		<b>Escuela Politécnica Superior</b> <b>Ingeniería Informática</b> <b>Prácticas de Sistemas Informáticos 2</b>			
<b>Grupo</b>	<b>2363</b>	<b>Práctica</b>	1A	<b>Fecha</b>	27/02/2019
<b>Alumno/a</b>		García Fernández Román			

## Práctica 1: Arquitectura de JAVA EE (Primera parte)

### Ejercicio 1

1. Ejecutar **asadmin start-domain** en la vm1
2. Desempaquetar el archivo comprimido P1-base (los archivos de los que se habla más adelante están en la carpeta donde se descomprima)
3. En el archivo build.properties en el anfitrión cambiamos:
  - a. as.host por el valor de la IP de la vm1
4. En el archivo postgres.properties en el anfitrión cambiamos:
  - a. db.host por el valor de la IP de la vm1
  - b. db.client por el valor de la IP de la vm1

**Nota: db.client no es el cliente final de la aplicación, hace referencia al cliente de la base de datos (PostgreSQL), es decir, el que va a hacer peticiones a la base de datos**

5. Creamos la base de datos en PostgreSQL en la vm1 con **createdb -U alumnodb visa**
6. Exportamos en el anfitrión la variable J2EE\_HOME:
  - a. J2EE\_HOME=/usr/local/glassfish-4.1.1/glassfish/
7. Ejecutamos en el anfitrión desde la carpeta P1-base **ant todo**

Si no funciona ejecutamos:

- a. ant compilar
- b. ant empaquetar
- c. ant desplegar

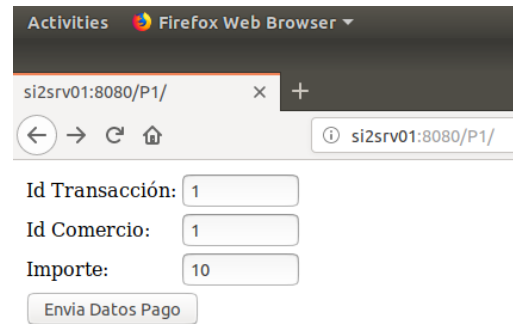
Si sigue sin funcionar comprobar la variable J2EE\_HOME: puede que tenga otra ruta como:  
/opt/glassfish4/glassfish

8. Si todo ha funcionado deberíamos poder conectarnos con Tora con los siguientes parámetros:
  - a. Dirección ip: <La IP de la vm1>
  - b. Nombre (de la bd): visa
  - c. Puerto: 5432
  - d. Usuario: alumnodb
  - e. Contraseña: <vacío>

Para ver el contenido de la bd en Tora:

- a. Clic en **tools** (en el top de la ventana)
- b. Clic en **schema browser**
- c. En la parte de la derecha, en un desplegable que selecciona el **schema** (que por defecto pone **information\_schema**) seleccionar **public**

Accedemos con un navegador a la aplicación que se esta ejecutando en la vm1 y creamos una transacción.



The screenshot shows a Firefox Web Browser window with the address bar displaying 'si2srv01:8080/P1/'. The page contains a form with the following fields:

- Id Transacción:
- Id Comercio:
- Importe:
- Envia Datos Pago (button)

En la página de pago con tarjeta introducimos los datos de una tarjeta que exista en la base de datos. De esta forma nos aseguramos de que los datos que vamos a enviar los puede encontrar.



The screenshot shows a Firefox Web Browser window with the address bar displaying 'si2srv01:8080/P1/comienzapago'. The page title is 'Pago con tarjeta'. The form contains the following fields:

- Numero de visa:
- Titular:
- Fecha Emisión:
- Fecha Caducidad:
- CVV2:
- Pagar (button)

Below the form, the following transaction details are displayed:

- Id Transacción: 1
- Id Comercio: 1
- Importe: 10.0

Prácticas de Sistemas Informáticos II

Recibimos la página de pago realizado correctamente.



The screenshot shows a Firefox Web Browser window with the address bar displaying 'si2srv01:8080/P1/procesapago'. The page title is 'Pago con tarjeta'. The page content indicates a successful payment:

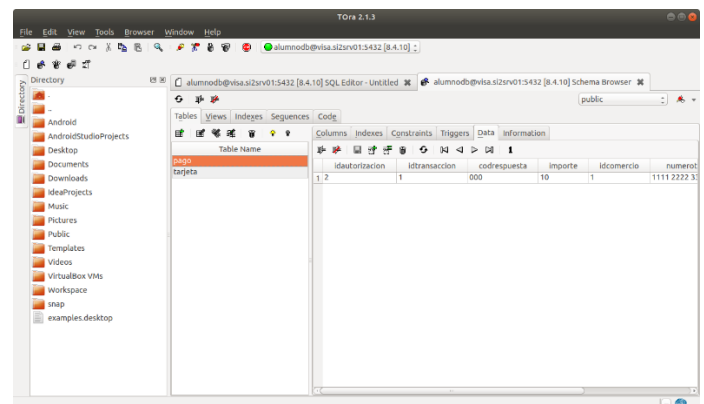
Pago realizado con éxito. A continuación se muestra el comprobante del mismo:

- idTransaccion: 1
- idComercio: 1
- importe: 10.0
- codRespuesta: 000
- idAutorizacion: 2

[Volver al comercio](#)

Prácticas de Sistemas Informáticos II

Comprobamos con Tora que el pago se ha guardado correctamente



The screenshot shows the Tora 2.1.3 interface. The 'pago' table is selected, and its structure is displayed in the 'Table Name' tab. The table has the following columns:

idautorizacion	idtransaccion	codrespuesta	importe	idcomercio	numero
1,2	1	000	10	1	1111 2222 3

Accedemos a *testdb.jsp*

Sistema de Pago con tarjeta x +

← → ↻ 🏠 ⓘ si2srv01:8080/P1/testdb.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

---

### Consulta de pagos

Id Comercio:

---

### Borrado de pagos

Id Comercio:

---

Prácticas de Sistemas Informáticos II

Comprobamos que podemos borrar pagos

Activities Firefox Web Browser ▾

Sistema de Pago con tarjeta x +

← → ↻ 🏠 ⓘ si2srv01:8080/P1/delpagos

## Pago con tarjeta

Se han borrado 1 pagos correctamente para el comercio 1

[Volver al comercio](#)

---

Prácticas de Sistemas Informáticos II

Comprobamos que podemos consultar pagos

Sistema de Pago con tarjeta x +

← → ↻ 🏠 ⓘ si2srv01:8080/P1/getpagos

## Pago con tarjeta

Lista de pagos del comercio 1

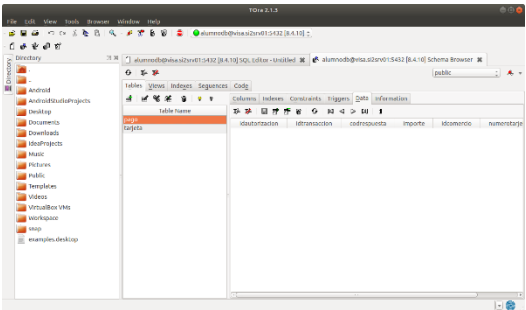
idTransaccion	Importe	codRespuesta	idAutorizacion
1	10.0	000	2

[Volver al comercio](#)

---

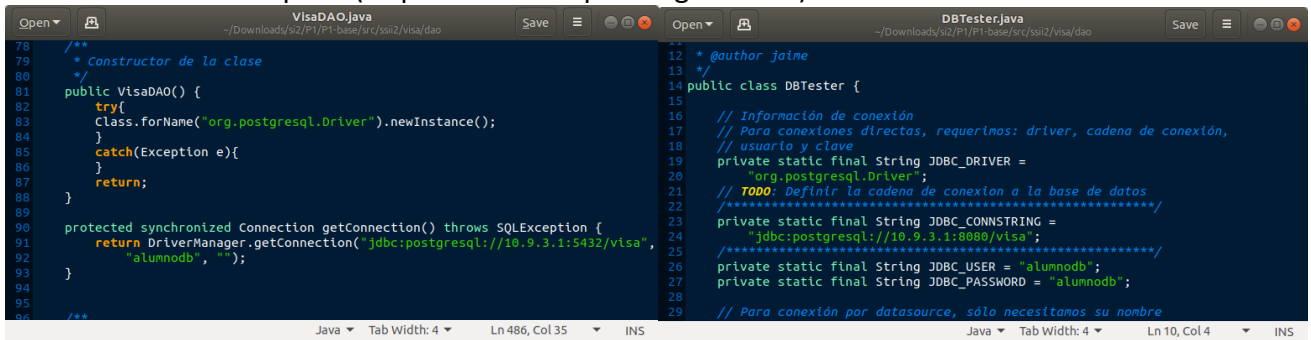
Prácticas de Sistemas Informáticos II

Comprobamos que el pago se ha borrado el  
pago de la base de datos con Tora



## Ejercicio 2

1. Hay que cambiar los archivos VisaDAO.java y DBTester.java del anfitrión con la configuración de nuestra maquina (la ip debe ser la que tenga la vm1)

