


| | | | | | |
|---|-------------|--|-----------|--------------|------------------|
|  | | Escuela Politécnica Superior Ingeniería Informática Práctica de Sistemas Informáticos 1 | | | |
| Grupo | 2323 | Práctica | 1B | Fecha | 6/03/2023 |
| Alumno/a | | Hidalgo, Gamborino, Sergio | | | |
| Alumno/a | | Ibáñez, González, Miguel | | | |

Práctica 1: Arquitectura de JAVA EE

Cuestión 1:

Abrir el archivo VisaDAOLocal.java y comprobar la definición de dicha interfaz. Anote en la memoria comentarios sobre las librerías Java EE importadas y las anotaciones utilizadas. ¿Para qué se utilizan? Comparar esta interfaz con el fichero de configuración del web service implementado en la práctica P1A

- java.sql.Connection: Para conectarse a la base de datos a través del JDBC.
- java.sql.PreparedStatement: Para ejecutar una query preparada.
- java.sql.ResultSet: Para tratar como objeto el resultado de la query.
- java.sql.SQLException: Provee la información de las excepciones tipo SQL.
- java.sql.Statement: Para rear queries no preparadas.
- java.util.ArrayList: Para utilizar una array como una lista.
- javax.ejb.Local: Para declarar que es local.

La primera diferencia es que al ser una interfaz, sus métodos no están definidos, además, los métodos que realizan las queries no están definidos pues se hará de forma privada. También se ha tenido que volver a colocar el método getPagos tal y como estaba en el fichero inicialmente proporcionado, porque PagoBean[] es el tipo que devuelve en la función de la interfaz. Todos los imports a excepción de javax.ejb.Local se utilizan en el fichero de la P1A, y por ello están en el fichero "VisaDAOBean.java".

Ejercicio 1

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

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

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

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

```
import javax.ejb.Stateless;

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

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

Se cambia el return de get pagos a como estaba anteriormente.

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 ssii2.visa.VisaDAOLocal;que
import javax.ejb.EJB;
```

Se han realizado estos imports en los ficheros del paquete ssii2.controlador, concretamente a "DelPagos.java", "GetPagos.java" y "ProcesaPago.java"

Se deberán eliminar estas otras importaciones que dejan de existir en el proyecto.

En esos ficheros se han eliminado los siguientes imports:

```
import javax.xml.ws.WebServiceRef;
import javax.xml.ws.BindingProvider;
import ssii2.visa.VisaDAOSService;
import ssii2.visa.VisaDAOSService;
```

Siendo los 2 últimos stubs generados 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;
```

Esto habrá que añadirlo en los archivos previamente mencionados, dentro de la clase (convenientemente donde estén declaradas las variables privadas, para tenerlo ordenado).

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;

try {
    VisaDAOWSService service = new VisaDAOWSService();
    dao = service.getVisaDAOWSPort();
    BindingProvider bp = (BindingProvider) dao;
    bp.getRequestContext().put(BindingProvider.ENDPOINT_ADDRESS_PROPERTY,
        getServletContext().getInitParameter("webmaster"));
} catch (Exception e) {
    System.err.println("[delPago] " + e);
    return;
}
```

Habrà que eliminar estas líneas de código de los anteriores ficheros.

Cuestión 2:

Abrir el archivo application.xml y explicar su contenido. Verifique el contenido de todos los archivos .jar / .war / .ear que se han construido hasta el momento (empleando el comando jar -tvf). Explique brevemente el contenido y ponga evidencias en la memoria.

Es un fichero que define la aplicación, dividida en el módulo del server "P1-ejb.jar" y el módulo del cliente "P1-ejb-cliente.war" con su ruta.

