		Escuela Politécnica Superior Ingeniería Informática Prácticas de Sistemas Informáticos 2			
Grupo	2402	Práctica	1B	Fecha	08/03/2021
Alumno/a	Rivas Molina, Lucía				
Alumno/a	Santo-Tomás López, Daniel				

Práctica 1B: 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? Comparar esta interfaz con el fichero de configuración del web service implementado en la práctica P1A.

Están importadas las librerías java.sql para el acceso a la base de datos:

- java.sql.connection para establecer la conexión con la base de datos.
- Java.sql.resultset para almacenar los resultados de las consultas.
- Java.sql.sqlexception con las excepciones.
- Java.sql.statement para ejecutar sentencias.
- Javax.ejb.local para indicar que es local.

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
- ☐ 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.

Hemos añadido @stateless en vez del webservice en la clase VisaDaoBean.java y en el método getPagos() hemos vuelto a devolver arrays en vez de arraylists.

```

29  @Stateless(mappedName="VisaDAOBean")
30  public class VisaDAOBean extends DBTester implements VisaDAOLocal {
31
32      private boolean debug = false;
33
34
35
36      ret = new PagoBean[pagos.size()];
37      ret = pagos.toArray(ret);
38
39      // Cerramos / devolvemos la conexion al pool
40      pcon.close();

```

Ejercicio 2:

Modificar el servlet `ProcesaPago` para que acceda al EJB local. Para ello, modificar el archivo `ProcesaPago.java` de la siguiente manera: (...).

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()`).

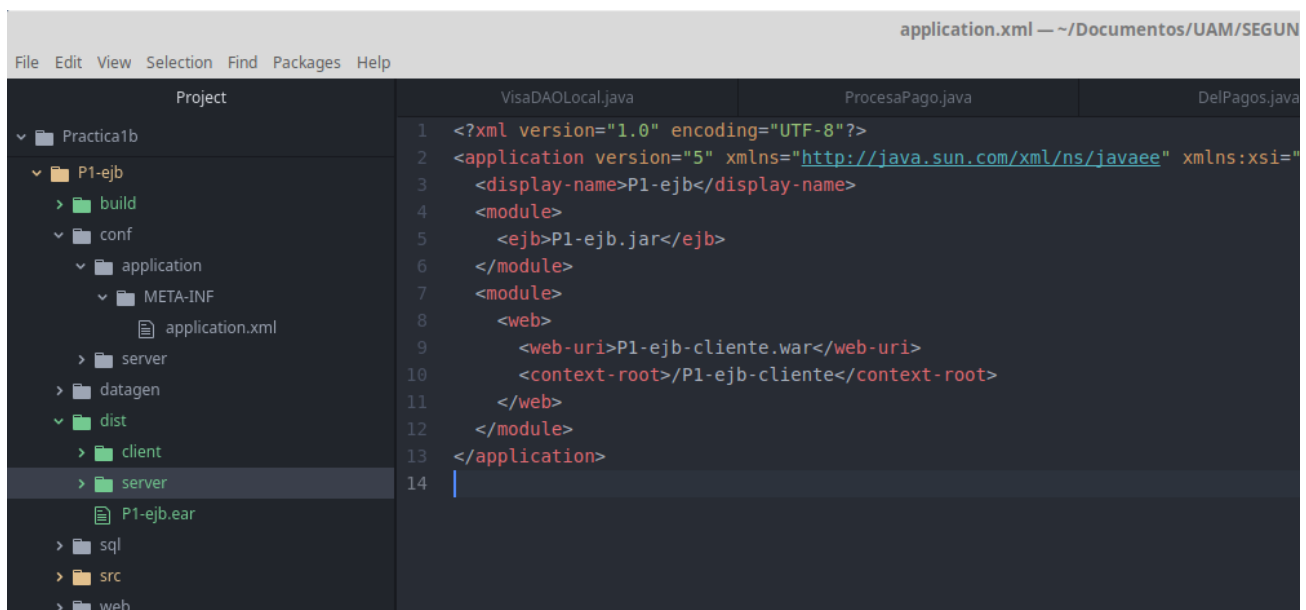
Todos los cambios se encuentran en el código que hemos modificado de la práctica. Como resumen, hemos seguido los pasos del enunciado: modificamos el servlet `ProcesaPago.java` añadiendo y eliminando unos imports, añadimos el objeto proxy al EJB local, eliminamos el webservice de la práctica anterior que estaba en un try catch y eliminamos las referencias a `BindingProvider`.

Además, hicimos los mismos cambios en los otros servlets que los necesitan, `DelPagos.java` y `GetPagos.java`, el cual hemos adaptado para que devuelva un array de `PagoBean`.