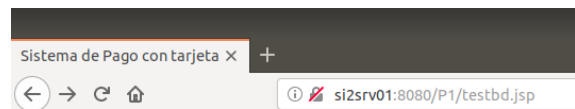


```
78 /**
79  * Constructor de la clase
80  */
81 public VisaDAO() {
82     try{
83         Class.forName("org.postgresql.Driver").newInstance();
84     }
85     catch(Exception e){
86     }
87     return;
88 }
89
90 protected synchronized Connection getConnection() throws SQLException {
91     return DriverManager.getConnection("jdbc:postgresql://10.9.3.1:5432/visa",
92     "alumnodb", "");
93 }
94
95 /**
96  */
```

```
12 * @author jaine
13 */
14 public class DBTester {
15
16     // Información de conexión
17     // Para conexiones directas, requerimos: driver, cadena de conexión,
18     // usuario y clave
19     private static final String JDBC_DRIVER =
20         "org.postgresql.Driver";
21
22     // TODO: Definir la cadena de conexión a la base de datos
23     //*****
24     private static final String JDBC_CONNSTRING =
25         "jdbc:postgresql://10.9.3.1:8080/visa";
26     //*****
27     private static final String JDBC_USER = "alumnodb";
28     private static final String JDBC_PASSWORD = "alumnodb";
29
30     // Para conexión por datasource, sólo necesitamos su nombre
```

Si no funciona

1. Reiniciar postgres ***sudo /etc/init.d/postgresql-8.4 restart***
2. Si cambias algún archivo
  - a. Ant limpiar-todo
  - b. Ant todo



## 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="10"/>
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="11/20"/>
CVV2:	<input type="text" value="123"/>
Modo debug:	<input type="radio"/> True <input type="radio"/> False
Direct Connection:	<input checked="" type="radio"/> True <input type="radio"/> False
Use Prepared:	<input type="radio"/> True <input type="radio"/> False
<input type="button" value="Pagar"/>	

### Consulta de pagos

Id Comercio:	<input type="text" value="1"/>
<input type="button" value="GetPagos"/>	

### Borrado de pagos

Id Comercio:	<input type="text" value="1"/>
<input type="button" value="DelPagos"/>	

Accedemos a testdbd.jsp y realizamos un pago con conexión directa marcada

Comprobamos la consulta



Sistema de Pago con tarjeta x +

← → ↻ 🏠 si2srv01:8080/P1/getpagos

## Pago con tarjeta

Lista de pagos del comercio 1

idTransaccion	Importe	codRespuesta	idAutorizacion
1	10.0	000	1

[Volver al comercio](#)

Prácticas de Sistemas Informáticos II

Probamos a borrar el pago



Sistema de Pago con tarjeta x +

← → ↻ 🏠 si2srv01:8080/P1/delpagos

## Pago con tarjeta

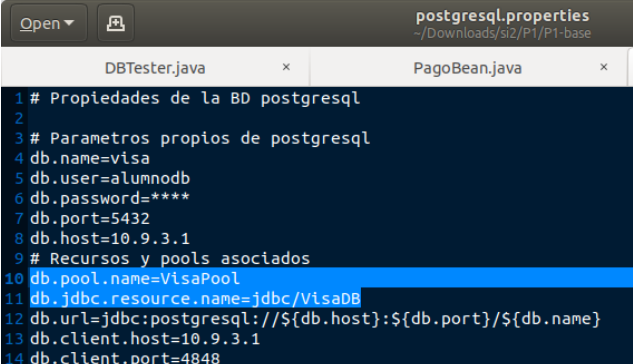
Se han borrado 1 pagos correctamente para el comercio 1

[Volver al comercio](#)

Prácticas de Sistemas Informáticos II

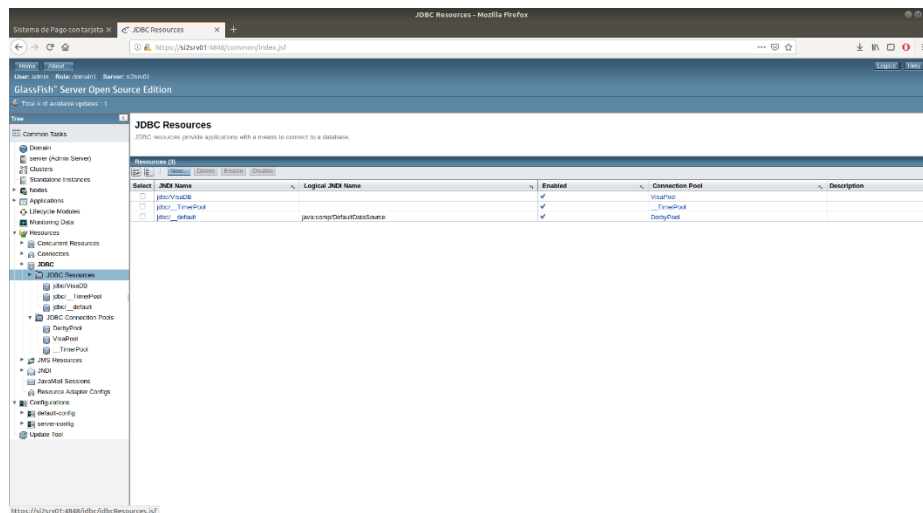
### Ejercicio 3

1. En el archivo postgresql.properties cambiar las variables  
db.pool.name=VisaPool  
db.jdbc.resource.name=jdbc/VisaDB

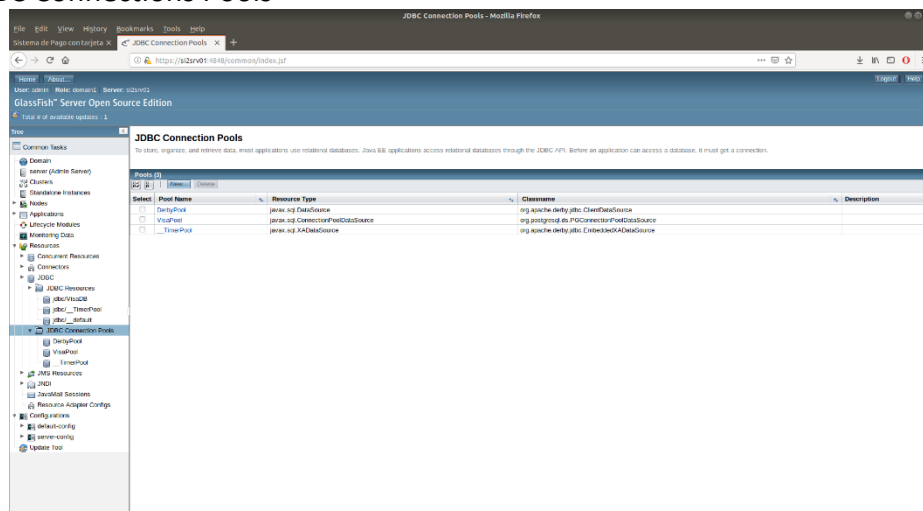


```
1 # Propiedades de la BD postgresql
2
3 # Parametros propios de postgresql
4 db.name=visa
5 db.user=alumnodb
6 db.password=****
7 db.port=5432
8 db.host=10.9.3.1
9 # Recursos y pools asociados
10 db.pool.name=VisaPool
11 db.jdbc.resource.name=jdbc/VisaDB
12 db.url=jdbc:postgresql://${db.host}:${db.port}/${db.name}
13 db.client.host=10.9.3.1
14 db.client.port=4848
```

