

**Escuela Politécnica Superior
Ingeniería Informática
Prácticas de Sistemas Informáticos 2**

Grupo	2401	Práctica	1b	Fecha	19/03/2021
Alumno/a	De las Heras Moreno, Martín				
Alumno/a	Valderrábano Zamorano, Santiago Manuel				

Práctica 1b: Arquitectura de JAVA EE

Cuestión número 1:

Las librerías son:

```
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import javax.ejb.Local;
```

Las única anotación que hay es `@Local` encima de la clase VisaDAOLocal, que indica que la clase va a estar en el mismo emplazamiento que el cliente.

Ejercicio 1:

Introducimos en VisaDAOBean las siguientes modificaciones como se indica en el enunciado:

```
import javax.ejb.Stateless;
@Stateless(mappedName="VisaDAOBean")
```

Aquí indicamos que la clase VisaDAO además de extender a DBTester, implementa la interfaz VisaDAOLocal:

```
public class VisaDAOBean extends DBTester implements VisaDAOLocal {
```

y además borramos el constructor por defecto y modificamos el método getPagos para que devuelva un array y así coincida con la definición VisaDAOLocal de la siguiente forma:

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

```
PagoBean[] ret = null;
ret = new PagoBean[pagos.size()];
ret = pagos.toArray(ret);
return ret;
```

Ejercicio número 2:

En este ejercicio realizamos las siguientes modificaciones en **todos los servelets**:

1. Añadimos los nuevos imports:

```
import javax.ejb.EJB;
import ssii2.visa.*;  
(Ya que esto incluye VisaDAOLocal)
```

2. Añadimos el atributo para acceder al EJB Local:

```
@EJB(name="VisaDAOBean", beanInterface=VisaDAOLocal.class)
private VisaDAOLocal dao;
```

3. Comentamos los siguientes imports:

```
// import javax.xml.ws.BindingProvider;
// import javax.xml.ws.WebServiceRef
// import javax.xml.ws.WebServiceException;
```

4. Comentamos las siguientes líneas:

```
// String rutaServicio = getServletContext().getInitParameter("rutaServicio");
//
// BindingProvider bp = (BindingProvider) dao;
// bp.getRequestContext().put(BindingProvider.ENDPOINT_ADDRESS_PROPERTY, rutaServicio);
```

Cuestión número 2:

El archivo application.xml se encuentra en ./conf/application/META-INF/application.xml. En este fichero se encuentra la información necesaria para el despliegue de la aplicación.

Para ver el contenido de los archivos .ear .jar y .war, ejecutuamos los siguientes comandos:

```
jar -tvf dist/P1-ejb.ear // jar -tvf dist/client/P1-ejb-cliente.war
```

The terminal window shows the following output:

```
martin@martin-XPS-13:~/Documents/si2/practica1b/P1-ejb$ jar -tvf dist/P1-ejb.ear
  0 Fri Mar 19 11:57:52 CET 2021 META-INF/
125 Fri Mar 19 11:57:56 CET 2021 META-INF/MANIFEST.MF
508 Sat Feb 11 23:33:08 CET 2021 META-INF/application.xml
21161 Fri Mar 19 11:57:26 CET 2021 P1-ejb-cliente.war
  670 Fri Mar 19 10:48:04 CET 2021 P1-ejb.jar

martin@martin-XPS-13:~/Documents/si2/practica1b/P1-ejb$ jar -tvf dist/client/P1-ejb-cliente.war
  0 Fri Mar 19 11:57:28 CET 2021 META-INF/
125 Fri Mar 19 11:57:26 CET 2021 META-INF/MANIFEST.MF
  0 Fri Mar 19 11:57:26 CET 2021 WEB-INF/
  0 Fri Mar 19 11:55:40 CET 2021 WEB-INF/classes/
  0 Fri Mar 19 11:56:26 CET 2021 WEB-INF/classes/ssii2/
  0 Fri Mar 19 11:56:26 CET 2021 WEB-INF/classes/ssii2/controlador/
  0 Fri Mar 19 11:56:26 CET 2021 WEB-INF/classes/ssii2/filtros/
  0 Fri Mar 19 11:56:26 CET 2021 WEB-INF/classes/ssii2/vista/
  0 Fri Mar 19 11:56:26 CET 2021 WEB-INF/classes/ssii2/vista/error/
  0 Fri Mar 19 10:58:42 CET 2021 WEB-INF/lib/
  0 Fri Mar 19 11:57:26 CET 2021 error/
2991 Fri Mar 19 11:55:40 CET 2021 WEB-INF/classes/ssii2/controlador/ComienzaPago.class
1513 Fri Mar 19 11:55:40 CET 2021 WEB-INF/classes/ssii2/controlador/DelPagos.class
1365 Fri Mar 19 11:56:26 CET 2021 WEB-INF/classes/ssii2/controlador/GetPagos.class
4982 Fri Mar 19 11:56:26 CET 2021 WEB-INF/classes/ssii2/controlador/ProcesaPago.class
1894 Fri Mar 19 11:55:40 CET 2021 WEB-INF/classes/ssii2/controlador/ServletRaiz.class
2608 Fri Mar 19 11:56:26 CET 2021 WEB-INF/classes/ssii2/filtros/CompruebaSesion.class
3170 Fri Mar 19 11:56:26 CET 2021 WEB-INF/classes/ssii2/vista/ValidadorTarjeta.class
616 Fri Mar 19 11:56:26 CET 2021 WEB-INF/classes/ssii2/vista/error/ErrorVisa.class
198 Fri Mar 19 11:56:26 CET 2021 WEB-INF/classes/ssii2/vista/error/ErrorVisaOV.class
209 Fri Mar 19 11:56:26 CET 2021 WEB-INF/classes/ssii2/vista/error/ErrorVisaFechaCaducidad.class
207 Fri Mar 19 11:56:26 CET 2021 WEB-INF/classes/ssii2/vista/error/ErrorVisaFechaEmision.class
201 Fri Mar 19 11:56:26 CET 2021 WEB-INF/classes/ssii2/vista/error/ErrorVisaNumero.class
202 Fri Mar 19 11:56:26 CET 2021 WEB-INF/classes/ssii2/vista/error/ErrorVisaTitular.class
6262 Fri Mar 19 11:57:26 CET 2021 WEB-INF/web.xml
455 Fri Mar 19 11:57:26 CET 2021 borradorerror.jsp
501 Fri Mar 19 11:57:26 CET 2021 borradook.jsp
509 Fri Mar 19 11:57:26 CET 2021 cabecera.jsp
283 Fri Mar 19 11:57:26 CET 2021 error/nuestraerror.jsp
2729 Fri Mar 19 11:57:26 CET 2021 formdatosvista.jsp
1257 Fri Mar 19 11:57:26 CET 2021 listapagos.jsp
1178 Fri Mar 19 11:57:26 CET 2021 pago.html
1142 Fri Mar 19 11:57:26 CET 2021 pagoxeoxtio.jsp
104 Fri Mar 19 11:57:26 CET 2021 pie.html
5011 Fri Mar 19 11:57:26 CET 2021 testbd.jsp

martin@martin-XPS-13:~/Documents/si2/practica1b/P1-ejb$ jar -tvf dist/server/P1-ejb.jar
```

```
jar -tvf dist/server/P1-ejb.jar
```

The terminal window shows the following output:

```
martin@martin-XPS-13:~/Documents/si2/practica1b/P1-ejb$ jar -tvf dist/server/P1-ejb.jar
  0 Fri Mar 19 12:02:40 CET 2021 META-INF/
125 Fri Mar 19 12:02:38 CET 2021 META-INF/MANIFEST.MF
  0 Fri Mar 19 11:53:02 CET 2021 ssii2/
  0 Fri Mar 19 11:53:02 CET 2021 ssii2/vista/
  0 Fri Mar 19 11:53:02 CET 2021 ssii2/vista/dao/
255 Fri Mar 19 10:48:04 CET 2021 META-INF/sun-ejb-jar.xml
1464 Fri Mar 19 11:53:02 CET 2021 ssii2/vista/PagoBean.class
  856 Fri Mar 19 11:53:02 CET 2021 ssii2/vista/TarjetaBean.class
  593 Fri Mar 19 11:53:02 CET 2021 ssii2/vista/VisaDAOLocal.class
1953 Fri Mar 19 11:53:02 CET 2021 ssii2/vista/dao/DBTester.class
  7042 Fri Mar 19 11:53:02 CET 2021 ssii2/vista/dao/VisaDAOBean.class
```

Ejercicio número 3:

Para desplegar nuestra aplicación, debemos indicar en los ficheros .properties donde se encuentran alojados BD, cliente y sevidor.

