean.class
856 Wed Feb 26 19:24:48 CET 2020 ssii2/visa/TarjetaBean.class
593 Wed Feb 26 19:24:48 CET 2020 ssii2/visa/VisaDAOLocal.class
1723 Wed Feb 26 19:24:48 CET 2020 ssii2/visa/dao/DBTester.class
7042 Wed Feb 26 19:24:48 CET 2020 ssii2/visa/dao/VisaDAOBean.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 por qué son esos valores.

Se ha establecido para ambas propiedades la dirección 10.8.6.2, porque tanto el cliente como el servidor se van a ejecutar en la misma máquina.

- 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 por qué son esos valores.**

Se ha establecido para db.client.host la dirección 10.8.6.1 y para db.host la dirección 10.8.6.2. Esto se debe a que el cliente de la base de datos estará en la misma máquina que hemos especificado anteriormente, pero la base de datos está en otra máquina distinta, es independiente.

Desplegar la aplicación de empresa ant desplegar

Además de los pasos anteriores, hemos desplegado la base de datos.

Podemos comprobar mediante la consola de Glassfish que nuestra aplicación se despliega correctamente como Enterprise Application.

The screenshot shows the GlassFish Administration Console interface. On the left, there's a navigation tree with options like Domain, Clusters, Standalone Instances, Nodes, Applications (with P1-ejb selected), Lifecycle Modules, Monitoring Data, Resources, Configurations, and Update Tool. The main panel has tabs for General and Descriptor. Under General, the application is named 'P1-ejb', status is 'Enabled' (checked), and it's associated with the 'server' virtual server. Other settings include Implicit CDI (Enabled), Java Web Start (Enabled), Location (\$com.sun.aas.instanceRootURL/applications/P1-ejb/), Deployment Order (100), Libraries, and Description. Below this is a table titled 'Modules and Components (9)' with columns for Module Name, Engines, Component Name, Type, and Action. The table lists several modules and their components, such as P1-ejb-cliente.war (engines: [web], components: default, jsp, DelPagos, ProcesaPago, GetPagos, ComienzaPago), P1-ejb.jar (engines: [ejb, weld], component: VisaDAOBean), and P1-ejb-cliente.war (engines: [ejb, weld], component: StatelessSessionBean).

Module Name	Engines	Component Name	Type	Action
P1-ejb-cliente.war	[web]	-----	-----	Launch
P1-ejb-cliente.war		default	Servlet	
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-ejb.jar		VisaDAOBean	StatelessSessionBean	

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.

Hemos comprobado que, si realizamos un pago mediante pago.html, nos lleva a comienzapago correctamente, donde introducimos los datos y se realiza adecuadamente. Por ejemplo, si realizamos un pago con id de transacción 13 para el comercio 5, se realiza correctamente.

Hemos comprobado que podemos realizar pagos correctos también mediante testbd.jsp, sin usar conexión directa. Podemos ver que el pago se realiza correctamente, por ejemplo un pago con id 2 y otro con id 4.

Podemos comprobar que la base de datos contiene los pagos descritos anteriormente:

	idautorizacion [PK] serial	idtransaccion character(16)	codrespuesta character(3)	importe double precision	idcomercio character(16)	numerotarjeta character(19)	fecha timestamp without time zone
1	1	2	000	10	1	1111 2222 3333 4444	2020-03-04 09:38:09.974544
2	2	4	000	50	1	1111 2222 3333 4444	2020-03-04 09:41:19.385737
3	3	13	000	10	5	1111 2222 3333 4444	2020-03-04 10:09:11.160351
*							

Si probamos a listar los pagos del comercio 5 desde testbd, obtenemos la misma información que hay en la base de datos, que es el pago con id de transacción 13:

Pago con tarjeta

Lista de pagos del comercio 5

idTransaccion	Importe	codRespuesta	idAutorizacion
13	10.0	000	3

[Volver al comercio](#)

Prácticas de Sistemas Informáticos II

Además, si probamos a eliminar todos los pagos del comercio 1, nos informa de que se han borrado los dos pagos realizados anteriormente, y podemos comprobar que en la base de datos se eliminan adecuadamente:

Pago con tarjeta

Se han borrado 2 pagos correctamente para el comercio 1

[Volver al comercio](#)

Prácticas de Sistemas Informáticos II

	idautorizacion [PK] serial	idtransaccion character(16)	codrespuesta character(3)	importe double precision	idcomercio character(16)	numerotarjeta character(19)	fecha timestamp without time zone
1	3	13	000	10	5	1111 2222 3333 4444	2020-03-04 10:09:11.160351
*							

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

Hemos añadido import java.io.Serializable tanto en PagoBean.java como en TarjetaBean.java, y hemos hecho que ambas implementen la interfaz Serializable.

```
import java.io.Serializable;

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

Además, hemos definido la interfaz remota del EJB en VisaDAORemote.java, que es una copia de VisaDAOLocal.java pero en remoto en vez de local.