En el cliente, al ejecutar el comando sobre "P1-ejb-cliente.war", se muestran los archivos y ficheros dentro del directorio build/client que resultan de la compilación.

```

sergio@seregio-pc: /opt/glassfish4/glassfish/domains/domain1/applications/P1-ejb-base
0 Mon Mar 06 23:30:08 CET 2023 META-INF/
106 Mon Mar 06 23:30:06 CET 2023 META-INF/MANIFEST.MF
0 Mon Feb 27 12:39:24 CET 2023 WEB-INF/
0 Mon Feb 27 12:39:22 CET 2023 WEB-INF/classes/
0 Mon Feb 27 12:39:22 CET 2023 WEB-INF/classes/ssii2/
0 Mon Feb 27 12:39:22 CET 2023 WEB-INF/classes/ssii2/controlador/
0 Mon Feb 27 12:39:22 CET 2023 WEB-INF/classes/ssii2/error/
0 Mon Feb 27 12:39:22 CET 2023 WEB-INF/classes/ssii2/filtros/
0 Mon Feb 27 12:39:22 CET 2023 WEB-INF/classes/ssii2/visa/
0 Mon Feb 27 12:39:22 CET 2023 WEB-INF/lib/
0 Mon Feb 27 12:39:24 CET 2023 error/
2844 Mon Mar 06 23:25:48 CET 2023 WEB-INF/classes/ssii2/controlador/ComienzaPago.class
1513 Mon Mar 06 23:25:48 CET 2023 WEB-INF/classes/ssii2/controlador/DelPagos.class
1365 Mon Mar 06 23:25:48 CET 2023 WEB-INF/classes/ssii2/controlador/GetPagos.class
4979 Mon Mar 06 23:25:48 CET 2023 WEB-INF/classes/ssii2/controlador/ProcesaPago.class
1894 Mon Feb 27 12:39:22 CET 2023 WEB-INF/classes/ssii2/controlador/ServletRaiz.class
611 Mon Feb 27 12:39:22 CET 2023 WEB-INF/classes/ssii2/error/ErrorVisa.class
188 Mon Feb 27 12:39:22 CET 2023 WEB-INF/classes/ssii2/error/ErrorVisaCVV.class
199 Mon Feb 27 12:39:22 CET 2023 WEB-INF/classes/ssii2/error/ErrorVisaFechaCaducidad.class
197 Mon Feb 27 12:39:22 CET 2023 WEB-INF/classes/ssii2/error/ErrorVisaFechaEmision.class
191 Mon Feb 27 12:39:22 CET 2023 WEB-INF/classes/ssii2/error/ErrorVisaNumero.class
192 Mon Feb 27 12:39:22 CET 2023 WEB-INF/classes/ssii2/error/ErrorVisaTitular.class
2608 Mon Feb 27 12:39:22 CET 2023 WEB-INF/classes/ssii2/filtros/CompruebaSesion.class
3135 Mon Feb 27 12:39:22 CET 2023 WEB-INF/classes/ssii2/visa/ValidadorTarjeta.class
6023 Mon Feb 27 12:39:24 CET 2023 WEB-INF/web.xml
455 Mon Feb 27 12:39:24 CET 2023 borradoerror.jsp
501 Mon Feb 27 12:39:24 CET 2023 borradoook.jsp
509 Mon Feb 27 12:39:24 CET 2023 cabecera.jsp
283 Mon Feb 27 12:39:24 CET 2023 error/muestraerror.jsp
2729 Mon Feb 27 12:39:24 CET 2023 formdatosvisa.jsp
1257 Mon Feb 27 12:39:24 CET 2023 listapagos.jsp
1178 Mon Feb 27 12:39:24 CET 2023 pago.html
1142 Mon Feb 27 12:39:24 CET 2023 pagoexito.jsp
104 Mon Feb 27 12:39:24 CET 2023 pie.html
5011 Mon Feb 27 12:39:24 CET 2023 testbdd.jsp

```

En el server, al ejecutar el comando sobre “P1-ejb.jar”, es igual desde el directorio build/server, pero el empaquetado es distinto (en vez de .war es .jar).

```

sergio@seregio-pc: /opt/glassfish4/glassfish/domains/domain1/applicat
0 Mon Mar 06 23:29:56 CET 2023 META-INF/
106 Mon Mar 06 23:29:54 CET 2023 META-INF/MANIFEST.MF
0 Mon Feb 27 12:39:22 CET 2023 ssii2/
0 Mon Feb 27 12:39:22 CET 2023 ssii2/visa/
255 Mon Feb 27 12:39:22 CET 2023 META-INF/sun-ejb-jar.xml
1714 Mon Feb 27 12:39:22 CET 2023 ssii2/visa/DBTester.class
1464 Mon Feb 27 12:39:22 CET 2023 ssii2/visa/PagoBean.class
856 Mon Feb 27 12:39:22 CET 2023 ssii2/visa/TarjetaBean.class
7029 Mon Feb 27 12:47:02 CET 2023 ssii2/visa/VisaDAOBean.class
593 Mon Mar 06 23:25:48 CET 2023 ssii2/visa/VisaDAOLocal.class