En build.properties indicamos que tanto as.host.client como as.host.server son igual a 10.1.7.2 ya que tanto cliente como servidor se encontrarán en el mismo servidor. Esto se debe a que hora estamos accediendo a la interfaz local del Enterprise JavaBean (VisaDAOLocal).

En postgresql.properties indicamos donde se encuentra la base de datos (db.host) y donde se encuentra la interfaz de acceso a la ella (db.client.host). Según está descrito en el esquema del enunciado, la base de datos se encuentra en db.host = 10.1.7.1 y la interfaz de acceso en db.client.host = 10.1.7.2

Ejercicio número 4:

En primer lugar, comprobamos que se ha desplegado correctamente todo. Para ello accedemos a la interfaz de acceso a la BD que se encuentra en <http://10.1.7.2:4848> como hemos definido en el ejercicio anterior.

Select	Name	Deployment Order	Enabled	Engines	Action
	P1-ejb	100	✓	ear, ejb, web	Redeploy Reload

Después, comprobamos el correcto funcionamiento de la aplicación:

1. Desde pago.html

Id Transacción:	<input type="text" value="1"/>
Id Comercio:	<input type="text" value="1"/>
Importe:	<input type="text" value="50"/>
<input type="button" value="Envia Datos Pago"/>	



Pago con tarjeta

Numero de visa:

Titular:

Fecha Emisión:

Fecha Caducidad:

CVV2:

Id Transacción: 1

Id Comercio: 1

Importe: 50.0

Prácticas de Sistemas Informáticos II



Pago con tarjeta

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

idTransaccion: 1

idComercio: 1

importe: 60.0

codRespuesta: 000

idAutorizacion: 1

[Volver al comercio](#)

Prácticas de Sistemas Informáticos II

2. Desde testbd.jsp:

Not secure | 10.1.7.2:8080/P1-ejb-cliente/testbd.jsp

Pago con tarjeta

Proceso de un pago

Id Transacción:	<input type="text" value="2"/>
Id Comercio:	<input type="text" value="1"/>
Importe:	<input type="text" value="50"/>
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 type="radio"/> False
Use Prepared:	<input type="radio"/> True <input type="radio"/> False
<input type="button" value="Pagar"/>	

Not secure | 10.1.7.2:8080/P1-ejb-cliente/procesapago

Pago con tarjeta

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

```
idTransaccion: 2
idComercio: 1
importe: 50.0
codRespuesta: 000
idAutorizacion: 2
```

[Volver al comercio](#)

alumnodb@visa.10.1.7.1:5432 [8.4.10] Schema Browser

Tables Views Indexes Sequences Code

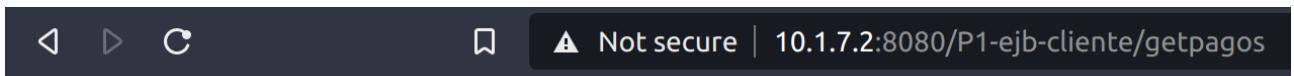
Columns Indexes Constraints Triggers Data Information

Table Name

pago

tarjeta

	idautorizacion	idtransaccion	codrespuesta	importe	idcomercio	numerotarjeta	fecha
1	1	1	000	60	1	1111 2222 3333 4444	19/03/21 04:25
2	2	2	000	50	1	1111 2222 3333 4444	19/03/21 04:26



Pago con tarjeta

Lista de pagos del comercio 1

idTransaccion	Importe	codRespuesta	idAutorizacion
1	60.0	000	1

[Volver al comercio](#)

Prácticas de Sistemas Informáticos II



Pago con tarjeta

Se han borrado 1 pagos correctamente para el comercio 1

[Volver al comercio](#)

Prácticas de Sistemas Informáticos II

Ejercicio número 5:

Para este ejercicio vamos a implementar un cliente remoto en EJB. Para ello, como se indica en el enunciado, usaremos como material de partida P1-ejb para implementar el servidor remoto.

En primer lugar, ejecutamos el comando ***cp P1-ejb -pr P1-ejb-servidor-remoto.***

A continuación, realizamos los siguientes cambios escritos en el enunciado:

- Copiar el fichero VisaDAOLocal.java al fichero VisaDAORemote.java.
- Cambiar, en VisaDAORemote.java, el nombre de la interfaz a VisaDAORemote y cambiar la anotación @Local por @Remote.
- Añadir import javax.ejb.Remote; y quitar import javax.ejb.Local
- Hacer que VisaDAOBean implemente ambas interfaces, la local y la remota.

Después, tenemos que serializar tanto TarjetaBean con PagoBean. Para ello, modificamos los ficheros .java correspondientes añadiendo:

- *import java.io.Serializable*
- *public class TarjetaBean implements Serializable*
- *public class PagoBean implements Serializable*

Por último, compilamos, empaquetamos y desplegamos la aplicación P1-ejb de la misma forma que en el ejercicio 3.

Ejercicio número 6:

En este ejercicio vamos a implementar el cliente remoto de EJB. Para ello, como se indica en el enunciado, usaremos como material de partida P1-base. A continuación, realizamos las siguientes modificaciones:

1. Cambiar en build.properties el nombre de la aplicación a P1-ejb-cliente-remoto
2. Eliminar el directorio P1-ejb-cliente-remoto/src/ssii2/visa/dao puesto que a partir de ahora la lógica se invocará de forma remota
3. Serializar tanto TarjetaBean como PagoBean de la misma forma que en el ejercicio anterior.
4. Copiar la interfaz VisaDAORemote.java que hemos creado en el ejercicio anterior a P1-ejb-cliente-remoto/src/ssii2/visa
5. Insertar en cada servelet la declaración del VisaDAORemote junto con la anotación en la que se especifica que la variable conecta con EJB como se indica en el enunciado, además de los imports necesarios.
6. Comentar la declaración VisaDAO dao en los servelets.
7. Crear el fichero glassfish-web.xml en web/WEB_INF con el contenido especificado en el enunciado únicamente modificando la dirección IP por 10.1.7.2
8. Cambiar las direcciones IP de build.properties por as.host=10.1.7.1 y postgresql.properties por db.host=10.1.7.1 y db.client.host=10.1.8.2
9. **Extra:** En ProcesaPago.java insertamos un if(pago == null) tras hacer dao.realizaPago(pago) ya que generaba un error al compilar.

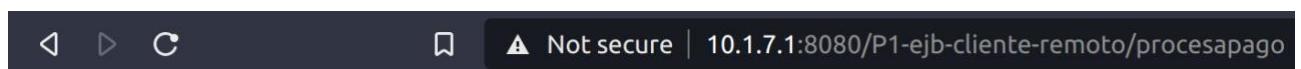
Para terminar, compilamos y desplegamos y nos conectamos a la aplicación del cliente para comprobar el correcto funcionamiento de todo.

A continuación incluimos las capturas que prueba dicho funcionamiento:

The screenshot shows a web page with the title "Pago con tarjeta". Below it, a section titled "Proceso de un pago" contains various input fields and radio buttons. The fields include:

- Id Transacción: 1
- Id Comercio: 1
- Importe: 75
- Numero de visa: 1111 2222 3333 4444
- Titular: Jose Garcia
- Fecha Emisión: 11/09
- Fecha Caducidad: 11/22
- CVV2: 123
- Modo debug: True False
- Direct Connection: True False
- Use Prepared: True False

At the bottom is a "Pagar" button.