```
import javax.ejb.Remote;

@Remote
public interface VisaDAORemote {
    public boolean comprouebaTarjeta(TarjetaBean tarjeta);
    public PagoBean realizaPago(PagoBean pago);
    public PagoBean[] getPagos(String idComercio);
    public int delPagos(String idComercio);
    public boolean isDebugEnabled();
    public boolean isPrepared();
    public void setPrepared(boolean prepared);
    public void setDebugEnabled(boolean debug);
    public int getDirectConnectionCount();
    public int getDSNConnectionCount();
    public boolean isDirectConnection();
    public void setDirectConnection(boolean directConnection);
}
```

Finalmente, hemos hecho que VisaDAOBean implemente tanto la interfaz local como la remota.

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

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.

Hemos añadido import java.io.Serializable tanto en PagoBean.java como en TarjetaBean.java, y hemos hecho que ambas implementen la interfaz Serializable, al igual que se hizo en el ejercicio anterior. A continuación, hemos eliminado la declaración de VisaDAO dao en los 3 servlets del cliente. Tras importar EJB y VisaDAORemote, hemos declarado de nuevo la variable dao pero utilizando la interfaz que hemos declarado. Además, en processRequest de ProcesaPago.java hemos tenido que modificar la comprobación !dao.realizaPago(pago) por dao.realizaPago(pago) == null. Finalmente, hemos creado el fichero glassfish-web.xml y hemos indicado la dirección 10.8.6.2 para indicar que ahí va a estar el servidor, y hemos modificado los ficheros build.properties y postgresql.properties para indicar que ahora estarán en la dirección 10.8.6.1, que es la del cliente.

Si probamos a lanzar la aplicación del cliente y realizar un pago con id 17 por ejemplo, podemos ver que se realiza correctamente y en la base de datos también se almacena correctamente (como no hemos replegado la base desde los ejercicios anteriores, también está uno de los pagos de un apartado anterior):



Pago con tarjeta

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

idTransaccion: 17
idComercio: 8
importe: 12.0
codRespuesta:
idAutorizacion:

[Volver al comercio](#)

Prácticas de Sistemas Informáticos II

	idautorizacion [PK] serial	idtransaccion character(16)	codrespuesta character(3)	importe double precision	idcomercio character(16)	numerotarjeta character(19)	fecha timestamp without time zone
1	3	13	000	10	5	1111 2222 3333 4444	2020-03-04 10:09:11.160351
2	4	17	000	12	8	1111 2222 3333 4444	2020-03-04 10:55:22.006301
*							

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;

Hemos creado dicho atributo y sus métodos setter y getter:

```
private String numero;
private String titular;
private String fechaEmision;
private String fechaCaducidad;
private String codigoVerificacion; /* CVV2 */
private double saldo;

/**
 * Devuelve el saldo de la tarjeta
 * @return saldo de la tarjeta
 */
public double getSaldo() {
    return saldo;
}

/**
 * Establece el saldo de la tarjeta
 * @param saldo
 */
public void setSaldo(double saldo) {
    this.saldo = saldo;
}
```

Archivo VisaDAOBean.java:

- Importar la definición de la excepción EJBException que debe lanzar el servlet para indicar que se debe realizar un rollback: import javax.ejb.EJBException;
- Declarar un prepared statement para recuperar el saldo de 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:
 - o Recuperar el saldo de la tarjeta a través del prepared statement declarado anteriormente.
 - o 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)
 - o 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.
 - o 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).
 - o 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.

Hemos importado la excepción EJBException y hemos creado dos consultas que utilizan prepared statement, una para obtener el saldo asociado a una tarjeta y otra para actualizarlo:

```
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 tarjeta.numeroTarjeta=?";
*****
```

A continuación, hemos modificado el método realizaPago. A la funcionalidad que ya había hemos añadido un try-catch para comprobar el saldo. Tras abrir una nueva conexión, preparamos el prepared statement pasándole como parámetro el número de tarjeta de la que comprobar el saldo. Como es una consulta que retorna un resultado, lo guardaremos en rs y comprobaremos su valor. Si su valor es mayor que el importe del pago entonces procedemos a actualizar el saldo de la tarjeta utilizando la otra consulta creada.