```

Y en la aplicación el empaquetado es tipo .ear (“P1-ejb.ear”) y muestra el .xml y el .MF en conf/application las referencias a ambos paquetes en el directorio dist.

```

sergio@seregio-pc: /opt/glassfish4/glassfish/domains/domain1/applicat
0 Tue Mar 07 00:03:04 CET 2023 META-INF/
117 Tue Mar 07 00:03:02 CET 2023 META-INF/MANIFEST.MF
508 Mon Feb 27 12:24:08 CET 2023 META-INF/application.xml
21203 Tue Mar 07 00:02:42 CET 2023 P1-ejb-cliente.war
6899 Tue Mar 07 00:02:28 CET 2023 P1-ejb.jar

```

Ejercicio 3:

Editar el archivo build.properties para que la propiedad as.host contenga la dirección IP del servidor de aplicaciones. Indica el valor y por qué es ese valor.

La ip del servidor que hostea la aplicación estará en la siguiente variable de “build.properties”

```
as.host=10.6.4.1
```

Editar el archivo postgresql.properties para la propiedad db.client.host y db.host contengan las direcciones IP adecuadas para que el servidor de aplicaciones se conecte al postgresql, ambos estando en servidores diferentes. Indica qué valores y porqué son esos valores.

La ip del servidor que hostea la base de datos estará en la siguiente variable de “postgresql.properties”

```
db.host=10.6.4.2
```

La ip del servidor que hostea la aplicación y se conectará como cliente estará en la siguiente variable de “postgresql.properties”

```
db.client.host=10.6.4.1
```

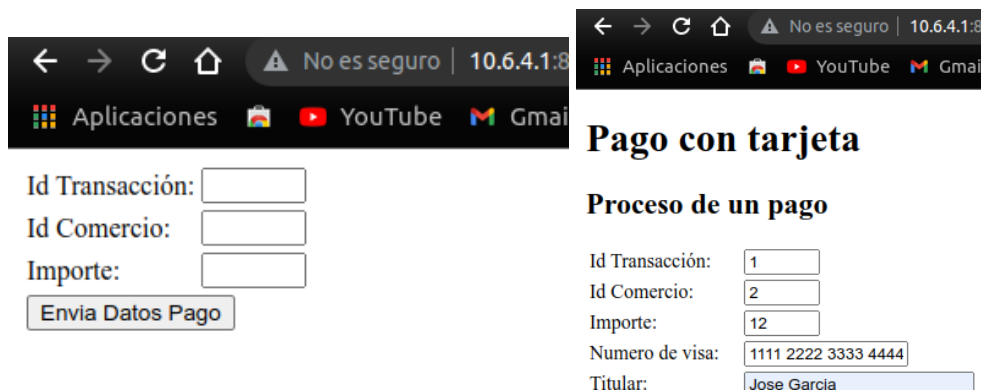
Ejercicio 4:

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.

Si la base de datos no se ha generado previamente, será necesario crearla usando ant regenerar-bd

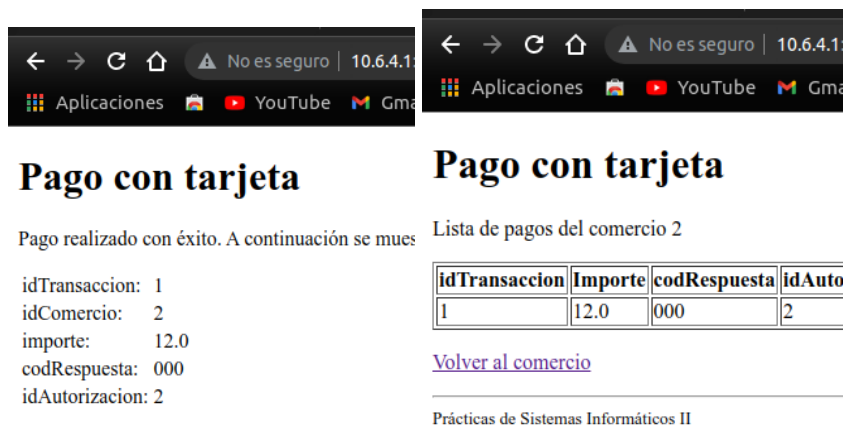
Incluya en la memoria de prácticas todos los pasos necesarios para resolver este ejercicio, así como las evidencias obtenidas. Se pueden incluir por ejemplo capturas de pantalla

Llamada a las páginas:



The screenshot shows a web browser with the address bar displaying '10.6.4.1:8080'. The page title is 'Pago con tarjeta'. Below the title, there is a section titled 'Proceso de un pago'. It contains several input fields: 'Id Transacción:' with a value of '1', 'Id Comercio:' with a value of '2', 'Importe:' with a value of '12', 'Numero de visa:' with a value of '1111 2222 3333 4444', and 'Titular:' with a value of 'Jose Garcia'. There is also a button labeled 'Envia Datos Pago'.