Cuestión número 2:

Editar 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 archivo `application.xml` se encuentra en el directorio `conf/application/META-INF`, el cual hace referencia a `P1-ejb.jar` y a `P1-ejb-cliente.war`



```
application.xml — ~/Documentos/UAM/SEGUN
File Edit View Selection Find Packages Help

Project
  Practica1b
    P1-ejb
      build
      conf
        application
          META-INF
            application.xml
      server
      datagen
      dist
      client
      server
      P1-ejb.ear
      sql
      src
      web

VisaDAOLocal.java
ProcesaPago.java
DelPagos.java

1 <?xml version="1.0" encoding="UTF-8"?>
2 <application version="5" xmlns="http://java.sun.com/xml/ns/javaee" xmlns:xsi="
3 <display-name>P1-ejb</display-name>
4 <module>
5 <ejb>P1-ejb.jar</ejb>
6 </module>
7 <module>
8 <web>
9 <web-uri>P1-ejb-cliente.war</web-uri>
10 <context-root>/P1-ejb-cliente</context-root>
11 </web>
12 </module>
13 </application>
14
```

Por un lado, el fichero `.war` contiene las páginas dinámicas y html, y además las clases de los servlets. El fichero `.jar` se genera cuando empaquetamos el servidor y contiene las clases del mismo y el descriptor del EJB. Finalmente, el fichero `.ear` contiene los ficheros anteriores y la descripción del `application.xml`, generándose al empaquetar la aplicación. Se adjunta la captura de dichos ficheros:

```
danist@danist-Lenovo-E51-80: ~/Documentos/UAM/SE
Archivo Editar Ver Buscar Terminal Ayuda
archivos del directorio foo/ en 'classes.jar':
jar cvfm classes.jar mymanifest -C foo/ .

danist@danist-Lenovo-E51-80:~/Documentos/UAM/SEGUNDOCUATRI/SI2/Practicalb/P1-ejb$ jar -tvf dist/client/P1-ejb-cliente.war
0 Mon Mar 15 17:52:38 CET 2021 META-INF/
125 Mon Mar 15 17:52:36 CET 2021 META-INF/MANIFEST.MF
0 Mon Mar 15 17:52:36 CET 2021 WEB-INF/
1 0 Mon Mar 15 17:49:16 CET 2021 WEB-INF/classes/
2 0 Mon Mar 15 17:51:38 CET 2021 WEB-INF/classes/ssii2/
3 0 Mon Mar 15 17:51:38 CET 2021 WEB-INF/classes/ssii2/controlador/
4 0 Mon Mar 15 17:51:38 CET 2021 WEB-INF/classes/ssii2/filtros/
5 0 Mon Mar 15 17:51:38 CET 2021 WEB-INF/classes/ssii2/visa/
0 Mon Mar 15 17:29:10 CET 2021 WEB-INF/classes/ssii2/visa/error/
0 Mon Mar 15 17:29:10 CET 2021 WEB-INF/lib/
0 Mon Mar 15 17:52:36 CET 2021 error/
2844 Mon Mar 15 17:49:16 CET 2021 WEB-INF/classes/ssii2/controlador/ComienzaPago.class
1513 Mon Mar 15 17:49:16 CET 2021 WEB-INF/classes/ssii2/controlador/DelPagos.class
1365 Mon Mar 15 17:51:38 CET 2021 WEB-INF/classes/ssii2/controlador/GetPagos.class
4919 Mon Mar 15 17:51:38 CET 2021 WEB-INF/classes/ssii2/controlador/ProcesaPago.class
1894 Mon Mar 15 17:49:16 CET 2021 WEB-INF/classes/ssii2/controlador/ServletRaiz.class
2608 Mon Mar 15 17:51:38 CET 2021 WEB-INF/classes/ssii2/filtros/CompruebaSesion.class
3170 Mon Mar 15 17:51:38 CET 2021 WEB-INF/classes/ssii2/visa/ValidadorTarjeta.class
616 Mon Mar 15 17:51:38 CET 2021 WEB-INF/classes/ssii2/visa/error/ErrorVisa.class
198 Mon Mar 15 17:51:38 CET 2021 WEB-INF/classes/ssii2/visa/error/ErrorVisaCVV.class
209 Mon Mar 15 17:51:38 CET 2021 WEB-INF/classes/ssii2/visa/error/ErrorVisaFechaCaducidad.class
207 Mon Mar 15 17:51:38 CET 2021 WEB-INF/classes/ssii2/visa/error/ErrorVisaFechaEmision.class
201 Mon Mar 15 17:51:38 CET 2021 WEB-INF/classes/ssii2/visa/error/ErrorVisaNumero.class
202 Mon Mar 15 17:51:38 CET 2021 WEB-INF/classes/ssii2/visa/error/ErrorVisaTitular.class
6043 Mon Mar 15 17:52:36 CET 2021 WEB-INF/web.xml
455 Mon Mar 15 17:52:36 CET 2021 borradoerror.jsp
501 Mon Mar 15 17:52:36 CET 2021 borradook.jsp
509 Mon Mar 15 17:52:36 CET 2021 cabecera.jsp
283 Mon Mar 15 17:52:36 CET 2021 error/muestraerror.jsp
2729 Mon Mar 15 17:52:36 CET 2021 formdatosvisa.jsp
1257 Mon Mar 15 17:52:36 CET 2021 listapagos.jsp
1178 Mon Mar 15 17:52:36 CET 2021 pago.html
1142 Mon Mar 15 17:52:36 CET 2021 pagoexito.jsp
104 Mon Mar 15 17:52:36 CET 2021 pie.html
5011 Mon Mar 15 17:52:36 CET 2021 testbd.jsp
danist@danist-Lenovo-E51-80:~/Documentos/UAM/SEGUNDOCUATRI/SI2/Practicalb/P1-ejb$ jar -tvf dist/server/P1-ejb.jar
0 Mon Mar 15 17:30:24 CET 2021 META-INF/
125 Mon Mar 15 17:30:22 CET 2021 META-INF/MANIFEST.MF
0 Mon Mar 15 17:29:14 CET 2021 ssii2/
0 Mon Mar 15 17:29:14 CET 2021 ssii2/visa/
0 Mon Mar 15 17:29:14 CET 2021 ssii2/visa/dao/
255 Mon Mar 15 17:30:22 CET 2021 META-INF/sun-ejb-jar.xml
1464 Mon Mar 15 17:29:14 CET 2021 ssii2/visa/PagoBean.class
856 Mon Mar 15 17:29:14 CET 2021 ssii2/visa/TarjetaBean.class
593 Mon Mar 15 17:29:14 CET 2021 ssii2/visa/VisaDAOLocal.class
1925 Mon Mar 15 17:29:14 CET 2021 ssii2/visa/dao/DBTester.class
6977 Mon Mar 15 17:29:14 CET 2021 ssii2/visa/dao/VisaDAOBean.class
danist@danist-Lenovo-E51-80:~/Documentos/UAM/SEGUNDOCUATRI/SI2/Practicalb/P1-ejb$ jar -tvf dist/P1-ejb.ear
0 Mon Mar 15 17:55:46 CET 2021 META-INF/
125 Mon Mar 15 17:55:44 CET 2021 META-INF/MANIFEST.MF
508 Sat Feb 11 23:33:00 CET 2012 META-INF/application.xml
20953 Mon Mar 15 17:52:36 CET 2021 P1-ejb-cliente.war
7083 Mon Mar 15 17:30:22 CET 2021 P1-ejb.jar
danist@danist-Lenovo-E51-80:~/Documentos/UAM/SEGUNDOCUATRI/SI2/Practicalb/P1-ejb$
```