```
try {
    // Obtener conexión
    con = getConnection();

    String getSaldo = SELECT_SALDO_QRY;
    errorLog(getSaldo);
    pstmt = con.prepareStatement(getSaldo);
    pstmt.setString(1, pago.getTarjeta().getNumero());

    rs = pstmt.executeQuery(getSaldo);

    if (rs.next()) {
        double saldo = rs.getDouble("saldo");

        if (saldo >= pago.getImporte()){
            String insert = UPDATE_SALDO_QRY;
            errorLog(insert);
            pstmt = con.prepareStatement(insert);
            pstmt.setDouble(1, saldo-pago.getImporte());
            pstmt.setString(2, pago.getTarjeta().getNumero());
            pstmt.executeUpdate();
            ret = true;
        }
    }
}
```

En caso de que no sea mayor, se devuelve nulo y se establece el id de autorización también a nulo. Se tratan las excepciones en caso de que haya ocurrido alguna. Finalmente, si ha habido algún problema se lanzará EJBException, y sino se continuará registrando el pago en la base de datos. Si durante ese proceso ocurriese otro problema también lanzaremos esta excepción.

```

        ret = false;
        if (!pstmt.execute()
            && pstmt.getUpdateCount() == 1) {
            ret = true;
        }
    } else {
        pago.setIdAutorizacion(null);
        return null;
    }
} else {
    ret = false;
}

} catch (Exception e) {
    errorLog(e.toString());
    ret = false;
} finally {
    try {
        if (rs != null) {
            rs.close(); rs = null;
        }
        if (stmt != null) {
            stmt.close(); stmt = null;
        }
        if (pstmt != null) {
            pstmt.close(); pstmt = null;
        }
    } catch (SQLException e) {
        pstmt.close(); pstmt = null;
        if (con != null) {
            closeConnection(con); con = null;
        }
    } catch (EJBException e) {
        if(!ret){
            throw new EJBException();
        }
    }
}
// Registrar el pago en la base de datos
try {
    // Obtener conexion
    con = getConnection();
    // Insertar en la base de datos el pago
}

```

- 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 importado la excepción EJBException y hemos añadido un try-catch en el if donde se llama a dao.realizaPago. En el catch invalidamos la sesión si existe y devolvemos la página de error.

```

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

```

Ejercicio 8:

Desplegar y probar la nueva aplicación creada.

Observamos la tarjeta de nuestro usuario de pruebas y vemos su saldo con pgAdmin, el cual es el que se establece por defecto en insert.sql:

Data Output						
	numerotarjeta character(19)	titular character varying(128)	validadesde character(5)	validahasta character(5)	codigoverificacion character(3)	saldo double precision
1	1111 2222 3333 4444	Jose Garcia	11/09	11/20	123	1000

- 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.

Si probamos a realizar un pago de 60 euros en el comercio 1 con id de transacción 19, vemos que efectivamente se actualiza el valor de la base de datos:



Pago con tarjeta

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

```

idTransaccion: 19
idComercio: 1
importe: 60.0
codRespuesta: 000
idAutorizacion: 1

```

[Volver al comercio](#)

Prácticas de Sistemas Informáticos II

Data Output						
	numerotarjeta character(19)	titular character varying(128)	validadesde character(5)	validahasta character(5)	codigoverificacion character(3)	saldo double precision
1	1111 2222 3333 4444	Jose Garcia	11/09	11/20	123	940

- 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.

Si probamos otra vez a realizar un pago de 40 euros al comercio 1 con id de transacción 19, vemos que esta vez nos sale la ventana de enviarError producido por la excepción EJBException y la base de datos no se ha modificado.



Pago con tarjeta

Prácticas de Sistemas Informáticos II

Data Output						
	numerotarjeta character(19)	titular character varying(128)	validadesde character(5)	validahasta character(5)	codigoverificacion character(3)	saldo double precision
1	1111 2222 3333 4444	Jose Garcia	11/09	11/20	123	940

- 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.

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.

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 pagos
Status:	<input checked="" type="checkbox"/> 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:	<input type="checkbox"/> Close All Connections Close all connections and disconnect on failure, otherwise reconnect automatically if available.		

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.

New JMS Destination Resource

The creation of a new Java Message Service (JMS) destination resource also creates an admin object resource.

Additional Properties (0)

Add Property	Delete Properties		
Select	Name	Value	Description

No items found.

Ejercicio 11:

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

```
practica1b > P1-jms > conf > mdb > META-INF > sun-ejb-jar.xml
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!DOCTYPE sun-ejb-jar PUBLIC "-//Sun Microsystems, Inc.//DTD Application Server 9.0 EJB 3.0//EN" "http://
3  <sun-ejb-jar>
4      <enterprise-beans>
5          <ejb>
6              <ejb-name>VisaCancelacionJMSBean</ejb-name>
7              <mdb-connection-factory>
8                  <jndi-name>jms/VisaConnectionFactory</jndi-name>
9                  </mdb-connection-factory>
10             </ejb>
11         </enterprise-beans>
12     </sun-ejb-jar>
```

•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 coincide con lo recibido por el mensaje.