Pago realizado y listado:



The screenshot shows a web browser with the address bar displaying '10.6.4.1:8080'. The page title is 'Pago con tarjeta'. Below the title, there is a message: 'Pago realizado con éxito. A continuación se muestra la lista de pagos del comercio 2'. Below this message, there is a table with the following data:

| idTransaccion | Importe | codRespuesta | idAuto |
|---------------|---------|--------------|--------|
| 1 | 12.0 | 000 | 2 |

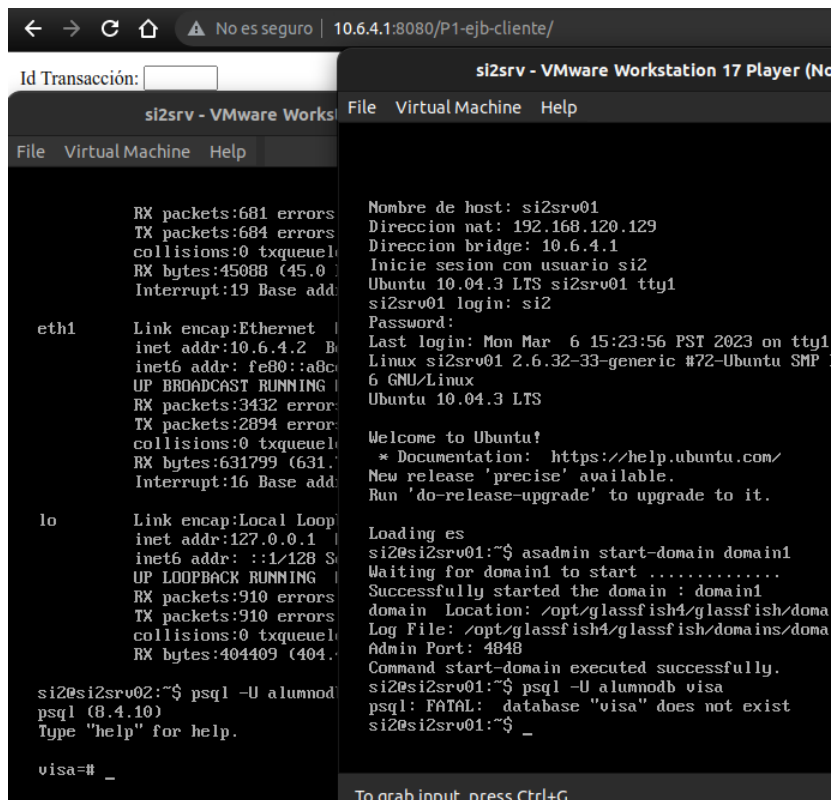
Below the table, there is a link labeled 'Volver al comercio'. At the bottom of the page, there is a footer that reads 'Prácticas de Sistemas Informáticos II'.

Borrado:



The screenshot shows a web browser with the address bar displaying '10.6.4.1:8080/P1-'. The page title is 'Pago con tarjeta'. Below the title, there is a message: 'Se han borrado 1 pagos correctamente para el comercio 2'. Below this message, there is a link labeled 'Volver al comercio'. At the bottom of the page, there is a footer that reads 'Prácticas de Sistemas Informáticos II'.

Ambas máquinas virtuales, la de la base de datos (10.6.4.2) y la que aloja la aplicación (10.6.4.1).



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

Fichero “VisaDAORemote.java”:

```
import javax.ejb.Remote;

/*
 * java.sql.Conection: Para conectarse a la base de datos a través
 * java.sql.PreparedStatement: Para ejecutar una query preparada
 * java.sql.ResultSet: Para tratar como objeto el resultado de la
 * java.sql.SQLException: Provee la información de las excepciones
 * java.sql.Statement: Para crear queries no preparadas
 * java.util.ArrayList: Para utilizar una array como una lista
 * javax.ejb.Remote: Para declarar que es remoto
 */

@Remote
public interface VisaDAORemote {
```

Fichero "VisaDAOBean.java":

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

Fichero "TarjetaBean.java":

```
import java.io.Serializable;
public class TarjetaBean implements java.io.Serializable {
```

Fichero "PagoBean.java":

```
import java.io.Serializable;

public class PagoBean implements java.io.Serializable
```

La ejecución es correcta y funcional (al igual que el anterior ejercicio).