2. Entrar en <ip>:4848
  - a. Usuario: admin
  - b. Password: adminadmin
3. Clic en JDBC resources

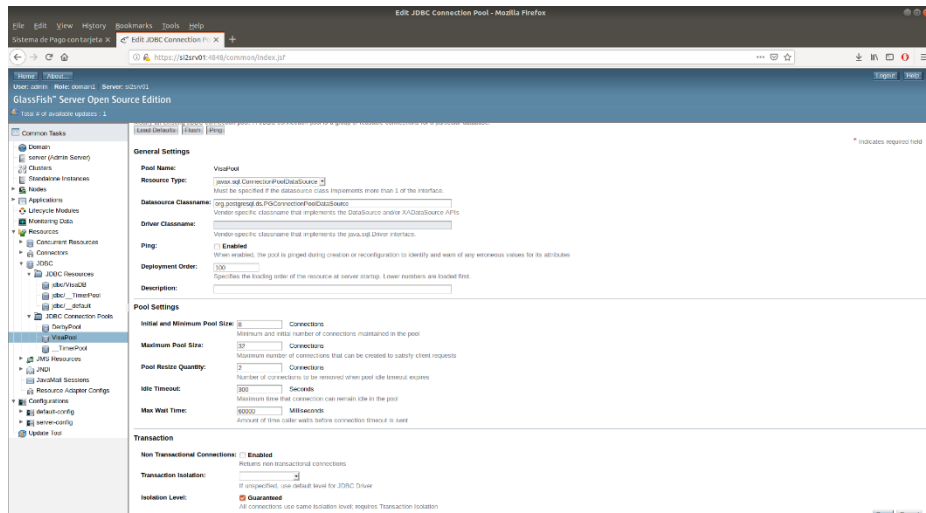


#### 4. Clic en JDBC Connections Pools

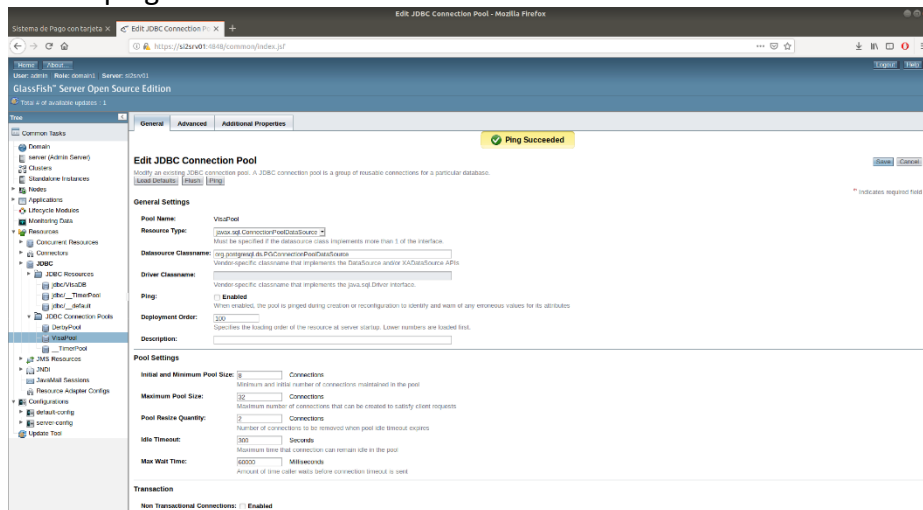


#### 5. Parámetros

- Initial minimum pool size: Tamaño mínimo inicial del pool. Si es muy alto, pero hay pocas conexiones, se desperdician recursos. Si es muy bajo y hay muchas conexiones, se pierde en rendimiento.
- Maximum pool size: Tamaño máximo del pool. Se debe ajustar con respecto a la memoria del servidor. Un numero excesivamente alto provoca caídas en el servidor. Uno muy bajo provoca grandes esperas en los clientes
- Pool resize quantity: Cantidad de conexiones, con timeout terminado, que debe haber para que se eliminen. Como el parámetro Initial minimum pool size, en exceso desperdicia recursos y en defecto sobrecargamos el servidor con procesos de creación y destrucción.
- Idle timeout: Tiempo máximo que una conexión puede permanecer asignada. Mismo impacto que el parámetro Pool resize quantity
- Max wait time: Timeout máximo para cada conexión. Tiene el mismo impacto que Pool resize quantity



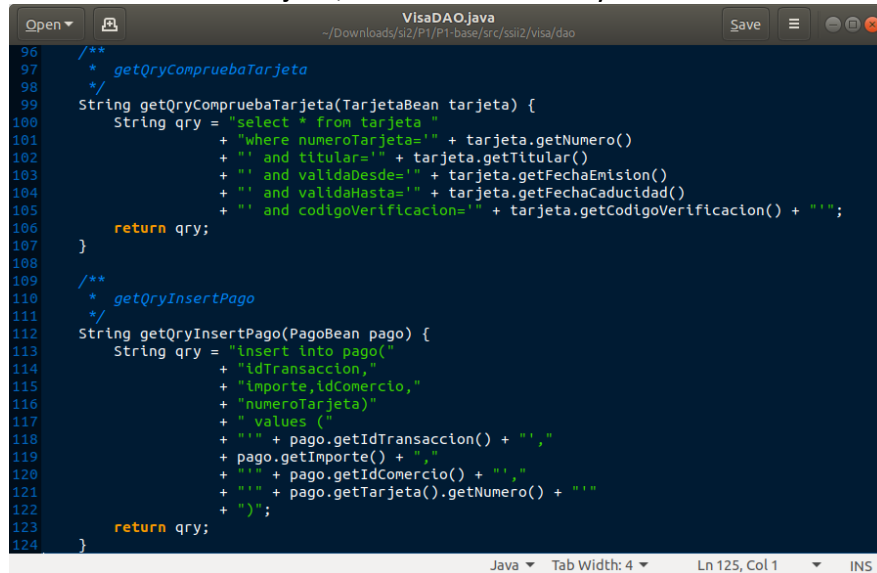
Por último, hacemos ping a la JDBC





## Ejercicio 4

Se encuentra en el archivo VisaDAO.java, entre las líneas 99 y 124

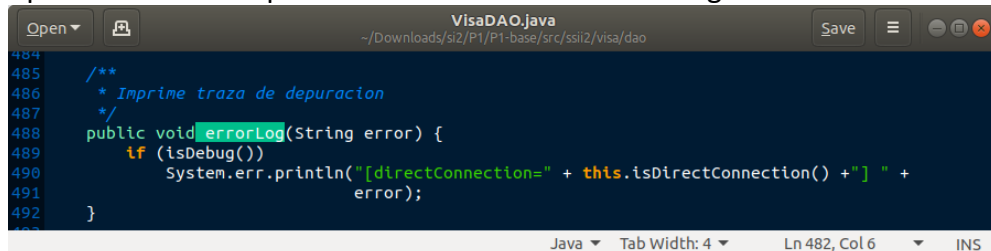


```
96  /**
97   * getQryCompruebaTarjeta
98   */
99  String getQryCompruebaTarjeta(TarjetaBean tarjeta) {
100      String qry = "select * from tarjeta"
101                  + " where numeroTarjeta=" + tarjeta.getNumero()
102                  + " and titular=" + tarjeta.getTitular()
103                  + " and validaDesde=" + tarjeta.getFechaEmision()
104                  + " and validaHasta=" + tarjeta.getFechaCaducidad()
105                  + " and codigoVerificacion=" + tarjeta.getCodigoVerificacion() + " ";
106      return qry;
107  }
108
109  /**
110   * getQryInsertPago
111   */
112  String getQryInsertPago(PagoBean pago) {
113      String qry = "insert into pago("
114                  + "idTransaccion,"
115                  + "importe,idComercio,"
116                  + "numeroTarjeta)"
117                  + " values ("
118                  + " " + pago.getIdTransaccion() + ","
119                  + " " + pago.getImporte() + ","
120                  + " " + pago.getIdComercio() + ","
121                  + " " + pago.getTarjeta().getNumero() + " "
122                  + ")";
123      return qry;
124  }
```

## Ejercicio 5

Localización del log de servidor en la vm1: /opt/glassfish4.1.2/glassfish/domains/domain1/logs/server.log

En VisaDAO podemos ver la implementación del método errorLog:



```
484
485  /**
486   * Imprime traza de depuracion
487   */
488  public void errorLog(String error) {
489      if (isDebug())
490          System.err.println("[directConnection=" + this.isDirectConnection() + "] " +
491                             error);
492  }
```

Este método es llamado en los métodos compruebaTarjeta, getPagos, realizaPago y delPagos de la clase VisaDAO.java