Ejercicio 3:

Preparar los PCs con el esquema descrito y realizar el despliegue de la aplicación:

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

Desplegar la aplicación de empresa

Ahora las IPS son 10.2.1.2 para el cliente y servidor y 10.2.1.1 para la base de datos, luego modificamos el build.properties y postgre.sql como primer paso:

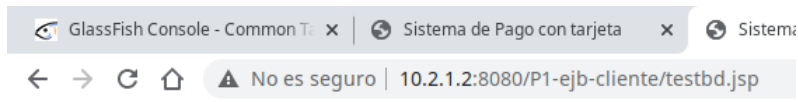
- as.host.client = 10.2.1.2
- as.host.server = 10.2.1.2
- db.host = 10.2.1.1
- db.client.host = 10.2.1.2

Desplegamos la aplicación con “ant desplegar” correctamente y pasamos al ejercicio siguiente para demostrar que verdaderamente funciona.

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-ejbcliente.

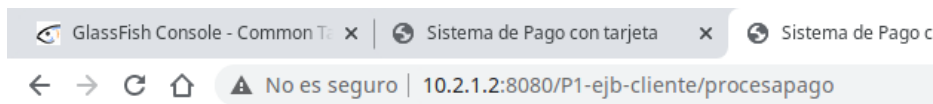
Primero realizamos un pago con testbd.jsp:



Pago con tarjeta

Proceso de un pago

Id Transacción:	<input type="text" value="1"/>
Id Comercio:	<input type="text" value="1"/>
Importe:	<input type="text" value="12"/>
Numero de visa:	<input type="text" value="1111 2222 3333 4444"/>
Titular:	<input type="text" value="Jose Garcia"/>
Fecha Emisión:	<input type="text" value="11/09"/>
Fecha Caducidad:	<input type="text" value="11/22"/>
CVV2:	<input type="text" value="123"/>
Modo debug:	<input 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 type="radio"/> False
<input type="button" value="Pagar"/>	