Ejercicio 6:

Realizar los cambios comentados en la aplicación P1-base para convertirla en P1-ejb-clienteremoto. 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.

En "build.properties"

```
nombre=P1-ejb-cliente-remoto
```

Los cambios en "PagoBean.java" y "TarjetaBean.java" son los mismos que en el ejercicio anterior.

Tras copiar "VisaDAORemote.java" al directorio del paquete visa, se añaden las siguientes líneas en "ProcesaPago.java", "ProcesaPago.java",

Los imports de java:

```
import javax.ejb.EJB;
```



```
import ssii2.visa.VisaDAORemote;
```

Y como variable privada:

```
@EJB(name = "VisaDAOBean", beanInterface = VisaDAORemote.class)

private VisaDAORemote dao;
```

Por tanto se debe eliminar estas líneas.

```
import ssii2.visa.dao.VisaDAO;
```

```
VisaDAO dao = new VisaDAO();
```

Después se cambiará como en el anterior apartado los ficheros “build.properties” y “postgresql.properties” para que el servidor sea el 10.6.4.2 y el cliente 10.6.4.1.

El pago se realiza correctamente

The screenshot shows a web browser window with the URL `10.6.4.1:8080/P1-ejb-cliente-remoto/procesapago`. The page title is "Pago con tarjeta" and the content states "Pago realizado con éxito. A continuación se muestra el detalle de la transacción:". The transaction details are as follows:

| | |
|-----------------|------|
| idTransaccion: | 1 |
| idComercio: | 2 |
| importe: | 12.0 |
| codRespuesta: | |
| idAutorizacion: | |

Below the details is a link: [Volver al comercio](#).

At the bottom of the browser window, a terminal window is open, showing a SQL query result:

```
idautorizacion | idtransaccion | codrespuesta
| importe | idcomercio | numerotarjeta
| fecha
+-----+-----+-----+
+-----+-----+-----+
+-----+-----+-----+
+-----+-----+-----+
+-----+-----+-----+
| 2 | 1 | 000
| 12 | 2 | 1111 2222 3333 4444
| 2023-03-12 16:07:57.995159
(1 row)
```

The terminal window also shows a prompt `(END)` and some red error messages.

Pago con tarjeta

Lista de pagos del comercio 2

| idTransaccion | Importe | codRespuesta | idAutoriza |
|---------------|---------|--------------|------------|
| 1 | 12.0 | 000 | 2 |

Ejercicio 7:

Modificar la aplicación VISA para soportar el campo saldo: Archivo TarjetaBean.java:

- Añadir el atributo saldo y sus métodos de acceso: `private double saldo;`

Archivo VisaDAOBean.java:

- Importar la definición de la excepción `EJBException` que debe lanzar el servlet para indicar que se debe realizar un rollback: `import javax.ejb.EJBException;`
- Declarar un prepared statement para recuperar el saldo de una tarjeta de la base de datos.
- Declarar un prepared statement para actualizar el nuevo saldo calculado en la base de datos.
- Modificar el método `realizaPago` con las siguientes acciones:
 - 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.
- Modificar el servlet `ProcesaPago` para que capture la posible interrupción `EJBException` lanzada por `realizaPago`, y, en caso de que se haya lanzado, devuelva la página de error mediante el método `enviaError` (recordar antes de retornar que se debe invalidar la sesión, si es que existe).
- Incluye en la memoria cada fragmento de código donde se han ido añadiendo las modificaciones solicitadas.