Accedemos a testbd.jsp y realizamos un pago con la etiqueta debug activada

Sistema de Pago con tarjeta x General Information x +

← → ↻ 🏠 [s12srv01:8080/p1/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

---

### Consulta de pagos

Id Comercio:

---

### Borrado de pagos

Id Comercio:

---

Prácticas de Sistemas Informáticos II

Comprobamos que se ha realizado correctamente

Sistema de Pago con tarjeta x General Information x +

← → ↻ 🏠 [s12srv01:8080/p1/procesapago](#)

## Pago con tarjeta

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

idTransaccion: 1

idComercio: 1

importe: 10.0

codRespuesta: 000

idAutorizacion: 2

[Volver al comercio](#)

---

Prácticas de Sistemas Informáticos II

Miramos los logs para ver que se escribe en el fichero lo que hemos comprobado

Log Viewer - Mozilla Firefox

<https://s12srv01:4848/common/logviewer/logViewer.js?instanceName=server&loglevel=INFO&viewResults=true>

Modify Search

Instance:

Log File:

Log Viewer Results (40)

Records before 411 | Log File Record Numbers 411 through 450 | Records after 450 | [↑](#)

Record Number	Log Level	Message	Logger	Timestamp	Name-Value Pairs
450	SEVERE	[directConnection=false] select idAutorizacion, codRespuesta from pago where idTransaccion = '1 ... (details)		Feb 24, 2019 09:56:11.731	(levelValue=1000, timeMills=1551030971731)
449	SEVERE	[directConnection=false] insert into pago(idTransaccion, importe, idComercio, numeroTarjeta) values ('1...', (details)		Feb 24, 2019 09:56:11.713	(levelValue=1000, timeMills=1551030971713)
448	SEVERE	[directConnection=false] select * from tarjeta where numeroTarjeta='1111 2222 3333 4444' and titular... (details)		Feb 24, 2019 09:56:11.655	(levelValue=1000, timeMills=1551030971655)
447	INFO	WebModule[nul] ServletContext.log() [INFO] Acceso correcto/procesapago (details)	javaw.enterprise.web	Feb 24, 2019 09:56:11.577	(levelValue=800, timeMills=1551030971577)
446	INFO	WebModule[nul] ServletContext.log() [INFO] Acceso correcto/testbd.jsp (details)	javaw.enterprise.web	Feb 24, 2019 09:56:32.132	(levelValue=800, timeMills=1551030932132)

## Ejercicio 6

Después de crear la estructura de directorios y copiar los archivos como indica el enunciado, asegurarse de cambiar el nombre de la clase (y su archivo .java) de VisaDAO a VisaDAOWS

1. Asegurarse de que VisaDAOWS.java contiene los siguientes imports:

```
ssii2.visa.*;

java.sql.Connection;

java.sql.PreparedStatement;

java.sql.ResultSet;

java.sql.SQLException;

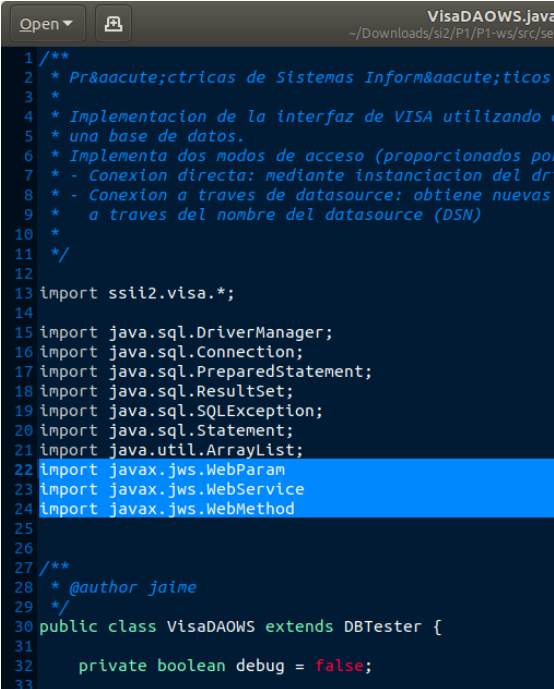
java.sql.Statement;

java.util.ArrayList;

javax.jws.WebMethod;

javax.jws.WebParam;

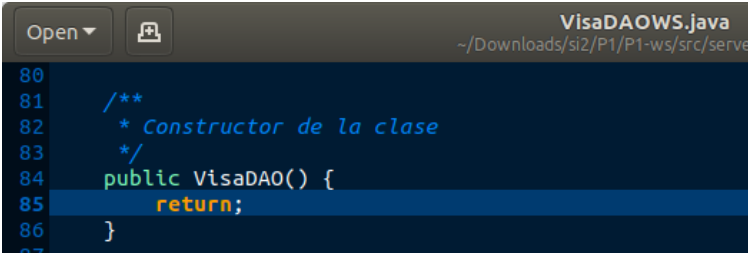
javax.jws.WebService;
```



```
Open [icon] VisaDAOWS.java
~/Downloads/si2/P1/P1-ws/src/se

1 /**
2  * Prácticas de Sistemas Informáticos
3  *
4  * Implementación de la interfaz de VISA utilizando
5  * una base de datos.
6  * Implementa dos modos de acceso (proporcionados por
7  * - Conexion directa: mediante instanciación del dr
8  * - Conexion a través de datasource: obtiene nuevas
9  * a través del nombre del datasource (DSN)
10 *
11 */
12
13 import ssii2.visa.*;
14
15 import java.sql.DriverManager;
16 import java.sql.Connection;
17 import java.sql.PreparedStatement;
18 import java.sql.ResultSet;
19 import java.sql.SQLException;
20 import java.sql.Statement;
21 import java.util.ArrayList;
22 import javax.jws.WebParam;
23 import javax.jws.WebService;
24 import javax.jws.WebMethod;
25
26
27 /**
28  * @author jaine
29  */
30 public class VisaDAOWS extends DBTester {
31
32     private boolean debug = false;
33 }
```

2. Vaciar el constructor de VisaDAOWS



```
Open [icon] VisaDAOWS.java
~/Downloads/si2/P1/P1-ws/src/serve

80
81 /**
82  * Constructor de la clase
83  */
84 public VisaDAO() {
85     return;
86 }
87
```

3. Poner la etiqueta @WebService() en la clase VisaDAOWS

```

26
27 /**
28  * @author jaime
29  */
30 @WebService()
31 public class VisaDAOWS extends DBTester {
32
33     private boolean debug = false;
34
35     /**
36      * TODO: Declara un atributo booleano "a

```

4. Poner la etiqueta @WebMethod(operationName = "<nombre del metodo>") en los métodos compruebaTarjeta, realizaPago, getPagos, delPagos, isPrepared, setPrepared, isDebug y setDebug, como en la siguiente imagen de ejemplo

```

208 }
209
210 /**
211  * Realiza el pago
212  * @param pago
213  * @return
214  */
215 @WebMethod(operationName = "realizaPago")
216 public synchronized boolean realizaPago(PagoBean pago) {
217     Connection con = null;
218     Statement stmt = null;
219     ResultSet rs = null;
220     boolean ret = false;

```

5. En los demás métodos indicaremos exclusivamente que no son métodos del servicio de esta manera: @WebMethod(exclude=true)  
También se pueden comentar, si no hacen falta para el servicio
6. Es necesario modificar el tipo de variable que devuelve realizaPago (antes un booleano, ahora un objeto entero), puesto que de otra manera el Web Service no podría acceder a los datos.

```

321
322     if(ret == true){
323         return pago;
324     } else{
325         return null;
326     }
327     return ret;
328 }
329 }
330
331 /**
332  *
333  * Buscar los pagos asociados a un comercio
334  * @param idComercio
335  * @return
336  */
337 public PagoBean[] getPagos(String idComercio) {
338     "

```

7. No debemos modificar los métodos de la base de datos.
8. De los métodos setDebug debemos excluir uno y declarar el otro como WebMethod.

```

474 }
475
476 /**
477  * @param debug the debug to set
478  */
479 @WebMethod(operationName = "setDebug")
480 public void setDebug(boolean debug) {
481     this.debug = debug;
482 }
483
484 /**
485  * @param debug the debug to set
486  */
487 @WebMethod(exclude=true)
488 public void setDebug(String debug) {
489     this.debug = (debug.equals("true"));
490 }
491
492