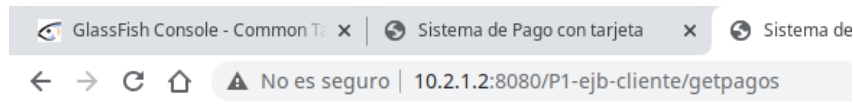
Pago con tarjeta

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

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

[Volver al comercio](#)

Luego comprobamos que se haya realizado accediendo a la lista de pagos:



Pago con tarjeta

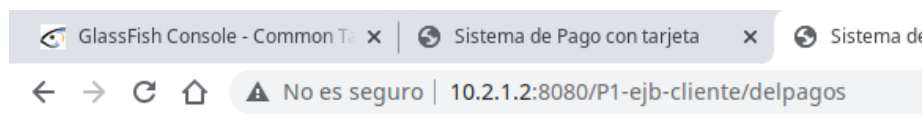
Lista de pagos del comercio 1

idTransaccion	Importe	codRespuesta	idAutorizacion
1	12.0	000	1

[Volver al comercio](#)

Prácticas de Sistemas Informáticos II

Por último, eliminamos el pago:



Pago con tarjeta

Se han borrado 1 pagos correctamente para el comercio 1

[Volver al comercio](#)

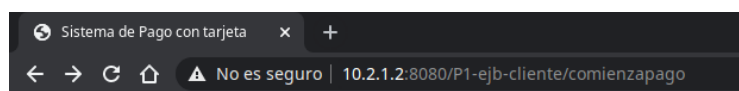
Prácticas de Sistemas Informáticos II

A continuación, realizamos el mismo pago, pero en pago.html y en modo incógnito:

Id Transacción:

Id Comercio:

Importe:



Pago con tarjeta

Numero de visa:

Titular:

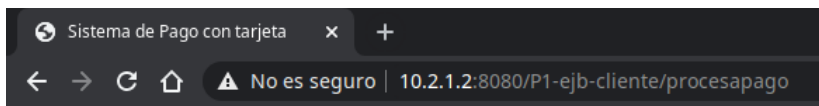
Fecha Emisión:

Fecha Caducidad:

CVV2:

Id Transacción: 2
Id Comercion: 2
Importe: 45.0

Prácticas de Sistemas Informáticos II



Pago con tarjeta

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

idTransaccion: 2
idComercio: 2
importe: 45.0
codRespuesta: 000
idAutorizacion: 1

[Volver al comercio](#)

Prácticas de Sistemas Informáticos II

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 creado la interfaz VisaDAORemote a partir de VisaDAOLocal pero poniéndole la etiqueta @Remote. También implementamos VisaDAOBean como interfaz remota y hemos hecho que PagoBean y TarjetaBean sean serializables. Volvemos a desplegar la aplicación como en el ejercicio anterior y hemos visto que funciona correctamente.

```
@Remote
public interface VisaDAORemote {
    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);
}

public class PagoBean implements Serializable {

public class TarjetaBean implements Serializable {

@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-EJBcliente remoto y 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 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.

Tras realizar todos los cambios del enunciado, volvemos a desplegar la aplicación para el cliente remoto en la dirección IP 10.2.1.1 y pasamos a realizar un pago y comprobar que verdaderamente se ha realizado:

Correo :: Entrada x | Curso: SISTEMAS INFORMÁTICOS x | 10.2.1.1:8080/P1-ej

← → ↻ 🏠 No es seguro | 10.2.1.1:8080/P1-ejb-cliente-remoto/

Id Transacción:

Id Comercio:

Importe:

Correo :: Entrada x | Curso: SISTEMAS INFORMÁTICOS x | Sistema de Pago con tarjeta

← → ↻ 🏠 No es seguro | 10.2.1.1:8080/P1-ejb-cliente-remoto/comienzapago

Pago con tarjeta

Numero de visa:

Titular:

Fecha Emisión:

Fecha Caducidad:

CVV2:

Id Transacción: 1
Id Comercio: 1
Importe: 12.0

Prácticas de Sistemas Informáticos II

Correo :: Entrada x | Curso: SISTEMAS INFORMÁTICOS x | Sistema de Pago con tarjeta

← → ↻ 🏠 No es seguro | 10.2.1.1:8080/P1-ejb-cliente-remoto/procesapago

Pago con tarjeta

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

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

[Volver al comercio](#)

Prácticas de Sistemas Informáticos II

Ejercicio 7:

Modificar la aplicación VISA para soportar el campo saldo.

Hemos seguido los pasos del enunciado:

- Añadimos el atributo saldo a TarjetaBean.java como double, así como sus métodos.
- Declaramos dos statements para hacer un select y un update del saldo en la base de datos.