```
static final String UPDATE_CANCELAR_QRY = "update pago set codRespuesta=999 where idautorizacion=?";  
static final String UPDATE_SALDO = "update tarjeta set saldo=? where idautorizacion=?";
```

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

Como en esta ocasión no tenemos acceso al pago realizado, mediante el id de autorización podremos conocer el número de tarjeta y el importe de un pago, con el objetivo de restaurar el saldo, devolviendo el importe gastado a esa tarjeta.

```
static final String UPDATE_SALDO = "update tarjeta" +  
    "set saldo=saldo+" +  
    "select importe" +  
    "from pago" +  
    "where idautorizacion=?"+  
    ")" +  
    "where numerotarjeta=" +  
    "select numerotarjeta" +  
    "from pago" +  
    "where idautorizacion=?"+  
    ")";
```

• **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.**

Como el mensaje contiene el id de la autorización en formato string, basta pasarlo como parámetro de los dos prepared statements que harán la actualización. El código será el mismo que en ejercicios anteriores:

```
try {  
    if (inMessage instanceof TextMessage) {  
        msg = (TextMessage) inMessage;  
        logger.info("MESSAGE BEAN: Message received: " + msg.getText());  
  
        int idAutorizacion = Integer.parseInt(msg.getText());  
  
        con = getConnection();  
  
        String updResponse = UPDATE_CANCELAR_QRY;  
        logger.info(updResponse);  
        pstmt = con.prepareStatement(updResponse);  
        pstmt.setInt(1, idAutorizacion);  
        ret = false;  
        if (!pstmt.execute()  
            && pstmt.getUpdateCount() == 1) {  
            ret = true;  
        }  
  
        if (ret != false){  
            String updSaldo = UPDATE_SALDO;  
            logger.info(updSaldo);  
            pstmt = con.prepareStatement(updSaldo);  
            pstmt.setInt(1, idAutorizacion);  
            pstmt.setInt(2, idAutorizacion);  
            ret = false;  
            if (!pstmt.execute()  
                && pstmt.getUpdateCount() == 1) {  
                ret = true;  
            }  
        }  
    }  
}
```

• **Control de errores en el método onMessage.**

El método no devuelve nada, sin embargo es necesario manejar correctamente las excepciones que podrían darse al conectar con la base de datos. Además, si nos da un error al modificar el código de respuesta no seguiremos actualizando el saldo para no dejar la base de datos en un estado inconsistente.

```

        }
    } else {
        logger.warning(
            "Message of wrong type: "
            + inMessage.getClass().getName());
    }
} catch (JMSEException e) {
    e.printStackTrace();
    mdc.setRollbackOnly();
} catch (Throwable te) {
    te.printStackTrace();
} finally {
    try {
        if (pst != null) {
            pst.close(); pst = null;
        }
        if (con != null) {
            closeConnection(con); con = null;
        }
    } catch (SQLException e) {
    }
}

```

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

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.

El método de acceso por la clase InitialContext lo hemos dejado comentado

```

public class VisaQueueMessageProducer {

    // TODO: Anotar los siguientes objetos para
    // conectar con la connection factory y con la cola
    // definidas en el enunciado
    @Resource(mappedName = "jms/VisaConnectionFactory")
    private static ConnectionFactory connectionFactory;
    @Resource(mappedName = "jms/VisaPagosQueue")
    private static Queue queue;

    // TODO: Inicializar connectionFactory
    // y queue mediante JNDI
    /*
    InitialContext jndi = new InitialContext();
    connectionFactory = (ConnectionFactory)jndi.lookup("jms/VisaConnectionFactory");
    queue = (Queue)jndi.lookup("jms/VisaPagosQueue");
    */
}

```

Ejercicio 13:

Automatice la creación de los recursos JMS (cola y factoría de conexiones) en el build.xml y jms.xml. Para ello, indique en jms.properties los nombres de ambos y el Physical Destination Name de la cola de acuerdo a los valores asignados en los ejercicios 7 y 8.

Recuerde también asignar las direcciones IP adecuadas a las variables as.host.mdb (build.properties) y as.host.server (jms.properties). ¿Por qué ha añadido esas IPs?

Hemos añadido en ambos lados la IP 10.8.6.2 porque es donde está el host de la aplicación.

```

as.home=${env.J2EE_HOME}
as.lib=${as.home}/lib
as.user=admin
as.host.client=10.8.6.2
as.host.server=10.8.6.2
as.port=4848
as.passwordfile=${basedir}/passwordfile
as.target=server
jms.factoryname=jms/VisaConnectionFactory
jms.name=jms/VisaPagosQueue
jms.physname=VisaPagosQueue
|

```

```

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=${as.home}/lib
as.user=admin
as.host.mdb=10.8.6.2
as.port=4848
as.passwordfile=${basedir}/passwordfile
as.target=server

```

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.

```

<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}" />

```

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).

Hemos seleccionado la aplicación P1-jms-mdb y hemos pulsado el botón Disable. Como observamos en la imagen anterior, ahora la columna Enables tiene una cruz.

Select	Name	Deployment Order	Enabled	Engines	Action
<input type="checkbox"/>	P1-jms-mdb	100	✗	ejb	Redeploy Reload

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.