```

9. También debemos agregar los métodos isDirectConnection y setDirectConnection de la clase padre BDTester.java.

```

461     return ret;
462 }
463
464 @WebMethod(operationName = "isDirectConnection")
465 public boolean isDirectConnection() {
466     return super.isDirectConnection();
467 }
468
469 @WebMethod(operationName = "setDirectConnection")
470 public void setDirectConnection(@WebParam(name = "directConnection")boolean
directConnection) {
471     super.setDirectConnection(directConnection);
472 }
473
474 /**
475  * TODO: Metodos isPrepared() y setPrepared()
476  */
477 /**
478  * @WebMethod(operationName = "isPrepared")
479  * public boolean isPrepared() {
480  *     return prepared;
481  * }
482  * @WebMethod(operationName = "setPrepared")
483  * public void setPrepared(@WebParam(name = "prepared")boolean prepared) {
484  *     this.prepared = prepared;
485  * }
486  */

```

10. Además de los cambios en el código, debemos modificar la ip de db.client.host a la ip de la vm2 en el fichero postgresql.properties
11. Respuesta a la pregunta
  - a. Porque si no el Web Service no podría acceder a los datos

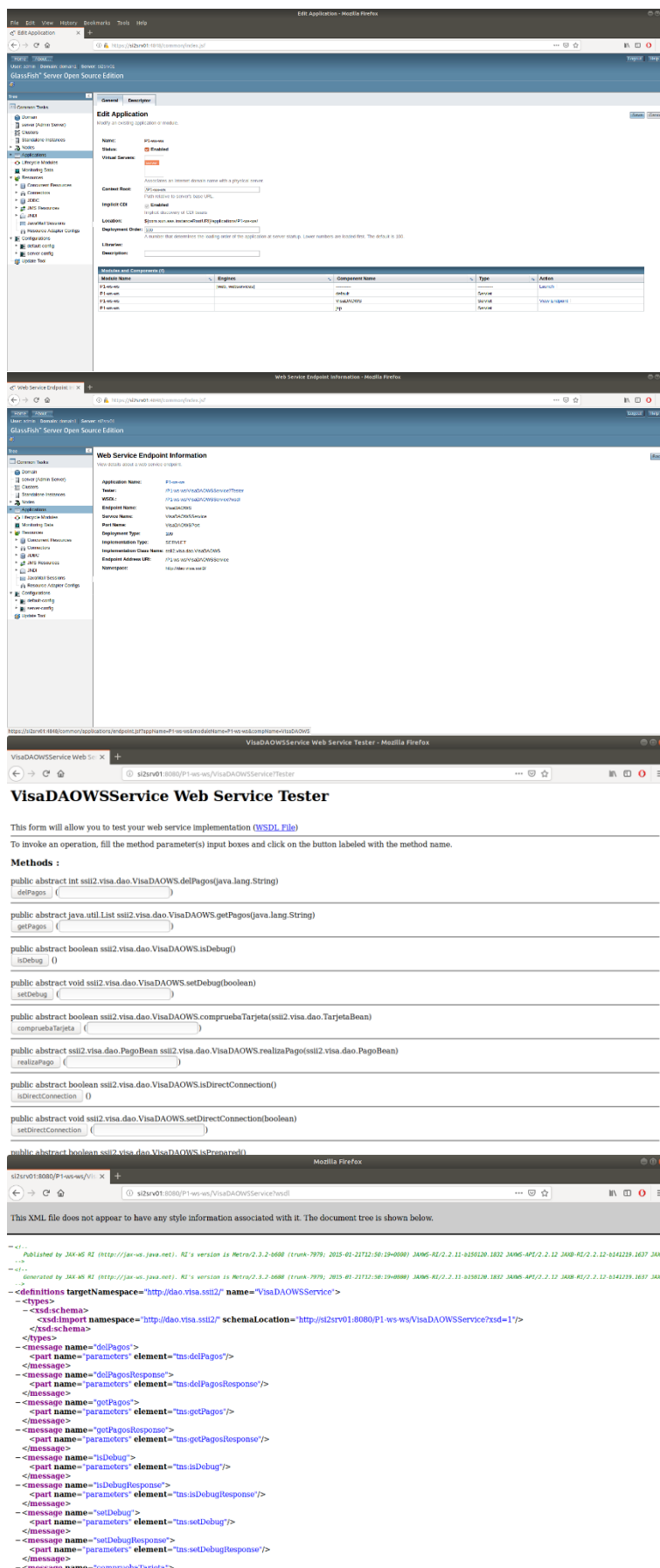
### Ejercicio 7

Accedemos a la página de administración de glassfish, al apartado de aplicaciones

Seleccionamos nuestra aplicación  
y vemos la información correspon-  
diente al endpoint

Desde el enlace a *view endpoint* accedemos al tester de los métodos

Desde el enlace a *view endpoint* accedemos al xml que contiene la configuración relativa a los métodos



- **¿En qué fichero están definidos los tipos de datos intercambiados con el webservice?**

En el fichero XML al que se accede desde Glassfish, se encuentra una URL del import donde se encuentran definidos los tipos de datos del servidor

- **¿Qué tipos de datos predefinidos se usan?**  
Boolean, int, double y string
- **¿Cuáles son los tipos de datos que se definen?**

```
<xs:element name="compruebaTarjeta" type="tns:compruebaTarjeta"/>
<xs:element name="compruebaTarjetaResponse" type="tns:compruebaTarjetaResponse"/>
<xs:element name="delPagos" type="tns:delPagos"/>
<xs:element name="delPagosResponse" type="tns:delPagosResponse"/>
<xs:element name="getPagos" type="tns:getPagos"/>
<xs:element name="getPagosResponse" type="tns:getPagosResponse"/>
<xs:element name="isDebug" type="tns:isDebug"/>
<xs:element name="isDebugResponse" type="tns:isDebugResponse"/>
<xs:element name="isDirectConnection" type="tns:isDirectConnection"/>
<xs:element name="isDirectConnectionResponse" type="tns:isDirectConnectionResponse"/>
<xs:element name="isPrepared" type="tns:isPrepared"/>
<xs:element name="isPreparedResponse" type="tns:isPreparedResponse"/>
<xs:element name="realizaPago" type="tns:realizaPago"/>
<xs:element name="realizaPagoResponse" type="tns:realizaPagoResponse"/>
<xs:element name="setDebug" type="tns:setDebug"/>
<xs:element name="setDebugResponse" type="tns:setDebugResponse"/>
<xs:element name="setDirectConnection" type="tns:setDirectConnection"/>
<xs:element name="setDirectConnectionResponse" type="tns:setDirectConnectionResponse"/>
<xs:element name="setPrepared" type="tns:setPrepared"/>
<xs:element name="setPreparedResponse" type="tns:setPreparedResponse"/>
```

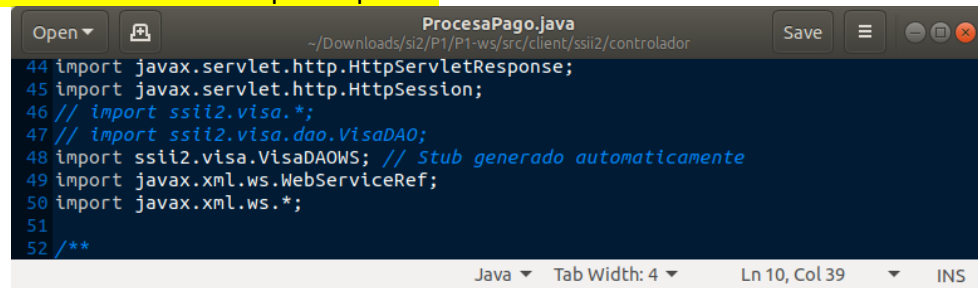
- **¿Qué etiqueta está asociada a los métodos invocados en el webservice?**  
xs:complexType
- **¿Qué etiqueta describe los mensajes intercambiados en la invocación de los métodos del webservice?**  
xs:sequence
- **¿En qué etiqueta se especifica el protocolo de comunicación con el webservice?**  
sesion-config
- **¿En qué etiqueta se especifica la URL a la que se deberá conectar un cliente para acceder al webservice?**  
schemalocation