Añadimos a TarjetaBean.java:

```
private double saldo;

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

/**
 * Establece el saldo
 * @param saldo el saldo
 */
public void setSaldo(String saldo) {
    this.saldo = saldo;
}
```

Añadimos a VisaDAOBean.java:

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

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

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

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

    if (rs.next()) {
        Double saldo = rs.getDouble("saldo");
        if (saldo < pago.getImporte()) {
            pago.setIdAutorizacion(null);
            return null;
        } else {
            String update = UPDATE_SALDO_QRY;
```

```

        Double saldo_decr = saldo - pago.getImporte();
        errorLog(update);
        pstmt = con.prepareStatement(update);
        pstmt.setDouble(1, saldo_decr);
        pstmt.setString(2,
pago.getTarjeta().getNumero());

        ret = null;
        if (!pstmt.execute()
            && pstmt.getUpdateCount() == 1) {
            ret = pago;
        }

    }
} else{
    throw new EJBException("No existe saldo en esa
tarjeta");

else {

```

Añadimos a RealizaPago.java:

```

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

            return;
        }

    } catch (EJBException e) {
        if(sesion != null) sesion.invalidate();
        enviaError(new Exception("No se ha completado el pago "),
request, response);
        return;
    }
}

```

Ejercicio 8

Desplegar y probar la nueva aplicación creada.

- Probar a realizar pagos correctos. Comprobar que disminuye el saldo de las tarjetas sobre las que realice operaciones. Añadir a la memoria las evidencias obtenidas.
- Realice una operación con identificador de transacción y de comercio duplicados. Compruebe que el saldo de la tarjeta especificada en el pago no se ha variado.
- Incluya en la memoria de prácticas todos los pasos necesarios para resolver este ejercicio así como las evidencias obtenidas. Se pueden incluir por ejemplo capturas de pantalla.

Pago correcto (se mira que tras realizar el pago se pueda visualizar en los pagos y que el saldo haya disminuido):

The screenshot shows a web browser with the address bar displaying "10.6.4.1:8080/P1-ejb-cliente/procesapago". The page title is "Pago con tarjeta". Below the title, it says "Pago realizado con éxito. A continuación se muestra la lista de pagos del comercio 12".

Payment details:

- idTransaccion: 1
- idComercio: 12
- importe: 12.0
- codRespuesta: 000
- idAutorizacion: 1

There is a link "Volver al comercio".

Below this, there is a table titled "Lista de pagos del comercio 12":

| idTransaccion | Importe | codRespuesta | idAutorizacion |
|---------------|---------|--------------|----------------|
| 1 | 12.0 | 000 | 1 |

There is another link "Volver al comercio" below the table.

Si se intenta hacer un pago incorrecto (duplicado), no se realizará el pago.

The screenshot shows a terminal window with the following content:

```

sergio@seregio-pc: /opt/glassfish4/glassfish/domains/domain1/applications/j2e...
sergio@seregio-pc: /opt/glassfish4/glassfish/dom... x sergio@seregio-pc: /opt/glassfish4/glassfish/dom... x

Schema | Name | Type | Owner
-----+-----+-----+-----
public | pago | table | alumnodb
public | pago_idautorizacion_seq | sequence | alumnodb
public | tarjeta | table | alumnodb
(3 rows)