```

} else {
    // TODO: Enviar argv[0] como mensaje de texto
    messageProducer = session.createProducer(queue);
    textMessage = session.createTextMessage();
    textMessage.setText(args[0]);
    System.out.println("Enviando el siguiente mensaje: " + textMessage.getText());
    messageProducer.send(textMessage);
    messageProducer.close();
    session.close();
    connection.close();
}

```

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

Al principio podemos observar que no funciona.

```

[nicolas@masternico97:~/Escritorio/SI2/practica1b/P1-jms$ /opt/glassfish4/glassfish/bin/appclient -targetserver 10.8.6.2 -client dist/clientjms
/P1-jms-clientjms.jar idAutorizacion
mar 19, 2020 11:46:54 PM org.hibernate.validator.internal.util.Version <clinit>
INFO: HV0000001: Hibernate Validator 5.1.2.Final
mar 19, 2020 11:46:55 PM com.sun.messaging.jms.ra.ResourceAdapter start
INFORMACIÓN: MQJMSRA_RA1101: GlassFish MQ JMS Resource Adapter: Version: 5.1.1 (Build 2-c) Compile: March 17 2015 1045
mar 19, 2020 11:46:55 PM com.sun.messaging.jms.ra.ResourceAdapter start
INFORMACIÓN: MQJMSRA_RA1101: GlassFish MQ JMS Resource Adapter starting: broker is REMOTE, connection mode is TCP
mar 19, 2020 11:46:55 PM com.sun.messaging.jms.ra.ResourceAdapter start
INFORMACIÓN: MQJMSRA_RA1101: GlassFish MQ JMS Resource Adapter Started:REMOTE
mar 19, 2020 11:46:56 PM com.sun.messaging.jmq.jmsclient.ExceptionHandler throwConnectionException
ADVERTENCIA: [C4003]: Error occurred on connection creation [localhost:7676]. - cause: java.net.ConnectException: Conexión rehusada (Connection refused)
mar 19, 2020 11:47:01 PM com.sun.messaging.jmq.jmsclient.ExceptionHandler throwConnectionException
ADVERTENCIA: [C4003]: Error occurred on connection creation [localhost:7676]. - cause: java.net.ConnectException: Conexión rehusada (Connection refused)
mar 19, 2020 11:47:06 PM com.sun.messaging.jmq.jmsclient.ExceptionHandler throwConnectionException
ADVERTENCIA: [C4003]: Error occurred on connection creation [localhost:7676]. - cause: java.net.ConnectException: Conexión rehusada (Connection refused)
mar 19, 2020 11:47:11 PM com.sun.messaging.jmq.jmsclient.ExceptionHandler throwConnectionException
ADVERTENCIA: [C4003]: Error occurred on connection creation [localhost:7676]. - cause: java.net.ConnectException: Conexión rehusada (Connection refused)
mar 19, 2020 11:47:16 PM com.sun.messaging.jmq.jmsclient.ExceptionHandler throwConnectionException
ADVERTENCIA: [C4003]: Error occurred on connection creation [localhost:7676]. - cause: java.net.ConnectException: Conexión rehusada (Connection refused)
mar 19, 2020 11:47:21 PM com.sun.messaging.jmq.jmsclient.ExceptionHandler throwConnectionException
ADVERTENCIA: [C4003]: Error occurred on connection creation [localhost:7676]. - cause: java.net.ConnectException: Conexión rehusada (Connection refused)
mar 19, 2020 11:47:26 PM com.sun.messaging.jmq.jmsclient.ExceptionHandler throwConnectionException
ADVERTENCIA: [C4003]: Error occurred on connection creation [localhost:7676]. - cause: java.net.ConnectException: Conexión rehusada (Connection refused)
mar 19, 2020 11:47:31 PM com.sun.messaging.jmq.jmsclient.ExceptionHandler throwConnectionException
ADVERTENCIA: [C4003]: Error occurred on connection creation [localhost:7676]. - cause: java.net.ConnectException: Conexión rehusada (Connection refused)
mar 19, 2020 11:47:36 PM com.sun.messaging.jmq.jmsclient.ExceptionHandler throwConnectionException
ADVERTENCIA: [C4003]: Error occurred on connection creation [localhost:7676]. - cause: java.net.ConnectException: Conexión rehusada (Connection refused)

