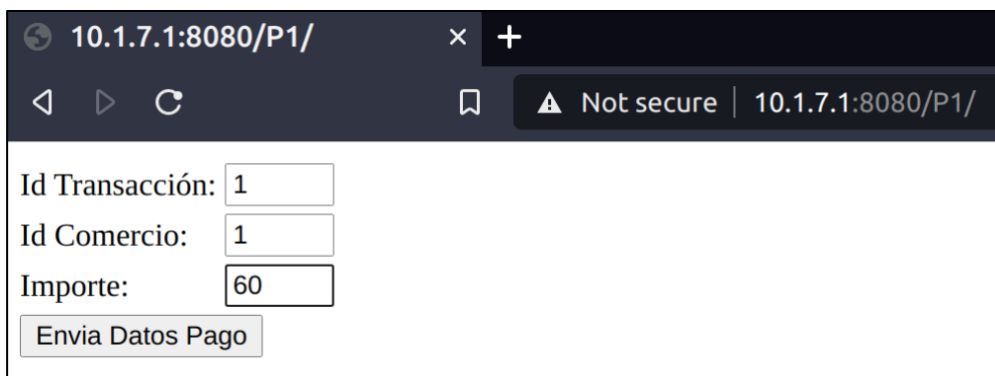
		<b>Escuela Politécnica Superior</b> <b>Ingeniería Informática</b> <b>Prácticas de Sistemas Informáticos 2</b>			
<b>Grupo</b>	<b>2401</b>	<b>Práctica</b>	1a	<b>Fecha</b>	05/03/2021
<b>Alumno/a</b>		De las Heras Moreno, Martín			
<b>Alumno/a</b>		Valderrábano Zamorano, Santiago Manuel			

## Práctica 1a: Arquitectura de JAVA EE

### Cuestión número 1:

Para este primer ejercicio, tras descargar los ficheros de Moodle, hemos configurado el fichero *build.properties* configurando *as.host=10.1.7.1*. A continuación, iniciamos la máquina virtual, configurada como descrita en el enunciado, e iniciamos el dominio *domain1*. Después, hemos desplegado haciendo uso de “*ant todo*” la aplicación web y hemos accedido a ella a través del navegador con la ruta: <http://10.1.7.1:8080/P1>



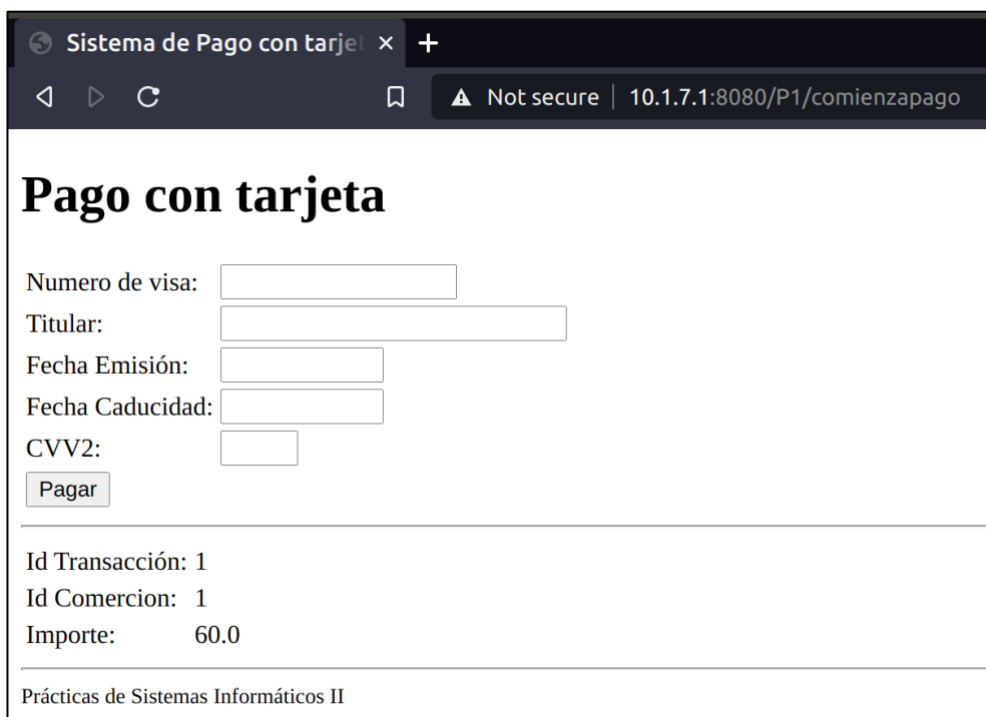
10.1.7.1:8080/P1/

Not secure | 10.1.7.1:8080/P1/

Id Transacción:

Id Comercio:

Importe:



Sistema de Pago con tarjeta

Not secure | 10.1.7.1:8080/P1/comienzapago

### Pago con tarjeta

Numero de visa:

Titular:

Fecha Emisión:

Fecha Caducidad:

CVV2:

---

Id Transacción: 1

Id Comercion: 1

Importe: 60.0

---

Prácticas de Sistemas Informáticos II

Sistema de Pago con tarjeta x +

◀ ▶ ↻ 🔖 ⚠ Not secure | 10.1.7.1:8080/P1/procesapago

# 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

Comprobamos en Tora que se ha añadido el pago correctamente a la BD.

alumnodb@visa.10.1.7.1:5432 [8.4.10] SQL Editor - Untitled x alumnodb@visa.10.1.7.1:5432 [8.4.10] Schema Browser x

public

Tables Views Indexes Sequences Code

Table Name

Columns Indexes Constraints Triggers Data Information

Table Name	idautorizacion	idtransaccion	codrespuesta	importe	idcomercio	numerotarjeta
pago	1	1	000	60	1	1111 2222 3333 4444
tarjeta						