## Ejercicio 8

1. Modificar los imports del archivo ProcesaPago.java de la carpeta cliente

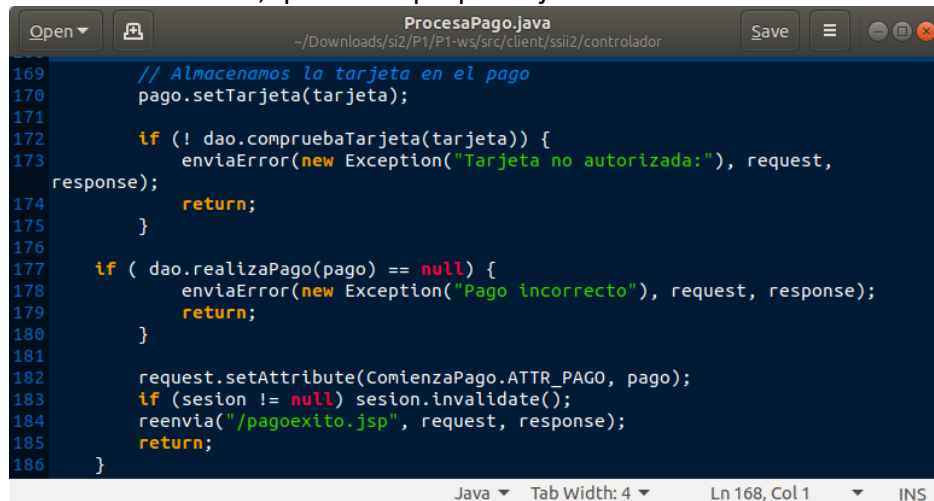
```
//import ssii2.visa.dao.VisaDAO;
//import ssii2.visa.VisaDAOWSService
import ssii2.visa.VisaDAOWS; // Stub generado automáticamente
import javax.xml.ws.WebServiceRef;
```

`import javax.xml.ws.*;` -> IMPORTANTE EL \* si se hace con lo que viene en el enunciado faltarán clases por importar



```
44 import javax.servlet.http.HttpServletResponse;
45 import javax.servlet.http.HttpSession;
46 // import ssii2.visa.*;
47 // import ssii2.visa.dao.VisaDAO;
48 import ssii2.visa.VisaDAOWS; // Stub generado automaticamente
49 import javax.xml.ws.WebServiceRef;
50 import javax.xml.ws.*;
51
52 /**
```

2. Modificar las llamadas a `compruebaTarjeta` y `realizaPago` para que mantengan la consistencia con la implementación actual, que usa el propio objeto como retorno en vez de un boolean



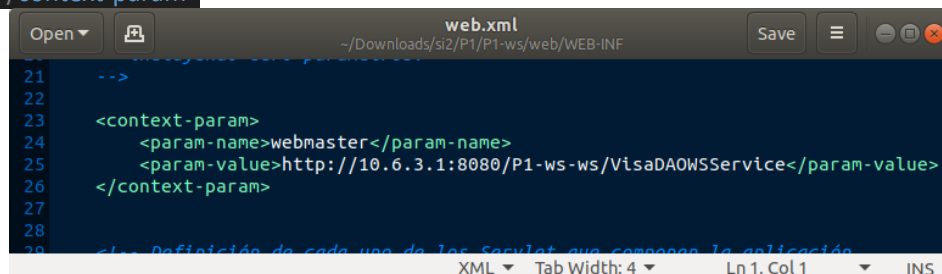
```
169 // Almacenamos la tarjeta en el pago
170 pago.setTarjeta(tarjeta);
171
172 if (! dao.compruebaTarjeta(tarjeta)) {
173     enviaError(new Exception("Tarjeta no autorizada:"), request,
response);
174     return;
175 }
176
177 if ( dao.realizaPago(pago) == null) {
178     enviaError(new Exception("Pago incorrecto"), request, response);
179     return;
180 }
181
182 request.setAttribute(ComienzaPago.ATTR_PAGO, pago);
183 if (sesion != null) sesion.invalidate();
184 reenvia("/pagoexitoso.jsp", request, response);
185 return;
186 }
```

## Ejercicio 9

1. Cambiar el archivo `web/WEB-INF/web.xml` para que contenga información de contexto para conectarse a otro servlet

Descomentar las líneas de `context-param` y rellenarlas con la dirección del servicio en la vm1:

```
<context-param>
<param-name>webmaster</param-name>
<param-value>http://10.6.3.1:8080/P1-ws-ws/VisaDAOWSService</param-value>
</context-param>
```



```
21 -->
22
23 <context-param>
24     <param-name>webmaster</param-name>
25     <param-value>http://10.6.3.1:8080/P1-ws-ws/VisaDAOWSService</param-value>
26 </context-param>
27
28
29 <!-- Definición de cada uno de los Servlet que componen la aplicación -->
```

2. Modificar `ProcesaPago.java` para que utilice el contexto del servlet

```
// VisaDAO dao = new VisaDAO();
VisaDAOWSService service = new VisaDAOWSService();
VisaDAOWS dao = service.getVisaDAOWSPort();
BindingProvider bp = (BindingProvider) dao;
bp.getRequestContext().put(BindingProvider.ENDPOINT_ADDRESS_PROPERTY,
getServletContext().getInitParameter("webmaster"));
```



```

153
154 // VisaDAO dao = new VisaDAO();
155 VisaDAOWSService service = new VisaDAOWSService();
156 VisaDAOWSPort dao = service.getVisaDAOWSPort();
157 BindingProvider bp = (BindingProvider) dao;
158 bp.getRequestContext().put(BindingProvider.ENDPOINT_ADDRESS_PROPERTY,
159     getServletContext().getInitParameter("webmaster"));
160
161 HttpSession session = request.getSession(false);

```

## Ejercicio 10

1. Poner la etiqueta `@WebMethod(operationName = "delPagos")` en el metodo `delPagos` de `VisaDAOWS.java` (servidor)

```

411 // Borrar los pagos asociados a un comercio
412 /**
413  *
414  * @param idComercio
415  * @return numero de registros afectados
416  */
417 @WebMethod(operationName = "delPagos")
418 public int delPagos(@WebParam(name = "idComercio") String idComercio) {
419

```

2. Cambiar `DelPagos.java` para que tenga los imports que tiene `ProcesaPago.java` (ejercicio 8)

```

12 package ssii2.controlador;
13
14
15 import java.io.IOException;
16 import javax.servlet.ServletException;
17 import javax.servlet.http.HttpServletRequest;
18 import javax.servlet.http.HttpServletResponse;
19 import ssii2.visa.PagoBean;
20 import ssii2.visa.dao.VisaDAO;
21 import ssii2.visa.VisaDAOWS;
22 import ssii2.visa.VisaDAOWSService;
23 import javax.xml.ws.WebServiceRef;
24 import javax.xml.ws.*;
25 import java.util.*;
26
27 /**
28  *
29  * @author phaya
30  */
31 public class GetPagos extends ServletRaiz {

```

3. Cambiar `DelPagos.java` para que use el contexto igual que `ProcesaPago.java` (ejercicio 9)

```

52 * @param request objeto de petici&oacute;n
53 * @param response objeto de respuesta
54 */
55 protected void processRequest(HttpServletRequest request, HttpServletResponse
response)
56     throws ServletException, IOException {
57
58     // VisaDAO dao = new VisaDAO();
59     VisaDAOWSService daoserv = new VisaDAOWSService();
60     VisaDAOWSPort dao = daoserv.getVisaDAOWSPort();
61     BindingProvider bp = (BindingProvider) dao;
62     bp.getRequestContext().put(BindingProvider.ENDPOINT_ADDRESS_PROPERTY,
63         getServletContext().getInitParameter("webmaster"));
64
65     /* Se recoge de la petici&oacute;n el par&aacute;metro idComercio*/
66     String idComercio = request.getParameter(PARAM_ID_COMERCIO);
67
68     /* Petici&oacute;n de los pagos para el comercio */

```

4. Hacer lo mismo en `GetPagos.java`
5. También hace falta modificar el tipo de dato que devuelve `getPagos` en `VisaDAOWS.java` a un `arraylist` de `PagoBean`:

`@WebMethod(operationName = "getPagos")`

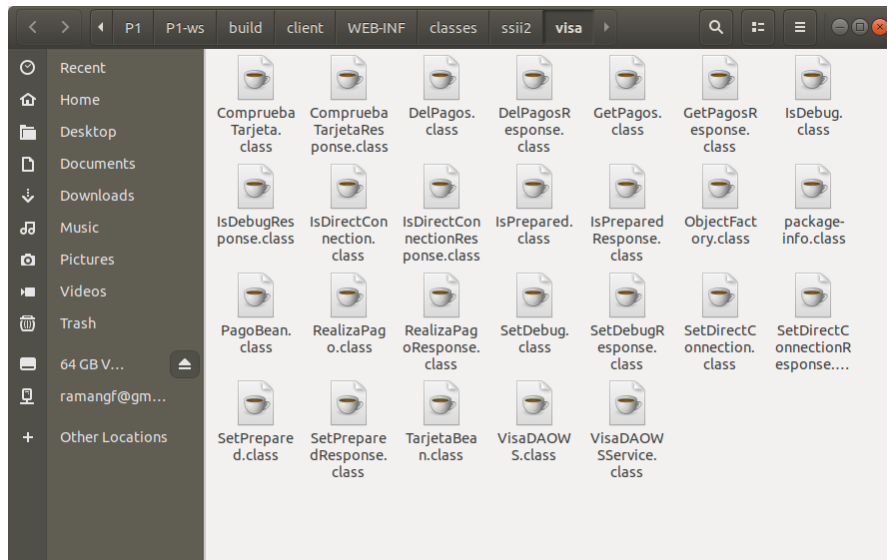
```
public ArrayList<PagoBean> getPagos(@WebParam(name = "idComercio")String
idComercio) {
```

6. En GetPagos.java Hay que reconvertir el ArrayList<PagoBean> en PagoBean[] con el método toArray()

```
ArrayList<PagoBean> arraylistPagos = new ArrayList<PagoBean>(pagoslst.size());
arraylistPagos.addAll(pagoslst);
PagoBean[] pagos = new PagoBean[arraylistPagos.size()];
pagos = arraylistPagos.toArray(pagos);
```

## Ejercicio 11

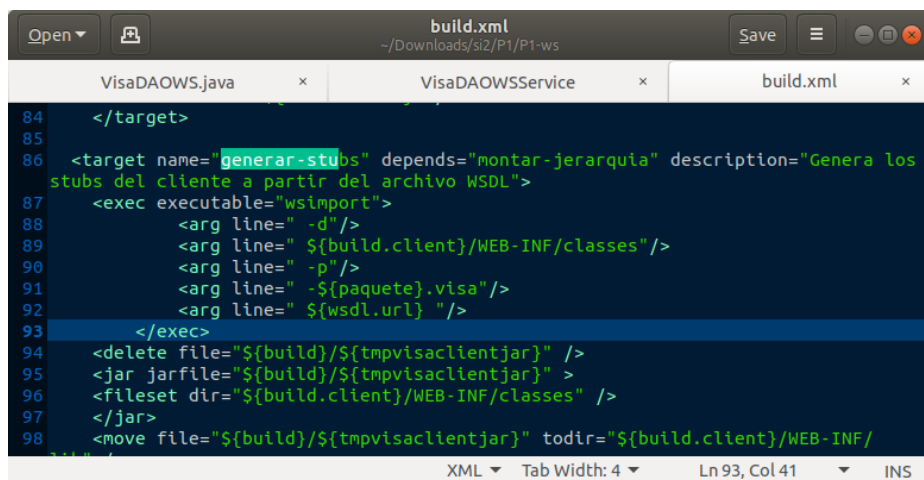
1. Ejecutar la siguiente línea  
wsimport -d build/client/WEB-INF/classes/ -p ssii2.visa <http://10.9.3.1:8080/P1-ws-ws/Visa-DAOWSservice?wsdl>
2. Comprobar la ruta (...)/P1-ws/build/client/WEB-INF/classes/ssii2/visa y verificar que se han creado todas las clases



## Ejercicio 12

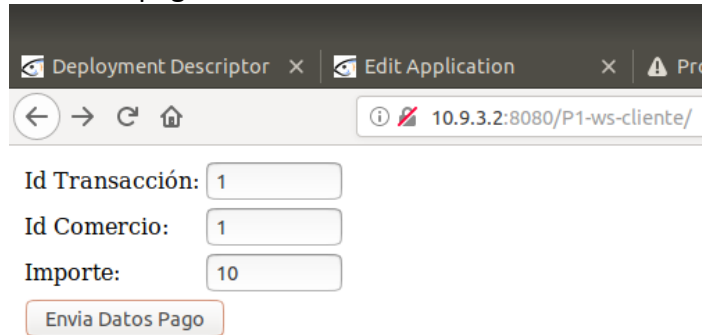
En build.xml buscar la entrada <target name="generar-stubs" (...) y meter las líneas de código para poder usar wsimport:

```
<target name="generar-stubs" depends="montar-jerarquia" description="Genera los
stubs del cliente a partir del archivo WSDL">
<exec executable="wsimport">
<arg line=" -d"/>
<arg line=" ${build.client}/WEB-INF/classes"/>
<arg line=" -p"/>
<arg line=" -${paquete}.visa"/>
<arg line=" ${wsdl.url} "/>
</exec>
```



## Ejercicio 13

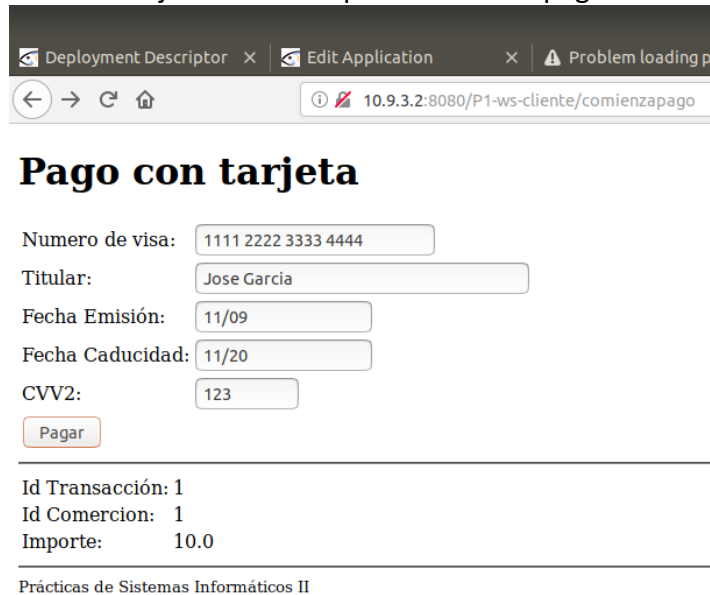
Creamos una transacción de pago



The screenshot shows a web browser window with two tabs: 'Deployment Descriptor' and 'Edit Application'. The address bar displays '10.9.3.2:8080/P1-ws-cliente/'. The form contains the following fields and buttons:

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

Utilizamos los datos de una tarjeta existente para realizar el pago



The screenshot shows a web browser window with two tabs: 'Deployment Descriptor' and 'Edit Application'. The address bar displays '10.9.3.2:8080/P1-ws-cliente/comienzapago'. The page title is 'Pago con tarjeta'. The form contains the following fields and buttons:

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

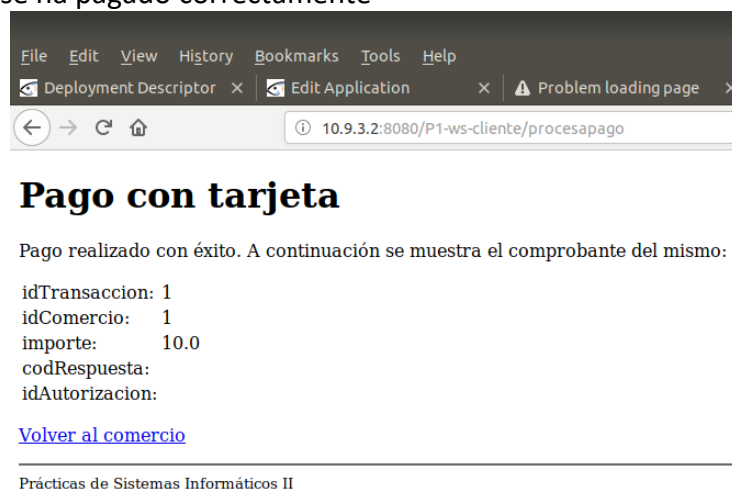
---

Id Transacción: 1  
Id Comercion: 1  
Importe: 10.0

---

Prácticas de Sistemas Informáticos II

Observamos que se ha pagado correctamente



The screenshot shows a web browser window with three tabs: 'Deployment Descriptor', 'Edit Application', and 'Problem loading page'. The address bar displays '10.9.3.2:8080/P1-ws-cliente/procesapago'. The page title is 'Pago con tarjeta'. The text on the page reads:

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

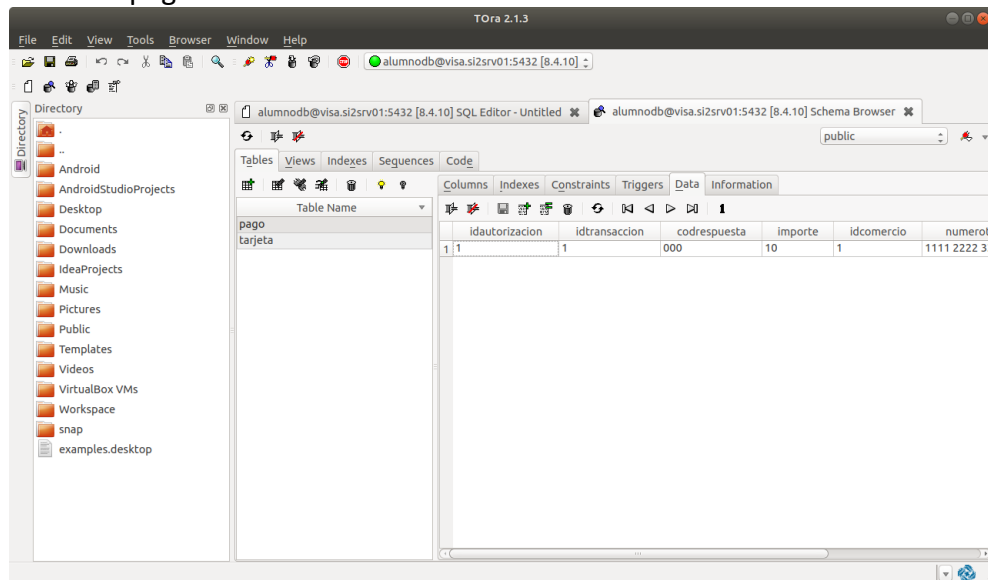
idTransaccion: 1  
idComercio: 1  
importe: 10.0  
codRespuesta:  
idAutorizacion:

[Volver al comercio](#)

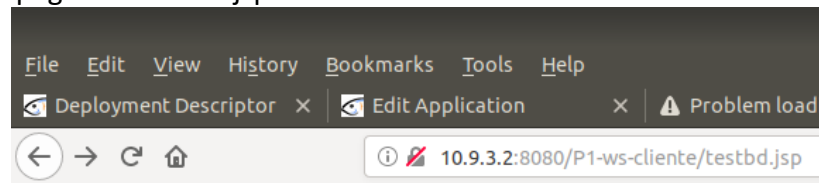
---

Prácticas de Sistemas Informáticos II

## Comprobamos el pago en la base de datos



Probamos otro pago con testdb.jsp



## Pago con tarjeta

### Proceso de un pago

Id Transacción:	<input type="text" value="2"/>
Id Comercio:	<input type="text" value="2"/>
Importe:	<input type="text" value="101"/>
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="11/20"/>
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"/>	

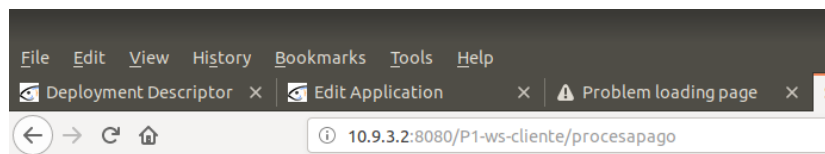
### Consulta de pagos

Id Comercio:	<input type="text" value="1"/>
<input type="button" value="GetPagos"/>	

### Borrado de pagos

Id Comercio:	<input type="text" value="1"/>
<input type="button" value="DelPagos"/>	

Prácticas de Sistemas Informáticos II



## Pago con tarjeta

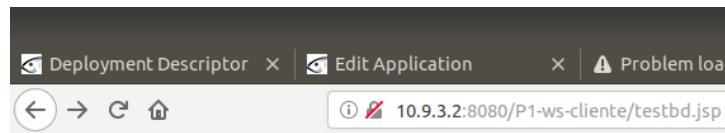
Pago realizado con éxito. A continuación se muestra el comprobante del mismo:

idTransaccion: 2  
idComercio: 2  
importe: 101.0  
codRespuesta:  
idAutorizacion:

[Volver al comercio](#)

Prácticas de Sistemas Informáticos II

Probamos a consultar la transacción



## 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

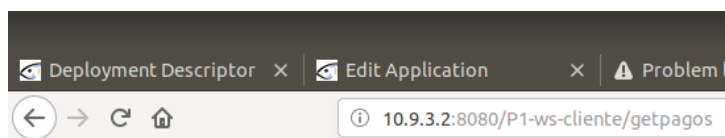
### Consulta de pagos

Id Comercio:

### Borrado de pagos

Id Comercio:

Prácticas de Sistemas Informáticos II



## Pago con tarjeta

Lista de pagos del comercio 2

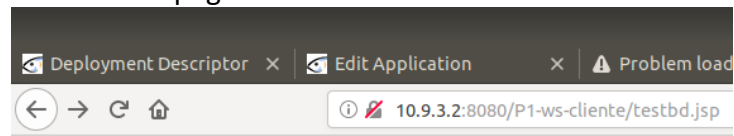
idTransaccion	Importe	codRespuesta	idAutorizacion
2	101.0	000	2

[Volver al comercio](#)

Prácticas de Sistemas Informáticos II

**Nota: se ha borrado conscientemente la transacción anterior, pero no tenemos capturas del proceso**

Probamos a borrar la el último pago realizado



## 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

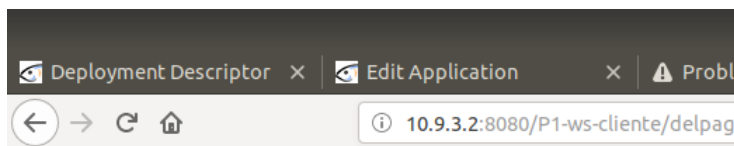
### Consulta de pagos

Id Comercio:

### Borrado de pagos

Id Comercio:

Prácticas de Sistemas Informáticos II



## Pago con tarjeta

Se han borrado 1 pagos correctamente para el comercio 2

[Volver al comercio](#)

Prácticas de Sistemas Informáticos II



### ***Cuestión número 1:***

**Teniendo en cuenta el diagrama de la Figura 3, indicar las páginas html, jsp y servlets por los que se pasa para realizar un pago desde pago.html, pero en el caso de uso en que se introduce una tarjeta cuya fecha de caducidad ha expirado**

Cuando hay un error como fecha de caducidad incorrecta, pasa por los siguientes archivos: ComienzaPago, formadatosvisa, ProcesaPago, VisaDAO, ProcesaPago y muestraerror

### ***Cuestión número 2:***

**De los diferentes servlets que se usan en la aplicación, ¿podría indicar cuáles son los encargados de solicitar la información sobre el pago con tarjeta cuando se usa pago.html para realizar el pago?**

El servlet encargado es ProcesaPago

### ***Cuestión número 3:***

**Cuando se accede a pago.html para hacer el pago, ¿qué información solicita cada servlet? Respecto a la información que manejan, ¿cómo la comparten? ¿dónde se almacena?**

ComienzaPago recibe las peticiones de comienzo de pago. Solicita id de transacción, id de comercio, importe y ruta de retorno. Crea un objeto PagoBean que usa para almacenar estos valores y devolverlo para que puedan acceder otros servlets.

### ***Cuestión número 4:***

**Enumere las diferencias que existen en la invocación de servlets, a la hora de realizar el pago, cuando se utiliza la página de pruebas extendida testbd.jsp frente a cuando se usa pago.html. ¿Podría indicar por qué funciona correctamente el pago cuando se usa testbd.jsp a pesar de las diferencias observadas?**

Cuando se hace por medio de pago.html se ejecuta el servlet ComienzaPago mientras que al hacerlo por medio del testbd.jsp no utiliza el servlet ComienzaPago sino que le pasa los campos directamente a ProcesaPago.

Esto funciona porque ProcesaPago tiene la capacidad de realizar la misma funcionalidad que ComienzaPago creando una instancia de PagoBean y rellenándola con los valores necesarios.