mar 19, 2020 11:47:36 PM com.sun.messaging.jms.ra.ManagedConnection <init>
GRAVE: MQJMSRA_MC4001: constructor:Aborting:JMSException on createConnection=[C4003]: Error occurred on connection creation [localhost:7676]. - cause: java.net.ConnectException: Conexión rehusada (Connection refused), error code: C4003
com.sun.messaging.jns.JMSException: [C4003]: Error occurred on connection creation [localhost:7676]. - cause: java.net.ConnectException: Conexión rehusada (Connection refused)
    at com.sun.messaging.jmq.jmsclient.ExceptionHandler.throwConnectionException(ExceptionHandler.java:280)
    at com.sun.messaging.jmq.jmsclient.ExceptionHandler.handleConnectException(ExceptionHandler.java:226)
    at com.sun.messaging.jmq.jmsclient.PortMapperClient.readBrokerPorts(PortMapperClient.java:278)
    at com.sun.messaging.jmq.jmsclient.PortMapperClient.init(PortMapperClient.java:156)
    at com.sun.messaging.jmq.jmsclient.PortMapperClient.<init>(PortMapperClient.java:98)
    at com.sun.messaging.jmq.jmsclient.protocol.tcp.TCPConnectionHandler.<init>(TCPConnectionHandler.java:171)
    at com.sun.messaging.jmq.jmsclient.protocol.tcp.TCPStreamHandler.openConnection(TCPStreamHandler.java:141)
    at com.sun.messaging.jmq.jmsclient.ConnectionInitiator.createConnection(ConnectionInitiator.java:785)
    at com.sun.messaging.jmq.jmsclient.ConnectionInitiator.createConnectionNew(ConnectionInitiator.java:260)
    at com.sun.messaging.jmq.jmsclient.ConnectionInitiator.createConnection(ConnectionInitiator.java:214)
    at com.sun.messaging.jmq.jmsclient.ConnectionInitiator.createConnection(ConnectionInitiator.java:164)
    at com.sun.messaging.jmq.jmsclient.ProtocolHandler.init(ProtocolHandler.java:871)
    at com.sun.messaging.jmq.jmsclient.ProtocolHandler.<init>(ProtocolHandler.java:1594)
    at com.sun.messaging.jmq.jmsclient.ConnectionImpl.openConnection(ConnectionImpl.java:2470)
    at com.sun.messaging.jmq.jmsclient.ConnectionImpl.init(ConnectionImpl.java:1156)
    at com.sun.messaging.jmq.jmsclient.ConnectionImpl.<init>(ConnectionImpl.java:468)
    at com.sun.messaging.jmq.jmsclient.UnifiedConnectionImpl.<init>(UnifiedConnectionImpl.java:66)
    at com.sun.messaging.jmq.jmsclient.XAConnectionImpl.<init>(XAConnectionImpl.java:64)
    at com.sun.messaging.jmq.jmsclient.XAConnectionFactory.createXAConnection(XAConnectionFactory.java:110)
    at com.sun.messaging.jms.ra.ManagedConnection.<init>(ManagedConnection.java:204)
    at com.sun.messaging.jms.ra.ManagedConnectionFactory.createManagedConnection(ManagedConnectionFactory.java:223)
    at com.sun.enterprise.resource.allocator.ConnectorAllocator.createResource(ConnectorAllocator.java:160)
    at com.sun.enterprise.resource.pool.ConnectionPool.createSingleResource(ConnectionPool.java:967)
    at com.sun.enterprise.resource.pool.ConnectionPool.createResource(ConnectionPool.java:1189)
    at com.sun.enterprise.resource.pool.datastructure.RWLockDataStructure.addResource(RWLockDataStructure.java:98)
    at com.sun.enterprise.resource.pool.ConnectionPool.addResource(ConnectionPool.java:282)
    at com.sun.enterprise.resource.pool.ConnectionPool.createResourceAndAddToPool(ConnectionPool.java:1512)
    at com.sun.enterprise.resource.pool.ConnectionPool.createResources(ConnectionPool.java:944)
    at com.sun.enterprise.resource.pool.ConnectionPool.initPool(ConnectionPool.java:230)
    at com.sun.enterprise.resource.pool.ConnectionPool.internalGetResource(ConnectionPool.java:511)
    at com.sun.enterprise.resource.pool.ConnectionPool.getResource(ConnectionPool.java:381)
    at com.sun.enterprise.resource.pool.PoolManagerImpl.getResourceFromPool(PoolManagerImpl.java:245)
    at com.sun.enterprise.resource.pool.PoolManagerImpl.getResource(PoolManagerImpl.java:170)