```
79 private static final String SELECT_SALDO_TARJETA_QRY =  
80     "select saldo from tarjeta " +  
81     "where numeroTarjeta=? ";  
82 private static final String UPDATE_SALDO_TARJETA_QRY =  
83     "update tarjeta " +  
84     "set saldo = ? " +  
85     "where numeroTarjeta=? ";  
86
```

- Modificamos el método realizaPago y el servlet ProcesaPago para adaptarlos al saldo.

Ejercicio 8:

Desplegar y probar la nueva aplicación creada.

Todos los saldos están inicializados a 1000. De modo que a continuación podemos observar cómo ha disminuido el saldo al realizar el pago.

1. Realizamos un pago: le quitamos 500 euros a Jose García.

eliminar usuarios linux mint - B x Correo :: Enviados: Re: Duda pr x 10.2.1.2:8080/f

← → ↻ 🏠 ⚠ No es seguro | 10.2.1.2:8080/P1-ejb-cliente/testbd.jsp

Pago con tarjeta

Proceso de un pago

Id Transacción:

Id Comercio:

Importe:

Numero de visa:

Titular:

Fecha Emisión:

Fecha Caducidad:

CVV2:

Modo debug: ☐ True ☐ False

Direct Connection: ☐ True ☐ False

Use Prepared: ☐ True ☐ False

2. Pago realizado con éxito:

eliminar usuarios linux mint - B x Correo :: Enviados: Re: Duda pr x 10.2.1.2:8080/f

← → ↻ 🏠 ⚠ No es seguro | 10.2.1.2:8080/P1-ejb-cliente/procesapago

Pago con tarjeta

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

idTransaccion: 1
idComercio: 1
importe: 500.0
codRespuesta: 000
idAutorizacion: 1

[Volver al comercio](#)

3. Miramos la base de datos y comprobamos que, de 1000 euros, la cuenta de Jose García ha bajado 500 euros:

Result

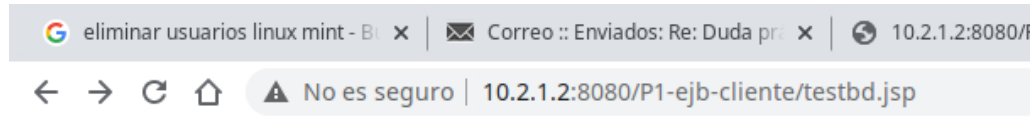
Execution plan

Visualize

Logging

#	^	numerotarjeta	titular	validadesde	validahasta	codigoverificacion	saldo
1		1111 2222 3333 4444	Jose Garcia	11/09	11/22	123	500

4. Realizamos una operación con identificador de transacción y de comercio duplicados para luego ver que ha sido erróneo. Ahora utilizamos la cuenta de Gabriel Avila.



Pago con tarjeta

Proceso de un pago

Id Transacción:	<input type="text" value="1"/>
Id Comercio:	<input type="text" value="1"/>
Importe:	<input type="text" value="300"/>
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="01/22"/>
CVV2:	<input type="text" value="207"/>
Modo debug:	<input type="radio"/> True <input type="radio"/> False
Direct Connection:	<input type="radio"/> True <input type="radio"/> False
Use Prepared:	<input type="radio"/> True <input type="radio"/> False
<input type="button" value="Pagar"/>	

5. Comprobamos que el pago es erróneo:



Pago con tarjeta

Pago incorrecto

Prácticas de Sistemas Informáticos II

6. Vemos en la base de datos que su saldo no ha variado pues sigue a mil.

Result	Execution plan Visualize Logging						
#	^	numerotarjeta	titular	validadesde	validahasta	codigoverificacion	saldo
1		2347 4840 5058 7931	Gabriel Avila Locke	11/09	01/22	207	1000

Ejercicio 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.

The screenshot shows the GlassFish Server Open Source Edition administration console. The left sidebar contains a tree view with the following structure:

- Common Tasks
- Domain
 - server (Admin Server)
 - Clusters
 - Standalone Instances
 - Nodes
 - Applications
 - Lifecycle Modules
 - Monitoring Data
 - Resources
 - Concurrent Resources
 - Connectors
 - JDBC
 - JMS Resources
 - Connection Factories (selected)
 - Destination Resources
 - JNDI
 - JavaMail Sessions
 - Resource Adapter Configs
 - Configurations
 - default-config
 - server-config
 - Update Tool

The main content area is titled 'New JMS Connection Factory'. It includes a description: '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: *
- Resource Type:
- Description:
- Status: ☒ Enabled

Pool Settings

- Initial and Minimum Pool Size: Connections
Minimum and initial number of connections maintained in the pool
- Maximum Pool Size: Connections
Maximum number of connections that can be created to satisfy client requests
- Pool Resize Quantity: Connections
Number of connections to be removed when pool idle timeout expires
- Idle Timeout: Seconds
Maximum time that connection can remain idle in the pool
- Max Wait Time: Milliseconds
Amount of time caller waits before connection timeout is sent
- On Any Failure: ☐ Close All Connections
Close all connections and reconnect on failure, otherwise reconnect only when used
- Transaction Support:
Level of transaction support. Overwrite the transaction support attribute in the Resource Adapter in a downward compatible way.
- Connection Validation: ☐ Required
Validate connections, allow server to reconnect in case of failure

Ejercicio 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.

The screenshot shows the GlassFish Server Open Source Edition administration console. The left sidebar contains a tree view with the following structure:

- Common Tasks
- Domain
 - server (Admin Server)
 - Clusters
 - Standalone Instances
 - Nodes
 - Applications
 - Lifecycle Modules
 - Monitoring Data
 - Resources
 - Concurrent Resources
 - Connectors
 - JDBC
 - JMS Resources
 - Connection Factories
 - Destination Resources (selected)
 - JNDI
 - Configurations
 - default-config
 - server-config
 - Update Tool

The main content area is titled 'Edit JMS Destination Resource'. It includes a description: 'Editing a Java Message Service (JMS) destination resource also modifies the associated admin object resource.'

Load Defaults

- JNDI Name:
- Physical Destination Name *
Destination name in the Message Queue broker. If the destination does not exist, it will be created automatically when needed.
- Resource Type: *
- Deployment Order:
Specifies the loading order of the resource at server startup. Lower numbers are loaded first.
- Description:
- Status: ☒ Enabled

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. Incluya en la clase VisaCancelacionJMSBean: (...)

Modificamos el fichero sun-ejb-jar.xml: para ello simplemente añadimos la siguiente sentencia:

```
<jndi-name>jms/VisaConnectionFactory</jndi-name>
```

Añadimos las sentencias SQL en VisaCancelacionJMSBean:

```
1  *
2  @MessageDriven(mappedName = "jms/VisaPagosQueue")
3  public class VisaCancelacionJMSBean extends DBTester implements MessageListener {
4      static final Logger logger = Logger.getLogger("VisaCancelacionJMSBean");
5      @Resource
6      private MessageDrivenContext mdc;
7
8      private static final String UPDATE_CANCELA_QRY = "update pago set codRespuesta = 999 where idAutorizacion=?";
9      private static final String DESHACER_PAGO_QRY = "update tarjeta "+
10         "set saldo = saldo + importe "+
11         " from pago where pago.idAutorizacion=? "+
12         " and pago.numeroTarjeta = tarjeta.numeroTarjeta";
13
14 }
```

Implementamos el método onMessage() para implementar ambas actualizaciones SQL y con el control de errores:

```
1  public void onMessage(Message inMessage) {
2      TextMessage msg = null;
3      Connection con = null;
4      PreparedStatement pstmt = null;
5      int id;
6      try {
7          if (inMessage instanceof TextMessage) {
8              msg = (TextMessage) inMessage;
9              logger.info("MESSAGE BEAN: Message received: " + msg.getText());
10
11              id = Integer.parseInt(msg.getText());
12
13              // Obtener conexion
14              con = getConnection();
15
16              // Actualizamos el codigo de respuesta del pago
17              logger.info(UPDATE_CANCELA_QRY);
18              pstmt = con.prepareStatement(UPDATE_CANCELA_QRY);
19              pstmt.setInt(1, id);
20              pstmt.execute();
21
22              // Deshacemos el pago
23              logger.info(DESHACER_PAGO_QRY);
24              pstmt = con.prepareStatement(DESHACER_PAGO_QRY);
25              pstmt.setInt(1, id);
26              pstmt.execute();
27
28          } else {
29              logger.warning("
30                  "Message of wrong type: "
31                  + inMessage.getClass().getName());
32          }
33      } catch (JMSException e) {
34          e.printStackTrace();
35          mdc.setRollbackOnly();
36      } catch (Throwable te) {
37          te.printStackTrace();
38      }
39  }
```

Ejercicio 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.

Añadimos los Resources en las variables queue y connectionFactory:

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

Y luego hacemos la búsqueda mediante JNDI, lo dejamos también comentado en el código. En este método JNDI damos transparencia de ubicación ya que no es necesario saber la ubicación de los recursos.

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

connection = connectionFactory.createConnection();
session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE);
```