visa=# select * from tarjeta ;
visa=# select * from tarjeta ;
visa=# select * from tarjeta where numerotarjeta='1111 2222 3333 4444';
      numerotarjeta | titular | validadesde | validahasta | codigoverificacion | saldo
-----+-----+-----+-----+-----+-----
1111 2222 3333 4444 | Jose Garcia | 11/09 | 11/24 | 123 | 988
(1 row)

```

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

The screenshot shows the GlassFish Server Open Source Edition administration console. The top bar indicates the user is 'admin' on domain 'domain1' at server '10.6.4.1'. The main content area is titled "Edit JMS Connection Factory".

General Settings:

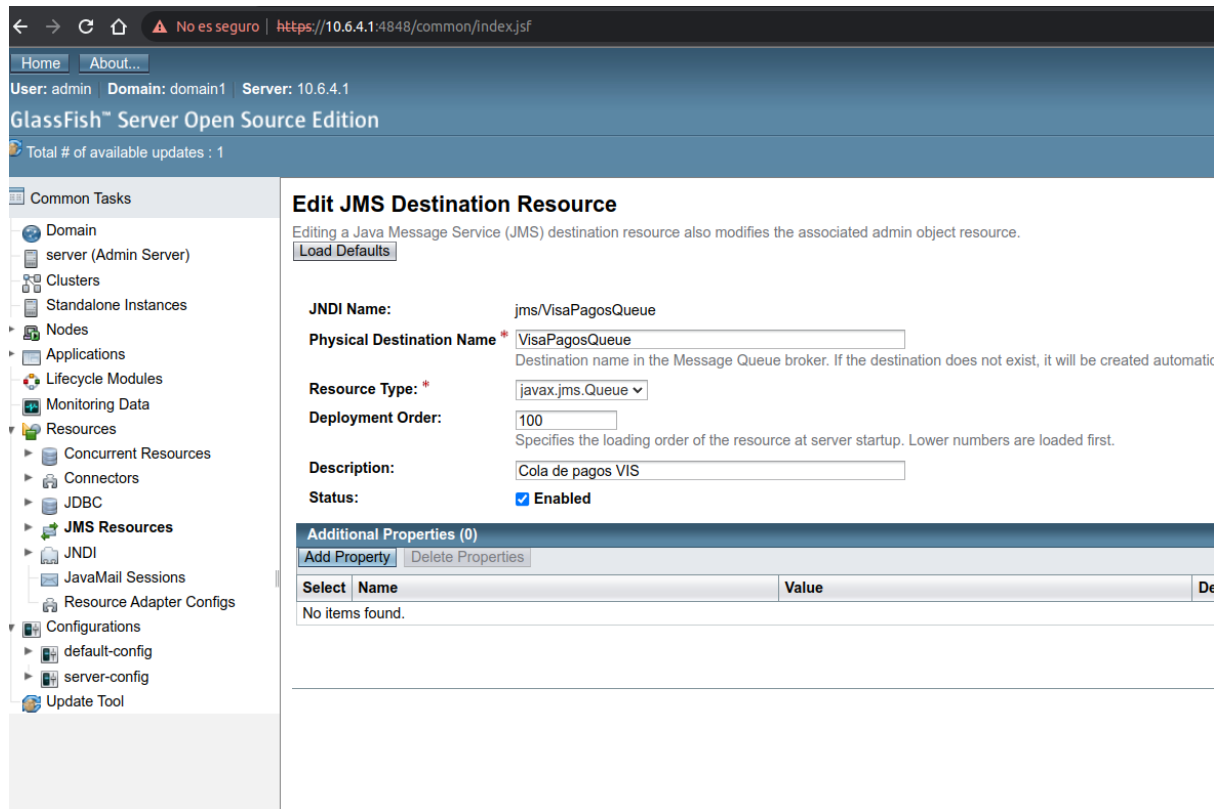
- JNDI Name: jms/VisaConnectionFactory
- Logical JNDI Name:
- Resource Type: javax.jms.TopicConnectionFactory
- Description: Factoría de conexiones a la cola de pagos
- Status: ☒ Enabled

Pool Settings:

- Initial and Minimum Pool Size: 8 Connections (Minimum and initial number of connections maintained in the pool)
- Maximum Pool Size: 32 Connections (Maximum number of connections that can be created to satisfy client requests)
- Pool Resize Quantity: 2 Connections (Number of connections to be removed when pool idle timeout expires)
- Idle Timeout: 300 Seconds (Maximum time that connection can remain idle in the pool)
- Max Wait Time: 60000 Milliseconds (Amount of time caller waits before connection timeout is sent)
- On Any Failure: ☐ Close All Connections (Close all connections and reconnect on failure, otherwise reconnect only when used)

Ejercicio 10

En la máquina virtual donde se encuentra el servidor de aplicaciones (10.X.Y.Z2), declare manualmente la conexión empleando la consola de administración, tal y como se adjunta en la Figura 5 Incluye una captura de pantalla donde se muestre dicha consola de administración con los cambios solicitados.



Ejercicio 11:

- Modifique el fichero sun-ejb-jar.xml para que el MDB conecte adecuadamente a su connection factory
- Incluya en la clase VisaCancelacionJMSBean:
 - Consulta SQL necesaria para obtener el código de respuesta del pago cuyo idAutorizacion coincida con lo recibido por el mensaje
 - 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 obtenga el idAutorización de la cola de mensajes, compruebe si el pago con dicho idAutorizacon tiene código de respuesta 000 y en ese caso actualice el código de respuesta y rectifique el saldo de la tarjeta. Para ello tome de ejemplo el código SQL de ejercicios anteriores, de modo que se use un prepared statement que haga bind del idAutorizacion para cada mensaje recibido.
 - Control de errores en el método onMessage y cierre de conexiones.

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

Nombre de la connection factory

```
<ejb>
  <ejb-name>VisaCancelacionJMSBean</ejb-name>
  <!-- DONE - definir el nombre de la connection factory -->
  <mdb-connection-factory>
    <jndi-name>VisaConnectionFactory</jndi-name>
  </mdb-connection-factory>
</ejb>
```

En el fichero "VisaCancelacionJMSBean.java":