    at com.sun.enterprise.resource.pool.PoolManagerImpl.getResource(PoolManagerImpl.java:170)
    at com.sun.enterprise.connectors.ConnectionManagerImpl.getResource(ConnectionManagerImpl.java:354)
    at com.sun.enterprise.connectors.ConnectionManagerImpl.internalGetConnection(ConnectionManagerImpl.java:387)
    at com.sun.enterprise.connectors.ConnectionManagerImpl.allocateConnection(ConnectionManagerImpl.java:242)
    at com.sun.enterprise.connectors.ConnectionManagerImpl.allocateConnection(ConnectionManagerImpl.java:171)
    at com.sun.enterprise.connectors.ConnectionManagerImpl.allocateConnection(ConnectionManagerImpl.java:166)
    at com.sun.messaging.jms.ra.ConnectionFactoryAdapter._allocateConnection(ConnectionFactoryAdapter.java:204)
    at com.sun.messaging.jms.ra.ConnectionFactoryAdapter.createConnection(ConnectionFactoryAdapter.java:162)
    at com.sun.messaging.jms.ra.ConnectionFactoryAdapter.createConnection(ConnectionFactoryAdapter.java:144)
    at ssitz.VisaQueueMessageProducer.main(visaQueueMessageProducer.java:78)
Caused by: java.net.ConnectException: Conexión rehusada (Connection refused)
    at java.net.PlainSocketImpl.socketConnect(Native Method)
    at java.net.AbstractPlainSocketImpl.doConnect(AbstractPlainSocketImpl.java:350)
    at java.net.AbstractPlainSocketImpl.connectToAddress(AbstractPlainSocketImpl.java:206)
    at java.net.AbstractPlainSocketImpl.connect(AbstractPlainSocketImpl.java:188)
    at java.net.SocksSocketImpl.connect(SocksSocketImpl.java:392)
    at java.net.Socket.connect(Socket.java:607)
    at java.net.Socket.connect(Socket.java:556)
    at java.net.Socket.<init>(Socket.java:452)
    at java.net.Socket.<init>(Socket.java:229)
    at com.sun.messaging.jmq.jmsclient.PortMapperClient.makePortMapperClientSocket(PortMapperClient.java:316)
    at com.sun.messaging.jmq.jmsclient.PortMapperClient.readBrokerPorts(PortMapperClient.java:243)
    ... 39 more
mar 19, 2020 11:47:36 PM com.sun.enterprise.resource.allocator.ConnectorAllocator createResource
ADVERTENCIA: RAR5038:Unexpected exception while creating resource for pool jms/vtsaConnectionFactory-Connection-Pool. Exception : javax.resource.ResourceException: MQJMSRA_MC4001: constructor:Aborting:JMSException on createConnection=[C4003]: Error occurred on connection creation [localhost:7676]. - cause: java.net.ConnectException: Conexión rehusada (Connection refused), error code: C4003
mar 19, 2020 11:47:36 PM com.sun.enterprise.connectors.ConnectionManagerImpl internalGetConnection
ADVERTENCIA: RAR5117 : Failed to obtain/create connection from connection pool [ jms/vtsaConnectionFactory-Connection-Pool ]. Reason : com.sun.appserv.connectors.internal.api.PoolingException: MQJMSRA_MC4001: constructor:Aborting:JMSException on createConnection=[C4003]: Error occurred on connection creation [localhost:7676]. - cause: java.net.ConnectException: Conexión rehusada (Connection refused), error code: C4003
Exception : com.sun.messaging.jns.JMSException: MQRA:CFAI:allocation failure:createConnection:Error in allocating a connection. Cause: MQJMSRA_MC4001: constructor:Aborting:JMSException on createConnection=[C4003]: Error occurred on connection creation [localhost:7676]. - cause: java.net.ConnectException: Conexión rehusada (Connection refused), error code: C4003

```

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.

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: 10.8.6.2
Name or IP address; if name, must contain only alphanumeric, underscore, dash, or dot characters

Port: \$JMS_PROVIDER_PORT
Listener port for servicing JMS requests

Admin Username: * admin
User name for maintaining the JMS service; can be up to 255 characters, must contain only alphanumeric, underscore, dash, or dot characters

Admin Password: *
Password for JMS administrator

Confirm New Password: *

Save Cancel

Tras reiniciar, haciendo `asadmin stop-domain domain1` y volviendo a hacer `asadmin start-domain domain1` volvemos a ejecutar el comando anterior y vemos que ya funciona.

```
nicolas@masternico97:~/Escritorio/SI2/practicaIB/P1-jms$ /opt/glassfish4/glassfish/bin/appclient -targetserver 10.8.6.2 -client dist/clientjms/P1-jms-clientjms.jar idAutorizacion
mar 20, 2020 12:39:57 AM org.hibernate.validator.internal.util.Version <clinit>
INFO: HV000001: Hibernate Validator 5.1.2.Final
mar 20, 2020 12:39:58 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 20, 2020 12:39:58 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 20, 2020 12:39:58 AM com.sun.messaging.jms.ra.ResourceAdapter start
INFORMACIÓN: MQJMSRA_RA1101: GlassFish MQ JMS Resource Adapter Started:REMOTE
Enviando el siguiente mensaje: idAutorizacion
```

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.