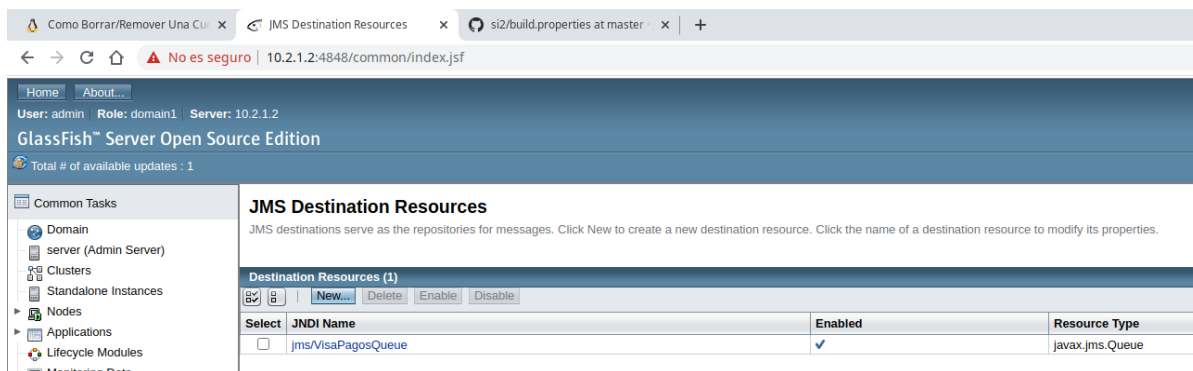
Ejercicio 13:

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

Borre desde la consola de administración de Glassfish la connectionFactory y la cola creadas manualmente y ejecute: `cd P1-jms` ant todo Compruebe en la consola de administración del Glassfish que, efectivamente, los recursos se han creado automáticamente. Incluye una captura de pantalla, donde se muestre la consola de administración con los recursos creados. Revise el fichero jms.xml y anote en la memoria de prácticas cuál es el comando equivalente para crear una cola JMS usando la herramienta asadmin.

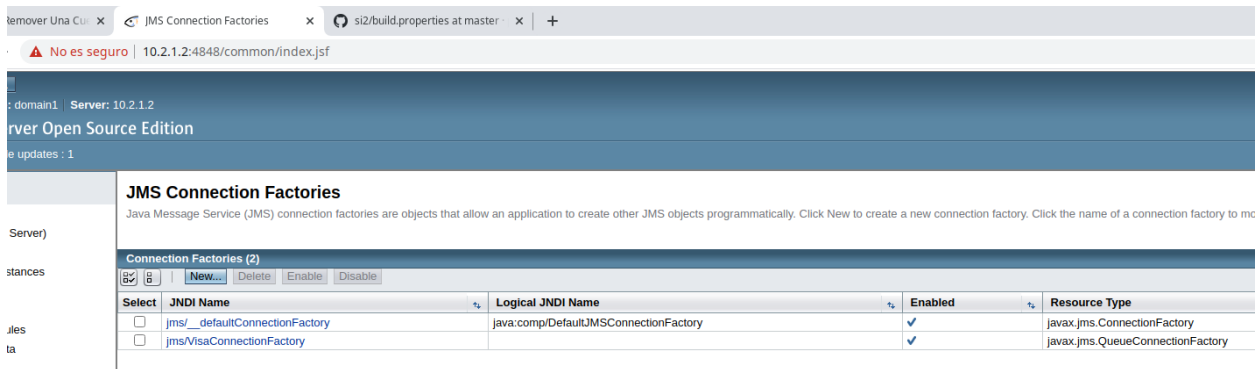
Como nos piden, modificamos los archivos build.properties y jms.properties. Asignamos la IP 10.2.1.2 al as.host.mdb porque es la IP donde hemos ido trabajando y donde está ubicada la aplicación. Además, en el fichero jms.properties le asignamos el VisaPagosQueue al jsm.name, el VisaConnectionFactory a jms.factoryname, Visa a jms.physname y la IP 10.2.1.2 al as.host.server pues es básicamente la IP que usamos para el servidor.

Ejecutamos los comandos del enunciado y vemos que la cola de mensajes se ha creado en el admin:



The screenshot shows the GlassFish Server Open Source Edition administration console. The browser address bar indicates the URL is `10.2.1.2:4848/common/index.jsf`. The page title is "GlassFish™ Server Open Source Edition". The user is logged in as "admin" with the role "domain1" on server "10.2.1.2". The "JMS Destination Resources" section is active, showing a table with one resource: "jms/VisaPagosQueue". The table has columns for "Select", "JNDI Name", "Enabled", and "Resource Type". The "jms/VisaPagosQueue" resource is selected and enabled, with a resource type of "javax.jms.Queue".

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



El comando final sería:

```
asadmin --user admin --passwordfile passwordfile --host 10.2.1.2 --port 4848 create-jms-resource --restype javax.jms.QueueConnectionFactory --enabled=true --property Name=VisaPagosQueue jms/VisaPagosQueue
```

Ejercicio 14:

Importante: Detenga la ejecución del MDB con la consola de administración para poder realizar satisfactoriamente el siguiente ejercicio (check de 'Enabled' en Applications/P1-jmsmdb y guardar los cambios).