```
private static final String GET_RESP_CODE_QRY =
    "Select codrespuesta from pago" +
    " where idautorización=?";
private static final String UPDATE_RESP_CODE_QRY =
    "update pago set codrespuesta=999" +
    "where idautorización=?";
private static final String UPDATE_SALDO_QRY =
    "update tarjeta " +
    "set saldo=saldo + importe from pago" +
    "where pago.idautorizacion=? and" +
    "pago.numerotarjeta=tarjeta.numerotarjeta";

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

            String select = GET_RESP_CODE_QRY;
            pstmt = con.prepareStatement(select);
            pstmt.setString(1, Integer.toString(idautorizacion));
            rs = pstmt.executeQuery();

            String codresp = rs.getString("codrespuesta");
```

```

        if (codresp.equals("000")) {

            select = UPDATE_CANCELA_QRY;
            pstmt = con.prepareStatement(select);
            pstmt.setString(1,
Integer.toString(idautorizacion));
            rs = pstmt.executeQuery();

            select = UPDATE_SALDO_QRY;
            pstmt = con.prepareStatement(select);
            pstmt.setString(1,
Integer.toString(idautorizacion));
            rs = pstmt.executeQuery();

        }

    } 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 (rs != null) {
            rs.close();
            rs = null;
        }

        if (pstmt != null) {
            pstmt.close();
            pstmt = null;
        }

        if (con != null) {
            closeConnection(con);
            con = null;
        }

    } catch (SQLException e) {
    }

}

```



```
}
```

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

En el fichero "VisaQueueMessageProducer.java":

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

@Resource(mappedName = "jms/VisaPagosQueue")
private static Queue queue;

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

Ejercicio 13

Automatice la creación de los recursos JMS (cola y factoría de conexiones) en el build.xml y jms.xml. Para ello, indique en jms.properties los nombres de ambos y el Physical Destination Name de la cola de acuerdo a los valores asignados en los ejercicios 9 y 10. Recuerde también asignar las direcciones IP adecuadas a las variables as.host.mdb (build.properties) y as.host.server (jms.properties). ¿Por qué ha añadido esas IPs?

Borre desde la consola de administración de Glassfish la connectionFactory y la cola creadas manualmente y ejecute:

```
cd P1-jms
ant todo
```

Compruebe en la consola de administración del Glassfish que, efectivamente, los recursos se han creado automáticamente. Incluye una captura de pantalla, donde se muestre la consola de administración con los recursos creados. Revise el fichero jms.xml y anote en la memoria de prácticas cuál es el comando equivalente para crear una cola JMS usando la herramienta asadmido de la cola se debe ejecutar:n.

Se utiliza la misma IP que en la que se desplegaba el servicio del P1-ejb-transaccional en los .properties:

```
as.host.mdb=10.6.4.1
as.host.server=10.6.4.1
```

Se Asignan los estos nombres a las variables de jms.properties.

```
jms.factoryname=jms/VisaConnectionFactory
jms.name=jms/VisaPagosQueue
jms.physname=Visa
```

Se crean ambos recursos:

The image displays two side-by-side screenshots of the JBoss web console interface, specifically the JMS configuration section. Both screenshots show the URL <https://10.6.4.1:4848/common/index.jsf> and the server version 'er: 10.6.4.1'.

The left screenshot is titled 'JMS Connection Factories' and shows a table with two entries:

| Select | JNDI Name |
|--------------------------|--|
| <input type="checkbox"/> | jms/__defaultConnectionFactory |
| <input type="checkbox"/> | jms/VisaConnectionFactory |

The right screenshot is titled 'JMS Destination Resource' and shows a table with one entry:

| Select | JNDI Name |
|--------------------------|------------------------------------|
| <input type="checkbox"/> | jms/VisaPagosQueue |

Ejercicio 14

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. Para enviar mensajes a la cola de mensajes usando el cliente se debe usar el comando:

```
/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.

Para verificar el contenido de la cola se debe ejecutar:

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

Detenga la ejecución del MDB con la consola de administración para que no consuma los mensajes de la cola (check de ‘Enabled’ en Applications/P1-jms-mdb y guardar los cambios)

Envíe un mensaje a la cola, por ejemplo, el texto “idAutorizacion” y compruebe el contenido de la cola. Incluya en la memoria de prácticas evidencias de todos los pasos, por ejemplo, capturas de pantalla.

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

- Realice un pago con la aplicación web (P1-ejb-transaccional)
- 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

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