[Volver al comercio](#)



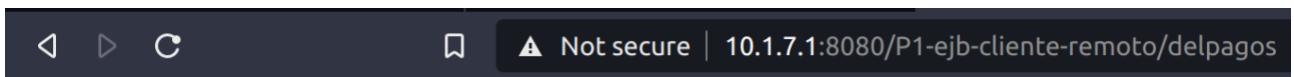
Pago con tarjeta

Lista de pagos del comercio 1

idTransaccion	Importe	codRespuesta	idAutorizacion
1	75.0	000	1

[Volver al comercio](#)

Prácticas de Sistemas Informáticos II



Pago con tarjeta

Error en el borrado de pagos para el comercio 1

[Volver al comercio](#)

Prácticas de Sistemas Informáticos II

Ejercicio número 7:

En este ejercicio, vamos a modificar la aplicación Visa para que soporte el campo saldo. Para ello, realizamos los siguientes pasos que se indican en el enunciado:

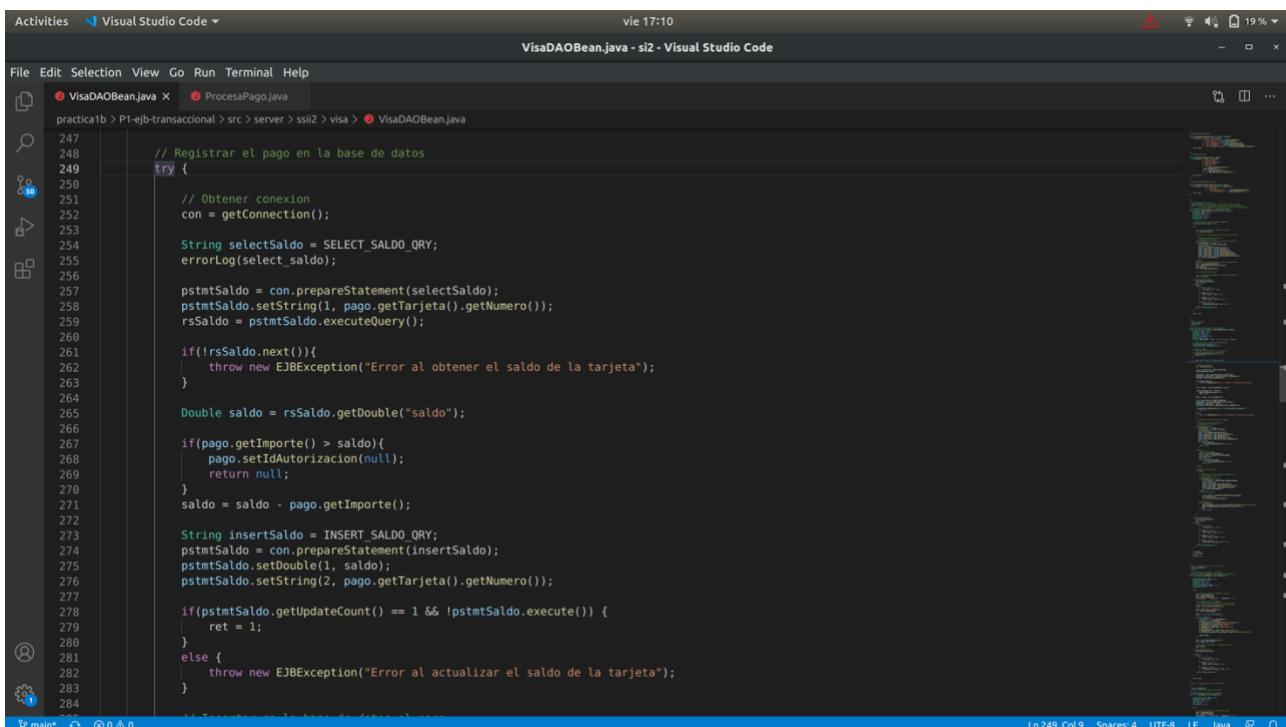
1. cp P1-ejb -r P1-ejb-transaccional
2. cd P1-ejb-transaccional
3. tar -xzvf /ruta/a/P1-ejb-transaccional-base.tgz
4. ant limpiar-todo
5. En TarjetaBean.java añadidos el campo **private double saldo** además de sus **métodos get y set**.
6. En VisaDAOBean.java,
 - 6.1. Importamos el nuevo tipo de excepción a tener en cuenta (import javax.ejb.EJBException;)

6.2. Programamos las siguientes cosas:

6.2.1. Prepared statements que se piden

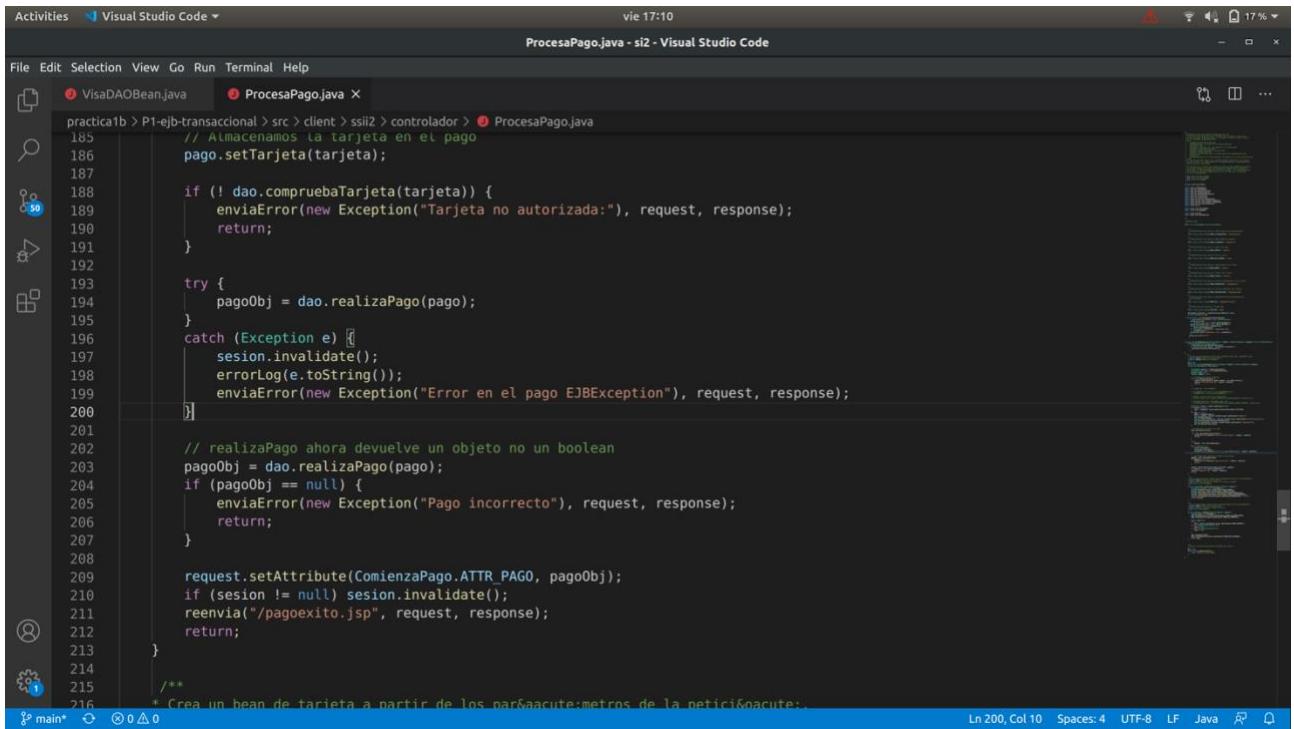
```
J VisaDAOBean.java X J ProcesaPago.java
practica1b > P1-ejb-transaccional > src > server > ssii2 > visa > J VisaDAOBean.java
  74   "insert into pago( " +
  75     "idTransaccion, importe, idComercio, numeroTarjeta) " +
  76     " values (?,?,?,?,?);"
  77
  78   private static final String SELECT_PAGO_TRANSACCION_QRY =
  79     "select idAutorizacion, codRespuesta " +
  80     " from pago " +
  81     " where idTransaccion = ?" +
  82     " and idComercio = ?;";
  83
  84   // Prepared statements del saldo
  85
  86   private static final String SELECT_SALDO_QRY =
  87     "select saldo " +
  88     "from tarjeta " +
  89     "where numeroTarjeta=?";
  90
  91   private static final String INSERT_SALDO_QRY =
  92     "update tarjeta " +
  93     "set saldo=? " +
  94     "where numeroTarjeta=?";
  95   /*****
```