```
nicolas@masternico97:~/Escritorio/SI2/practicaIB/P1-jms$ /opt/glassfish4/glassfish/bin/appclient -targetserver 10.8.6.2 -client dist/clientjms/P1-jms-clientjms.jar -browse
mar 20, 2020 12:45:25 AM org.hibernate.validator.internal.util.Version <clinit>
INFO: HV000001: Hibernate Validator 5.1.2.Final
mar 20, 2020 12:45:26 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 20, 2020 12:45:26 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 20, 2020 12:45:27 AM com.sun.messaging.jms.ra.ResourceAdapter start
INFORMACIÓN: MQJMSRA_RA1101: GlassFish MQ JMS Resource Adapter Started:REMOTE
Mensajes en cola:
idAutorizacion
```

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
- Cáncélelo con el cliente
- Obtenga evidencias de que se ha cancelado y de que el saldo se ha rectificado

Primero realizamos el pago como observamos en las siguientes imágenes:

Pago con tarjeta

Proceso de un pago

Id Transacción:	99
Id Comercio:	1
Importe:	25
Número de visa:	1111 2222 3333 4444
Titular:	Jose Garcia
Fecha Emisión:	11/09
Fecha Caducidad:	11/20
CVV2:	123
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 type="radio"/> True <input checked="" type="radio"/> False
Pagar	

Pago con tarjeta

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

```
idTransaccion: 99
idComercio: 1
importe: 25.0
codRespuesta: 000
idAutorizacion: 1
```

[Volver al comercio](#)

Prácticas de Sistemas Informáticos II

idautorizacion [PK] serial	idtransaccion character(16)	codrespuesta character(3)	importe double precision	idcomercio character(16)	numerotarjeta character(19)	fecha timestamp without time zone
1	99	000	25	1	1111 2222 3333 4444	2020-03-20 04:53:22.334782

numerotarjeta character(19)	titular character varying(128)	validadesde character(5)	validahasta character(5)	codigoverificacion character(3)	saldo double precision
1111 2222 3333 4444	Jose Garcia	11/09	11/20	123	975

A continuación lo cancelamos con el cliente:

```
nicolas@masternico97:~/Escritorio/SI2/practica1b/P1-jms$ /opt/glassfish4/glassfish/bin/appclient -targetserver 10.8.6.2 -client dist/clientjms/P1-jms-clientjms.jar 1
mar 20, 2020 12:54:55 PM org.hibernate.validator.internal.util.Version <clinit>
INFO: HV000001: Hibernate Validator 5.1.2.Final
mar 20, 2020 12:54:56 PM com.sun.messaging.jms.ra.ResourceAdapter start
INFORMACIÓN: MQJMSRA_RA1101: GlassFish MQ JMS Resource Adapter: Version: 5.1.1 (Build 2-c) Compile: March 17 2015 1045
mar 20, 2020 12:54:56 PM com.sun.messaging.jms.ra.ResourceAdapter start
INFORMACIÓN: MQJMSRA_RA1101: GlassFish MQ JMS Resource Adapter starting: broker is REMOTE, connection mode is TCP
mar 20, 2020 12:54:56 PM com.sun.messaging.jms.ra.ResourceAdapter start
INFORMACIÓN: MQJMSRA_RA1101: GlassFish MQ JMS Resource Adapter Started:REMOTE
Enviando el siguiente mensaje: 1
```

Como observamos en el log, no ha habido ningún error.

Record Number	Log Level	Message	Logger	Timestamp	Name-Value Pairs
256	INFO	update tarjeta set saldo=saldo+(select importe from pago where idautorizacion=?where numerotarjeta=... (details))	VisaCancelacionJMSBean	20-mar-2020 05:34:02.083	{levelValue=800, timeMillis=1584707642083}
255	INFO	update pago set codRespuesta=999 where idautorizacion=? (details)	VisaCancelacionJMSBean	20-mar-2020 05:34:02.079	{levelValue=800, timeMillis=1584707642079}
254	INFO	MESSAGE BEAN: Message received: 1(details)	VisaCancelacionJMSBean	20-mar-2020 05:34:02.078	{levelValue=800, timeMillis=1584707642078}
253	INFO	onMessage(details)	VisaCancelacionJMSBean	20-mar-2020 05:34:02.077	{levelValue=800, timeMillis=1584707642077}

Y como observamos en la base de datos tanto el codRespuesta como el saldo de la tarjeta se han modificado satisfactoriamente.

	numerotarjeta character(19)	titular character varying(128)	validadesde character(5)	validahasta character(5)	codigoverificacion character(3)	saldo double precision
1	1111 2222 3333 4444	Jose Garcia	11/09	11/20	123	1000

	idautorizacion [PK] serial	idtransaccion character(16)	codrespuesta character(3)	importe double precision	idcomercio character(16)	numerotarjeta character(19)	fecha timestamp without time zone
1	1	99	999	25	1	1111 2222 3333 4444	2020-03-20 04:53:22.334782