En primer lugar, modificamos el fichero VisaQueueMessageProducer.java para implementar los argumentos como mensajes como podemos ver en la siguiente captura:

```

81         } else {
82             messageProducer = session.createProducer(queue);
83             textMessage = session.createTextMessage();
84             textMessage.setText(args[0]);
85             System.out.println("Mensaje: " + textMessage.getText());
86             messageProducer.send(textMessage);
87             messageProducer.close();
88             session.close();
89             connection.close();
90         }
91     } catch (Exception e) {

```

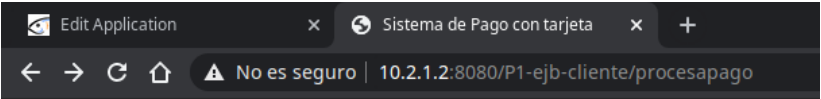
Luego ejecutamos el cliente con el comando del enunciado con la IP 10.2.1.2 y verificamos el contenido de la cola con el segundo comando, dando como resultado:

```

Mensaje: 1
danist@danist-Lenovo-E51-80:/$ ./opt/glassfish4/glassfish/bin/appclient -targetserver 10.2.1.2 -client ~/Documentos/UAM/SEGUNDOCUATRI/SI2/Practicalb/P1-jms/dist/clientjms/P1-jms-clientjms.jar 1
mar 19, 2021 11:52:15 AM org.hibernate.validator.internal.util.Version <clinit>
INFO: HV000001: Hibernate Validator 5.1.2.Final
mar 19, 2021 11:52:15 AM com.sun.messaging.jms.ra.ResourceAdapter start
INFORMACIÓN: MQJMSRA RA1101: GlassFish MQ JMS Resource Adapter: Version: 5.1.1 (Build 2-c) Compile: March 17 2015 1045
mar 19, 2021 11:52:15 AM com.sun.messaging.jms.ra.ResourceAdapter start
INFORMACIÓN: MQJMSRA RA1101: GlassFish MQ JMS Resource Adapter starting: broker is REMOTE, connection mode is TCP
mar 19, 2021 11:52:15 AM com.sun.messaging.jms.ra.ResourceAdapter start
INFORMACIÓN: MQJMSRA RA1101: GlassFish MQ JMS Resource Adapter Started:REMOTE
Mensaje: 1
danist@danist-Lenovo-E51-80:/$ ./opt/glassfish4/glassfish/bin/appclient -targetserver 10.2.1.2 -client ~/Documentos/UAM/SEGUNDOCUATRI/SI2/Practicalb/P1-jms/dist/clientjms/P1-jms-clientjms.jar -browse
mar 19, 2021 11:52:43 AM org.hibernate.validator.internal.util.Version <clinit>
INFO: HV000001: Hibernate Validator 5.1.2.Final
mar 19, 2021 11:52:43 AM com.sun.messaging.jms.ra.ResourceAdapter start
INFORMACIÓN: MQJMSRA RA1101: GlassFish MQ JMS Resource Adapter: Version: 5.1.1 (Build 2-c) Compile: March 17 2015 1045
mar 19, 2021 11:52:43 AM com.sun.messaging.jms.ra.ResourceAdapter start
INFORMACIÓN: MQJMSRA RA1101: GlassFish MQ JMS Resource Adapter starting: broker is REMOTE, connection mode is TCP
mar 19, 2021 11:52:43 AM com.sun.messaging.jms.ra.ResourceAdapter start
INFORMACIÓN: MQJMSRA RA1101: GlassFish MQ JMS Resource Adapter Started:REMOTE
Mensajes en cola:
1
Total time: 6 seconds
danist@danist-Lenovo-E51-80:/$

```

A continuación, realizamos un pago desde la aplicación web y comprobamos en la base de datos que se haya retirado el dinero correctamente:



Pago con tarjeta

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

idTransaccion: 1
idComercio: 1
importe: 200.0
codRespuesta: 000
idAutorizacion: 1

Result	Execution plan Visualize Logging							
#	^	idautorizacion	idtransaccion	codrespuesta	importe	idcomercio	numerotarjeta	fecha
1	1		1	000	200	1	1111 2222 3333 4444	19/03/21 03:56
Row: 2								

Result	Execution plan Visualize Logging						
#	^	numerotarjeta	titular	validadesde	validahasta	codigoverificacion	saldo
1		1111 2222 3333 4444	Jose Garcia	11/09	11/22	123	800

Lo cancelamos desde el cliente y comprobamos que su saldo vuelve a estar normal en la base de datos:

Result

Execution plan

Visualize

Logging

#	^	idautorizacion	idtransaccion	codrespuesta	importe	idcomercio	numerotarjeta	fecha
1	1		1	999	200	1	1111 2222 3333 4444	19/03/21 03:56

Row: 1

Result

Execution plan

Visualize

Logging

#	^	numerotarjeta	titular	validadesde	validahasta	codigoverificacion	saldo
1		1111 2222 3333 4444	Jose Garcia	11/09	11/22	123	1000