6.2.2. Modificar la función realizaPago:



```
Activities Visual Studio Code vie 17:10
VisaDAOBean.java - si2 - Visual Studio Code
File Edit Selection View Go Run Terminal Help
VisaDAOBean.java X ProcesaPago.java
practica1b > P1-ejb-transaccional > src > server > ssii2 > visa > VisaDAOBean.java
  247   // Registrar el pago en la base de datos
  248   try {
  249
  250     // Obtener conexion
  251     con = getConnection();
  252
  253     String selectSaldo = SELECT_SALDO_QRY;
  254     errorLog(selectSaldo);
  255
  256     pstmtSaldo = con.prepareStatement(selectSaldo);
  257     pstmtSaldo.setString(1, pago.getTarjeta().getNumero());
  258     rsSaldo = pstmtSaldo.executeQuery();
  259
  260     if(!rsSaldo.next()){
  261       throw new EJBException("Error al obtener el saldo de la tarjeta");
  262     }
  263
  264     Double saldo = rsSaldo.getDouble("saldo");
  265
  266     if(pago.getImporte() > saldo){
  267       pago.setIdAutorizacion(null);
  268       return null;
  269     }
  270
  271     saldo = saldo - pago.getImporte();
  272
  273     String insertSaldo = INSERT_SALDO_QRY;
  274     pstmtSaldo = con.prepareStatement(insertSaldo);
  275     pstmtSaldo.setDouble(1, saldo);
  276     pstmtSaldo.setString(2, pago.getTarjeta().getNumero());
  277
  278     if(pstmtSaldo.executeUpdate() == 1 && !pstmtSaldo.execute()) {
  279       ret = 1;
  280     }
  281     else {
  282       throw new EJBException("Error al actualizar el saldo de la tarjeta");
  283     }
  284
  285   } catch (SQLException e) {
  286     e.printStackTrace();
  287   }
  288 }
```

7. En ProcesaPago.java:



The screenshot shows the Visual Studio Code interface with the file 'ProcesaPago.java' open. The code implements a controller for a payment process. It starts by setting a card in the payment object, then checks if the card is authorized. If not, it sends an error and returns. Otherwise, it tries to perform the payment. If successful, it sets an attribute in the request and invalidates the session. If the payment fails, it sends an error and returns. Finally, it creates a bean from the payment parameters.

```
Activities Visual Studio Code vie 17:10
File Edit Selection View Go Run Terminal Help
VisaDAOBean.java ProcesaPago.java
practicatb > P1-ejb-transacional > src > client > ss12 > controlador > ProcesaPago.java
185 // Almacenamos la tarjeta en el pago
186 pago.setTarjeta(tarjeta);
187
188 if (! dao.compruebaTarjeta(tarjeta)) {
189     enviaError(new Exception("Tarjeta no autorizada:"), request, response);
190     return;
191 }
192
193 try {
194     pagoObj = dao.realizaPago(pago);
195 } catch (Exception e) {
196     sesion.invalidate();
197     errorLog(e.toString());
198     enviaError(new Exception("Error en el pago EJBException"), request, response);
199 }
200
201 // realizaPago ahora devuelve un objeto no un boolean
202 pagoObj = dao.realizaPago(pago);
203 if (pagoObj == null) {
204     enviaError(new Exception("Pago incorrecto"), request, response);
205     return;
206 }
207
208 request.setAttribute(ComienzaPago.ATTR_PAGO, pagoObj);
209 if (sesion != null) sesion.invalidate();
210 reenvia("/pagoxito.jsp", request, response);
211 return;
212 }
213
214 /**
215 * Crea un bean de tarjeta a partir de los parámetros de la petición.
216 */
Ln 200, Col 10 Spaces: 4 UTF-8 LF Java
```

Ejercicio número 8:

En este ejercicio comprobamos el correcto funcionamiento de lo realizado en el ejercicio 7. A

continuación, adjuntamos las capturas que lo demuestran:

Prueba 1:

1. Pagos correctos

1.1. Comprobar saldo

1.2. Realizar pago

1.3. Comprobar saldo

The screenshot shows a dual-pane interface. The left pane is a 'Schema Browser' titled 'alumnodb@visa.10.1.7.1:5432 [8.4.10] Schema Browser - Untitled'. It displays a table named 'pago' with two rows of data. The right pane is a 'SQL Editor' titled 'alumnodb@visa.10.1.7.1:5432 [8.4.10] SQL Editor - Untitled'. It shows the same table 'pago' with the same data.

Table Name	Columns	Indexes	Constraints	Triggers	Data	Information
pago	numerotarjeta				titular	validadesde
tarjeta	1 1111 2222 3333 4444				Jose Garcia	11/09
	2 2347 4840 5058 7931				Gabriel Avil...	11/09

The screenshot shows a payment form on a web page. The URL in the address bar is 'Not secure | 10.1.7.2:8080/P1-ejb-cliente/'. The form fields are:

- Id Transacción:
- Id Comercio:
- Importe:
-



The screenshot shows a payment form on a web page. The form fields are:

- Numero de visa:
- Titular:
- Fecha Emisión:
- Fecha Caducidad:
- CVV2:
-

Id Transacción: 1

Id Comercio: 1

Importe: 333.0

Pago con tarjeta

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

idTransaccion: 1
idComercio: 1
importe: 333.0
codRespuesta: 000
idAutorizacion: 1

[Volver al comercio](#)

Prácticas de Sistemas Informáticos II

Table Name	Columns	Indexes	Constraints	Triggers	Data	Information
pago	numerotarjeta	titular	validadesde	validahasta	codigoverificacion	saldo
tarjeta	84 6632 8822 8048 ... Clodoveo G...	05/10	05/22	154	1000	
	85 2000 7653 7205 ... John Pelaez...	07/09	07/22	934	1000	
	86 1111 2222 3333 ... Jose Garcia	11/09	11/22	123	667	
	87 1899 8476 1549 ... Luis Mojam...	05/09	01/22	370	1000	

2. Identificador de transacción y de comercio duplicados

2.1. Comprobar saldo

2.2. Realizar pagos

2.3. Comprobar saldo

Pago con tarjeta

Numero de visa:

Titular:

Fecha Emisión:

Fecha Caducidad:

CVV2:

Id Transacción: 1

Id Comercio: 1

Importe: 100.0

Prácticas de Sistemas Informáticos II



Pago con tarjeta

Pago incorrecto

Prácticas de Sistemas Informáticos II

Table Name	Columns	Indexes	Constraints	Triggers	Data	Information
pago	numerotarjeta	titular	validadesde	validahasta	codigoverificacion	saldo
tarjeta	84 6632 8822 8048 ... Clodoveo G...	05/10	05/22	154	1000	
	85 2000 7653 7205 ... John Pelaez...	07/09	07/22	934	1000	
	86 1111 2222 3333 ... Jose Garcia	11/09	11/22	123	667	
	87 1899 8476 1549 ... Luis Mojamo...	05/09	01/22	370	1000	
	88 3503 8882 5400 ... Restituta Av.	07/10	06/22	392	1000	

Ejercicio número 9:

En este ejercicio declaramos una factoría de conexiones de forma manual accediendo al formulario mostrado en la captura. Accedemos a este desde la consola de administración y lo rellenamos según se indica en el enunciado.

The screenshot shows the GlassFish administration console with the following details:

- Left Sidebar:** Shows navigation links for Activities, New JMS Connection Factory, Home, About, User: admin, Role: domain1, Server: 10.1.7.2, and GlassFish Server Open Source Edition.
- Top Bar:** Displays the date (vie 19:21), battery level (98%), and other system icons.
- Main Content:** A dialog titled "New JMS Connection Factory".
 - General Settings:** JNDI Name: `jms/VisaConnectionFactory`, Resource Type: `javax.jms.QueueConnectionFactory`, Description: "Factoría de conexiones a la cola de pagos", Status: Enabled.
 - Pool Settings:** Initial and Minimum Pool Size: 8 Connections, Maximum Pool Size: 32 Connections, Pool Resize Quantity: 2 Connections, Idle Timeout: 300 Seconds, Max Wait Time: 60000 Milliseconds.
 - On Any Failure:** Close All Connections (unchecked).
 - Transaction Support:** A dropdown menu showing options like `XA`, `Local`, and `No Transaction`.
 - Connection Validation:** Required (unchecked).
- Bottom Buttons:** OK and Cancel buttons.

JMS Connection Factories

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

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

Ejercicio número 10:

En este ejercicio creamos una cola de mensajes manualmente de la misma forma que en el ejercicio 9:

New JMS Destination Resource

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

JNDI Name: **Physical Destination Name:** **Resource Type:** **Description:** **Status:** Enabled

Additional Properties (0)

Add Property	Delete Properties		
Select	Name	Value	Description

No items found.

The screenshot shows the GlassFish administration interface. The left sidebar contains a tree view of server components: Common Tasks, Domain, Clusters, Standalone Instances, Nodes, Applications, Lifecycle Modules, Monitoring Data, Resources (with sub-options like Concurrent Resources, Connectors, JDBC, JMS Resources, JNDI, JavaMail Sessions, and Resource Adapter Configs), Configurations, and Update Tool. The main content area is titled "JMS Destination Resources". It includes a brief description: "JMS destinations serve as the repositories for messages. Click New to create a new destination resource. Click the name of a destination resource to modify its properties." Below this is a table titled "Destination Resources (1)". The table has columns: Select, JNDI Name, Enabled, Resource Type, and Description. There is one row with the value "jms/VisaPagosQueue" in the JNDI Name column, checked in the Enabled column, "javax.jms.Queue" in the Resource Type column, and "Cola de pagos VISA" in the Description column.

Ejercicio número 11:

En este ejercicio vamos a implementar un servicio que utilice la factoría de conexiones y la cola de mensajes para cancelar pagos a partir del idAutorizacion.

Partimos del código que se encuentra en Moodle llamado P1-jms-base.

En primer lugar, modificamos el fichero sun-ejb-jr.xml añadiendo lo especificado en el enunciado para conectarse correctamente a la conexión Factory.

```

sun-ejb-jar.xml X VisaCancelacionJMSBean.java
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://java.sun.com/xml/ns/javaee/sun-ejb-jar_3_0.dtd">
3  <sun-ejb-jar>
4      <enterprise-beans>
5          <ejb>
6              <ejb-name>VisaCancelacionJMSBean</ejb-name>
7              <mdb-connection-factory>
8                  <jndi-name>VisaConnectionFactory</jndi-name>
9                  </mdb-connection-factory>
10             </ejb>
11         </enterprise-beans>
12     </sun-ejb-jar>
13

```

A continuación, añadimos 2 queries a **VisaCancelacionJMSBean.java** que se encargan una poner el codRepuesta del pago a 999 y otra de modificar el saldo de la tarjeta con la que se realizó el pago que se ha cancelado y modificamos la función onMessage de **VisaCancelacionJMSBean.java** para que haga las llamadas necesarias a las queries declaradas anteriormente

```

VisaQueueMessageProducer.java  VisaCancelacionJMSBean.java X
practica1b > P1-jms > src > mdb > ssi2 > visa > VisaCancelacionJMSBean.java
25  /*
26  * @MessageDriven(mappedName = "jms/VisaPagosQueue")
27  public class VisaCancelacionJMSBean extends DBTester implements MessageListener {
28      static final Logger logger = Logger.getLogger("VisaCancelacionJMSBean");
29      @Resource
30      private MessageDrivenContext mdc;
31
32      private static final String UPDATE_CANCELAR_QRY = "update pago " +
33          "set codRespuesta=999 " +
34          "where pago.idAutorizacion=?";
35      private static final String UPDATE_RECTIFICA_QRY = "update tarjeta " +
36          "set saldo=? " +
37          "where tarjeta.numeroTarjeta=?";
38
39
40      public VisaCancelacionJMSBean() {
41      }
42
43      // TODO : Método onMessage de ejemplo
44      // Modificarlo para ejecutar el UPDATE definido más arriba,
45      // asignando el idAutorizacion a lo recibido por el mensaje
46      // Para ello conecte a la BD, prepareStatement() y ejecute correctamente
47      // la actualización
48      public void onMessage(Message inMessage) {
49          PreparedStatement pstmt = null;
50          TextMessage msg = null;
51          Connection con = null;
52          ResultSet rs = null;
53          String numTarjeta = "";
54          Double importe = 0;
55          Double saldo = 0;
56
57          try {
58
59              con = getConnection();
60              if (inMessage instanceof TextMessage) {
61                  msg = (TextMessage) inMessage;
62                  logger.info("MESSAGE BEAN: Message received: " + msg.getText());
63
64                  String query = UPDATE_CANCELAR_QRY;
65                  String idAutorizacion = msg.getText();
66
67                  pstmt = con.prepareStatement(query);
68                  pstmt.setInt(1, Integer.parseInt(idAutorizacion));
69                  pstmt.executeUpdate();
70
71                  query = "select numeroTarjeta, importe " +
72                      "from Pago " +
73                      "where pago.idAutorizacion=?";
74
75                  pstmt = con.prepareStatement(query);
76                  pstmt.setInt(1, Integer.parseInt(idAutorizacion));
77                  rs = pstmt.executeQuery();
78
79                  if (rs.next()) {
80                      importe = rs.getDouble("importe");
81                      tarjeta = rs.getString("numeroTarjeta");
82                  } else {
83                      throw new EJBException("Cancelacion no realizada: id no autorizado");
84                  }
85
86                  query = "select saldo " +
87                      "from tarjeta " +
88                      "where numeroTarjeta=?";
89
90                  pstmt = con.prepareStatement(query);
91                  pstmt.setString(1, tarjeta);
92                  rs = pstmt.executeQuery();
93
94                  if (rs.next()) {
95                      saldo = rs.getDouble("saldo");
96                  } else {
97                      throw new EJBException("Cancelacion anulada: no se encuentra el saldo de la tarjeta");
98                  }
99
100                 saldo = saldo + importe;
101
102                 query = UPDATE_RECTIFICA_QRY;
103                 pstmt = con.prepareStatement(query);
104                 pstmt.setDouble(1, saldo);
105                 pstmt.setString(2, tarjeta);
106                 pstmt.executeUpdate();
107
108
109

```

Estas modificaciones realizan lo siguiente.

1. Actualizar el código de respuesta a 999.
2. Rectificar el saldo del cliente que realizó el pago que se ha cancelado.

Ejercicio número 12:

En este ejercicio desarrollamos los dos métodos posibles de acceder a los objetos de factoría de conexiones y cola de mensajes de clientes, creados en los ejercicios 9 y 10 respectivamente, desde el cliente. El primer método consiste en declarar ambos objetos y utilizar anotaciones.

```
① VisaQueueMessageProducer.java ×  
practica1b > P1-jms > src > clientjms > ssi2 > ① VisaQueueMessageProducer.java  
 9  import javax.jms.JMSException;  
10 import javax.annotation.Resource;  
11 import javax.jms.QueueBrowser;  
12 import java.util.Enumeration;  
13 import javax.naming.InitialContext;  
14  
15 public class VisaQueueMessageProducer {  
16  
17     // TODO: Anotar los siguientes objetos para  
18     // conectar con la connection factory y con la cola  
19     // definidas en el enunciado  
20     @Resource(mappedName = "jms/NombreDeLaConnectionFactory")  
21     private static ConnectionFactory connectionFactory;  
22     @Resource(mappedName = "jms/NombreDeLaCola")  
23     private static Queue queue;  
24
```

El segundo método consiste usar la API JNDI de la siguiente forma:

```
① VisaQueueMessageProducer.java ×  
practica1b > P1-jms > src > clientjms > ssi2 > ① VisaQueueMessageProducer.java  
 63  
 64     if (args.length != 1) {  
 65         System.err.println("Uso: VisaQueueMessageProducer [-browse | <msg>]");  
 66         return;  
 67     }  
 68  
 69     try {  
 70         // TODO: Inicializar connectionFactory  
 71  
 72         // InitialContext jndi = new InitialContext();  
 73         // connectionFactory = (ConnectionFactory)jndi.lookup("jms/VisaConnectionFactory");  
 74         // queue = (Queue)jndi.lookup("jms/VisaPagosQueue");  
 75  
 76         // y queue mediante JNDI  
 77  
 78         connection = connectionFactory.createConnection();  
 79         session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE);  
 80         if (args[0].equals("-browse")) {  
 81             browseMessages(session);  
 82         } else {  
 83             // TODO: Enviar argv[0] como mensaje de texto  
 84         }  
 85     } catch (Exception e) {  
 86         System.out.println("Excepcion : " + e.toString());  
 87     } finally {  
 88         if (connection != null) {  
 89             try {  
 90                 connection.close();  
 91             } catch (JMSException e) {  
 92             }  
 93         } // if
```

Las ventajas del método JMS dinámico, respecto al estático, son las siguientes:

1. Al ser dinámico, se podrían introducir cambios en tiempo de ejecución
2. Aunque en ambos hay que definir en la instancia los nombres de los recursos, en el primer método es una etiqueta fija mientras que en el segundo se busca el recurso durante la ejecución por lo que puedes declarar varios recursos e ir usándolos según se necesiten.

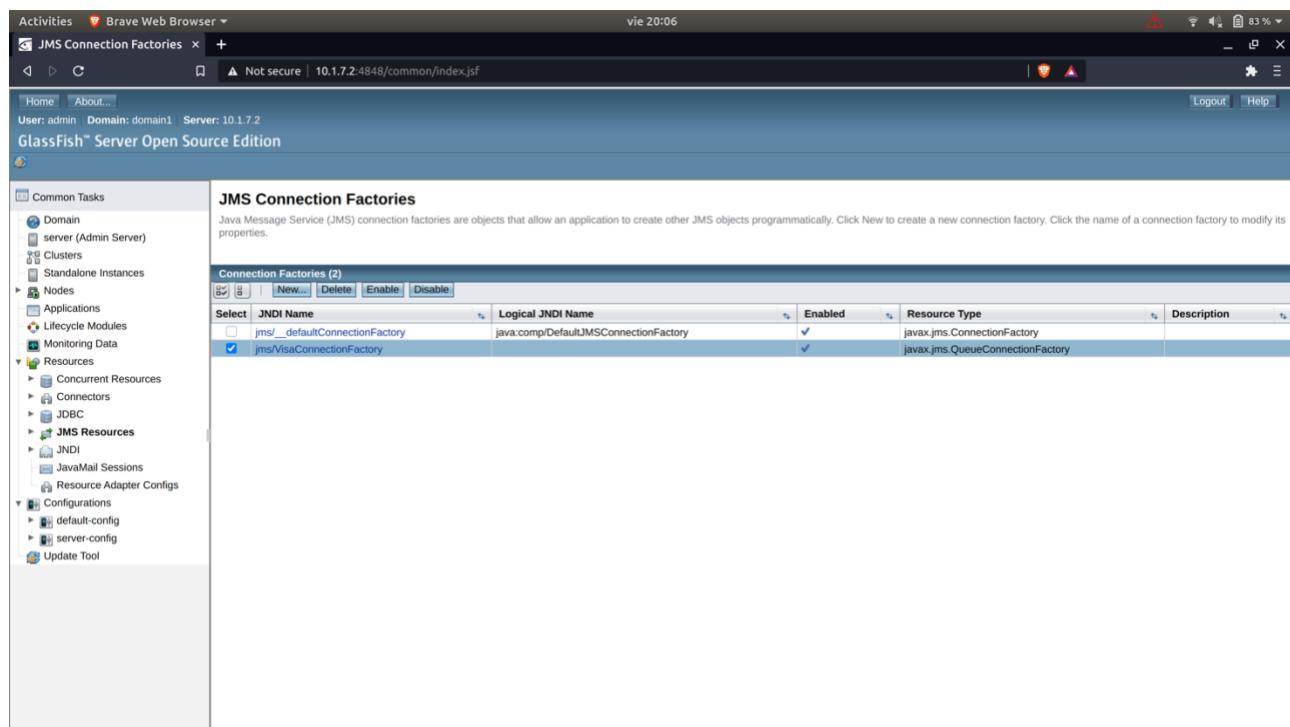
La principal desventaja de JMS dinámico a través de la API JNDI es que introduce una mayor complejidad en el sistema.

Ejercicio número 13:

En este ejercicio vamos a automatizar la creación de los recursos JMS. Para ello, modificamos en primer lugar los ficheros

- jms.properties
 - as.host.server = 10.1.7.2
 - as.host.client = 10.1.7.2
 - jms.factoryname = jms/VisaConnectionFactory
 - jms.name = jms/VisaPagosQueue
 - jms.physname = VisaPagosQueue
- build.properties
 - as.host.mdb = 10.1.7.2

Para probar que esto funciona correctamente, eliminamos manualmente los recursos a través de la consola de administración y después ejecutamos **ant todo** y comprobamos que se han creado bien los recursos:



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

Activities Brave Web Browser vie 20:06

JMS Connection Factories +

Not secure | 10.1.7.2:4848/common/index.jsf

User: admin Domain: domain1 Server: 10.1.7.2

GlassFish® Server Open Source Edition

Common Tasks

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

JMS Connection Factories

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

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

Activities Brave Web Browser vie 20:07

JMS Destination Resource +

Not secure | 10.1.7.2:4848/common/index.jsf

User: admin Domain: domain1 Server: 10.1.7.2

GlassFish® Server Open Source Edition

Common Tasks

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

JMS Destination Resources

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

Select	JNDI Name	Enabled	Resource Type	Description
<input checked="" type="checkbox"/>	jms/VisaPagosQueue	✓	javax.jms.Queue	Cola de pagos VISA

Activities Brave Web Browser vie 20:07

JMS Destination Resource + Not secure | 10.1.7.2:4848/common/index.jsf

User: admin Domain: domain1 Server: 10.1.7.2 Logout Help

GlassFish™ Server Open Source Edition

Common Tasks

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

JMS Destination Resources

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

Destination Resources (0)

New... Delete Enable Disable

Select	JNDI Name	Status	Enabled	Resource Type	Description
No items found.					

Después de **ant todo:**

Activities Brave Web Browser vie 20:28

JMS Connection Factories + Not secure | 10.1.7.2:4848/common/index.jsf

User: admin Domain: domain1 Server: 10.1.7.2 Logout Help

GlassFish™ Server Open Source Edition

Total # of available updates : 1

Common Tasks

- Domain
- server (Admin Server)
- Clusters
- Standalone Instances
- Nodes
- Applications
 - P1-ejb
- Lifecycle Modules
- Monitoring Data
- Resources
 - Concurrent Resources
 - Connectors
 - JDBC
 - JMS Resources
 - JNDI
 - JavaMail Sessions
 - Resource Adapter Configs
- Configurations
 - default-config
 - server-config
- Update Tool

JMS Connection Factories

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

Connection Factories (2)

New... Delete Enable Disable

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

Adjuntamos una captura del fichero **jms.xml** mostrando el comando para crear una cola JMS usando la herramienta asadmin:

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

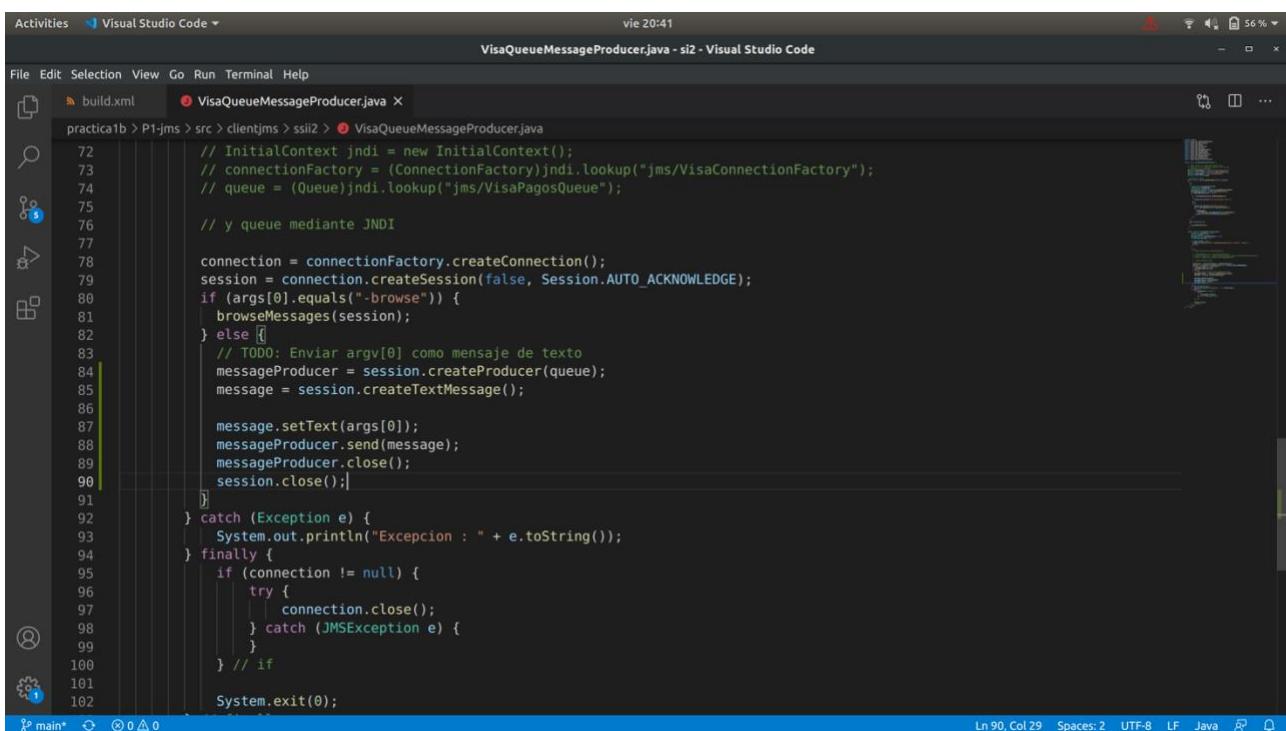
Ejercicio número 14:

En este ejercicio lo primero que hacemos es detener la ejecución de MDB en la consola de administración desactivando la casilla **Enabled** en Applications/P1-jms-mdb.

The screenshot shows the GlassFish Administration Console interface. On the left, there's a navigation tree with categories like Domain, Clusters, Standalone Instances, Nodes, Applications, Resources, and Configurations. Under Applications, 'P1-jms-mdb' is selected. The main panel has tabs for General and Descriptor. In the General tab, there's an 'Edit Application' form. The 'Name:' field contains 'P1-jms-mdb'. The 'Status:' section has two checkboxes: 'Enabled' (unchecked) and 'Implicit CDI' (checked). Below that, 'Location:' is set to '\$(com.sun.aas.instanceRootURI)/applications/P1-jms-mdb/' and 'Deployment Order:' is set to '100'. There are also 'Libraries:' and 'Description:' fields. A success message 'New values successfully saved.' with a green checkmark is shown above the form. At the bottom, there's a table titled 'Modules and Components (2)' with two rows. The first row has 'Module Name' as 'P1-jms-mdb', 'Engines' as '[ejb, weld]', 'Component Name' as '-----', and 'Type' as '-----'. The second row has 'Module Name' as 'P1-jms-mdb', 'Engines' as '[ejb, weld]', 'Component Name' as 'VisaCancelacionJMSBean', and 'Type' as 'MessageDrivenBean'. The 'Action' column is empty.

Module Name	Engines	Component Name	Type	Action
P1-jms-mdb	[ejb, weld]	-----	-----	
P1-jms-mdb	[ejb, weld]	VisaCancelacionJMSBean	MessageDrivenBean	

Después, modificamos el fichero **VisaQueueProducer.java** para implementar el vío de args[0] como mensaje de texto. Añadimos el siguiente código usando como referencia el Apéndice 2.1:



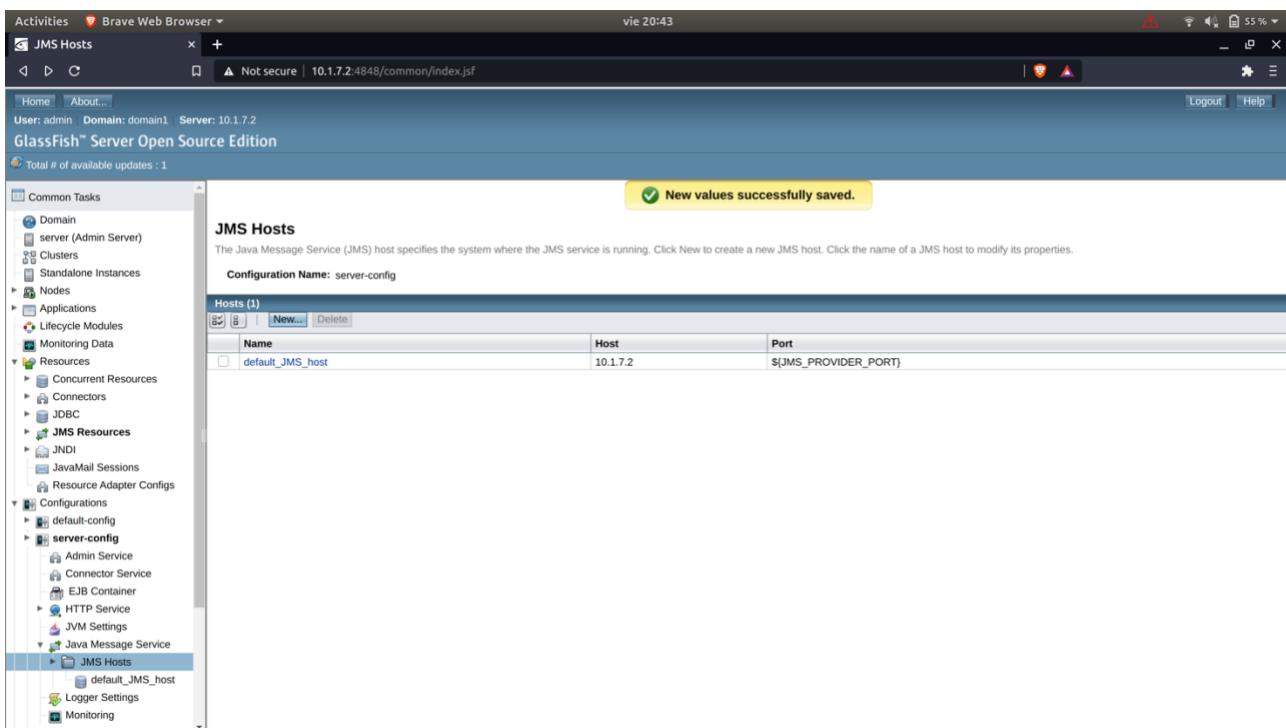
```

Activities  Visual Studio Code  vie 20:41
File Edit Selection View Go Run Terminal Help
build.xml  VisaQueueMessageProducer.java
practica1b > P1-jms > src > clientjms > ssi2 > VisaQueueMessageProducer.java
72     // InitialContext jndi = new InitialContext();
73     // connectionFactory = (ConnectionFactory)jndi.lookup("jms/VisaConnectionFactory");
74     // queue = (Queue)jndi.lookup("jms/VisaPagosQueue");
75
76     // y queue mediante JNDI
77
78     connection = connectionFactory.createConnection();
79     session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE);
80     if (args[0].equals("-browse")) {
81         browseMessages(session);
82     } else {
83         // TODO: Enviar argv[0] como mensaje de texto
84         messageProducer = session.createProducer(queue);
85         message = session.createTextMessage();
86
87         message.setText(args[0]);
88         messageProducer.send(message);
89         messageProducer.close();
90         session.close();
91     }
92 } catch (Exception e) {
93     System.out.println("Excepcion : " + e.toString());
94 } finally {
95     if (connection != null) {
96         try {
97             connection.close();
98         } catch (JMSEException e) {
99         }
99     } // if
100
101     System.exit(0);
102

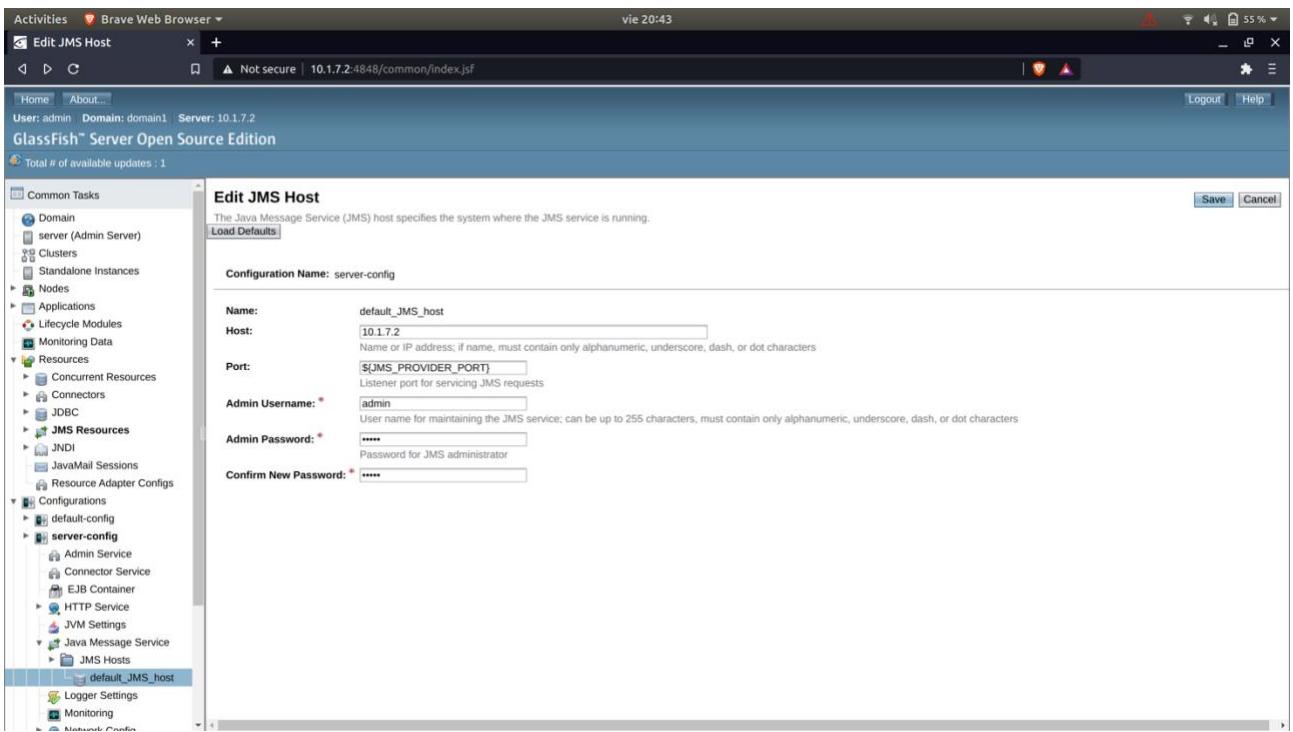
```

A continuación, ejecutamos el cliente en el PC de la siguiente forma:

1. Modificamos la variable `default_JMS_host`, desde la consola de administración, de **localhost** a **10.1.7.2**.



Name	Host	Port
default_JMS_host	10.1.7.2	\$JMS_PROVIDER_PORT



2. Detenemos la ejecución de MDB desde la consola de administración.

3. Ejecutamos los comandos que se indican en el enunciado:

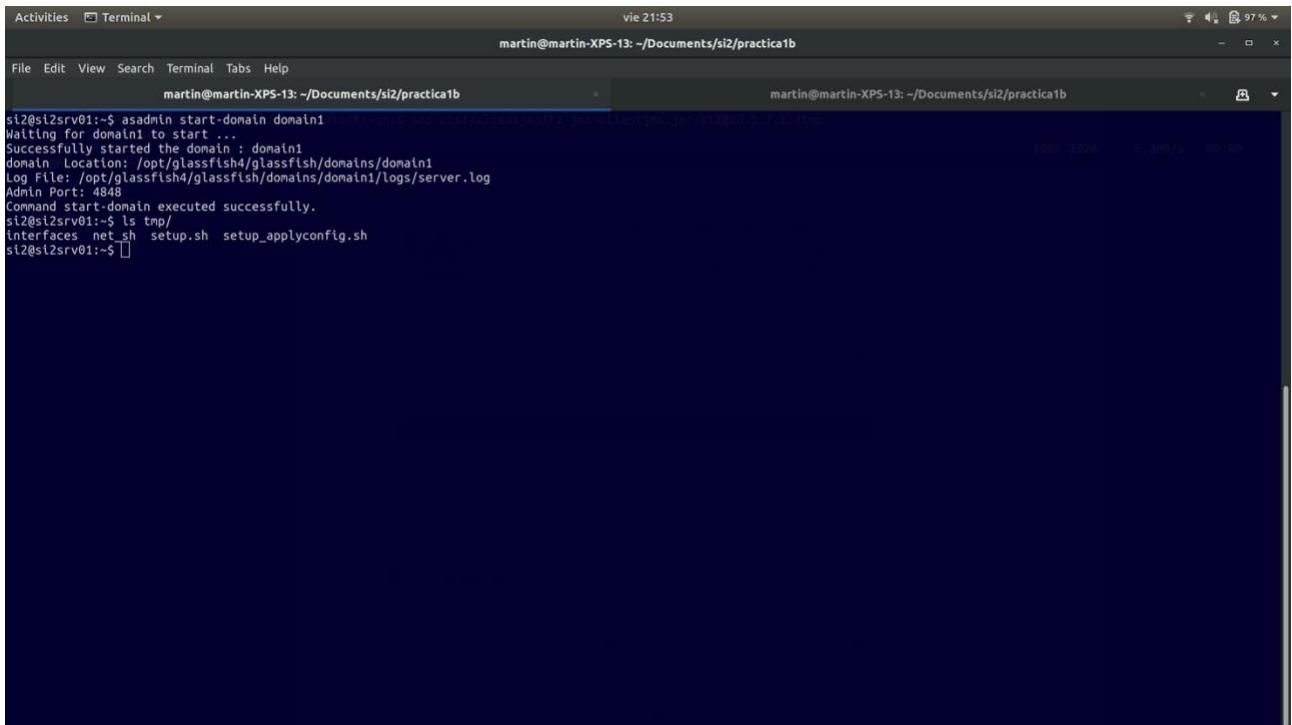
3.1. ***scp dist/clientjms/P1-jms-clientjms.jar si2@10.1.7.1:/tmp*** (desde el PC1)

Al hacer este comando no se ha copiado en la maquina virtual el archivo. Hemos probado distintas opciones como borrar el .profile de la MV o usando rsync y no lo hemos conseguido por lo que no hemos podido realizar el resto del ejercicio.

```

Activities Terminal vie 21:53
martin@martin-XPS-13: ~/Documents/si2/practica1b/P1-jms
File Edit View Search Terminal Tabs Help
martin@martin-XPS-13: ~/Documents/si2/practica1b/P1-jms$ scp dist/clientjms/P1-jms-clientjms.jar si2@10.1.7.1:/tmp
martin@martin-XPS-13: ~/Documents/si2/practica1b/P1-jms$ si2@10.1.7.1's password:
P1-jms-clientjms.jar
martin@martin-XPS-13: ~/Documents/si2/practica1b/P1-jms$ 

```



A screenshot of a Linux terminal window titled "Terminal". The window has two tabs: "martin@martin-XPS-13: ~/Documents/si2/practica1b" and "martin@martin-XPS-13: ~/Documents/si2/practica1b". The left tab contains the command "asadmin start-domain domain1" and its output, which shows the domain starting successfully. The right tab is empty.

```
Activities Terminal vie 21:53
martin@martin-XPS-13: ~/Documents/si2/practica1b
File Edit View Search Terminal Tabs Help
martin@martin-XPS-13: ~/Documents/si2/practica1b
martin@martin-XPS-13: ~/Documents/si2/practica1b
si2@si2srv01:~$ asadmin start-domain domain1
Waiting for domain1 to start ...
Successfully started the domain : domain1
domain1 Location: /opt/glassfish4/glassfish/domains/domain1
Log File: /opt/glassfish4/glassfish/domains/domain1/logs/server.log
Admin Port: 4848
Command start-domain executed successfully.
si2@si2srv01:~$ ls tmp/
Interfaces net.sh setup.sh setup_applyconfig.sh
si2@si2srv01:~$
```

Lo que nos faltaría por hacer es:

3.2. *si2@si2srv01:~\$ export JAVA_HOME=/usr/lib/jvm/java-8-oracle/* (VM1)

4. Introducimos el comando para verificar que el contenido de la cola (vacia)

CAPTURA

5. Realizamos dos peticiones de anular pago y volvemos a comprobar el contenido de la cola

CAPTURA

6. Volvemos a activar el MDB y probamos a realizar un pago y después lo cancelamos para comprobar el correcto funcionamiento de todo:

CAPTURA