Una vez dentro del servicio web, hemos probado a realizar un pago, y borrarlo, haciendo uso de la página de pruebas extendidas que se encuentra en <http://10.1.7.1:8080/P1/testbd.jsp>

## Borrado de pagos

Id Comercio:

---

Prácticas de Sistemas Informáticos II

Sistema de Pago con tarjeta x +

◀ ▶ ↻ 🔖 ⚠ Not secure | 10.1.7.1:8080/P1/delpagos

# Pago con tarjeta

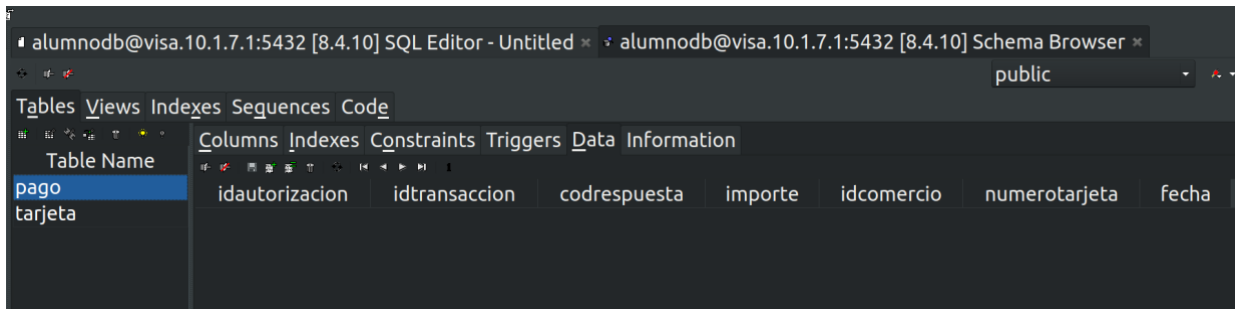
Se han borrado 1 pagos correctamente para el comercio 1

[Volver al comercio](#)

---

Prácticas de Sistemas Informáticos II

Para terminar, comprobamos en Tora que se ha borrado el pago correctamente.



## Cuestión número 2:

En esta cuestión, es necesario modificar varias variables del fichero *DBTester.java*, para que la conexión directa sea correcta.

Estas variables concretamente son:

- **JDBC\_DRIVER:** `org.apache.derby.jdbc.ClientDriver` → `org.postgresql.Driver`
- **JDBC\_CONSSTRING:** `jdbc:derby://10.1.1.1:1527/visa;create=true` → `jdbc:postgresql://10.1.7.1:5432/visa`
- **JDBC\_USER:** `APP` → `admin`
- **JDBC\_PASSWORD:** `APP` → `""`

A continuación, comprobamos que funcione todo correctamente realizando un pago con **Direct Connection** en *True*.



Sistema de Pago con tarjeta x +

Not secure | 10.1.7.1:8080/P1/procesapago

# Pago con tarjeta

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

idTransaccion: 2

idComercio: 1

importe: 60.0

codRespuesta: 000

idAutorizacion: 1

[Volver al comercio](#)

Prácticas de Sistemas Informáticos II

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

public

Tables Views Indexes Sequences Code

Table Name	Columns	Indexes	Constraints	Triggers	Data	Information	
pago	idautorizacion	idtransaccion	codrespuesta	importe	idcomercio	numerotarjeta	fecha
tarjeta	1 1	2	000	60	1	1111 2222 3333 ...	05/03/21

## Consulta de pagos

Id Comercio:

GetPagos

Sistema de Pago con tarjeta × +

◀ ▶ ↻

🔖

⚠ Not secure | 10.1.7.1:8080/P1/getpagos

# Pago con tarjeta

Lista de pagos del comercio 1

idTransaccion	Importe	codRespuesta	idAutorizacion
2	60.0	000	1

[Volver al comercio](#)

---

Prácticas de Sistemas Informáticos II

Para terminar, comprobamos que se borra bien:

## Borrado de pagos

Id Comercio:

---

Prácticas de Sistemas Informáticos II

Sistema de Pago con tarjeta × +

◀ ▶ ↻

🔖

⚠ Not secure | 10.1.7.1:8080/P1/delpagos

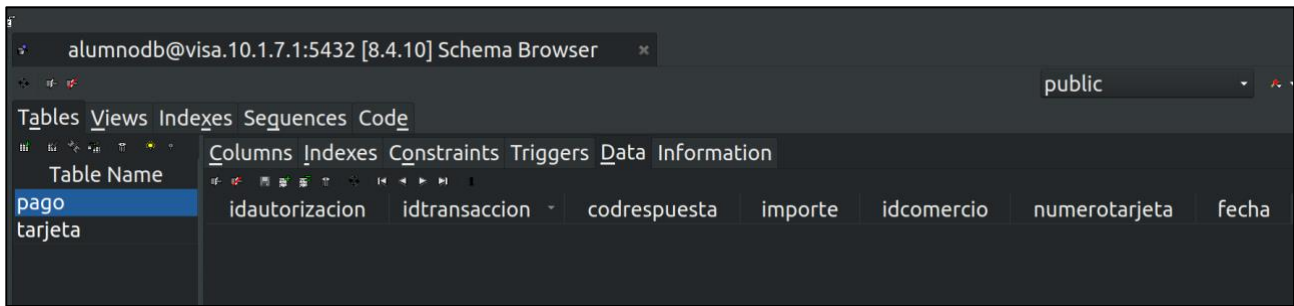
# Pago con tarjeta

Se han borrado 1 pagos correctamente para el comercio 1

[Volver al comercio](#)

---

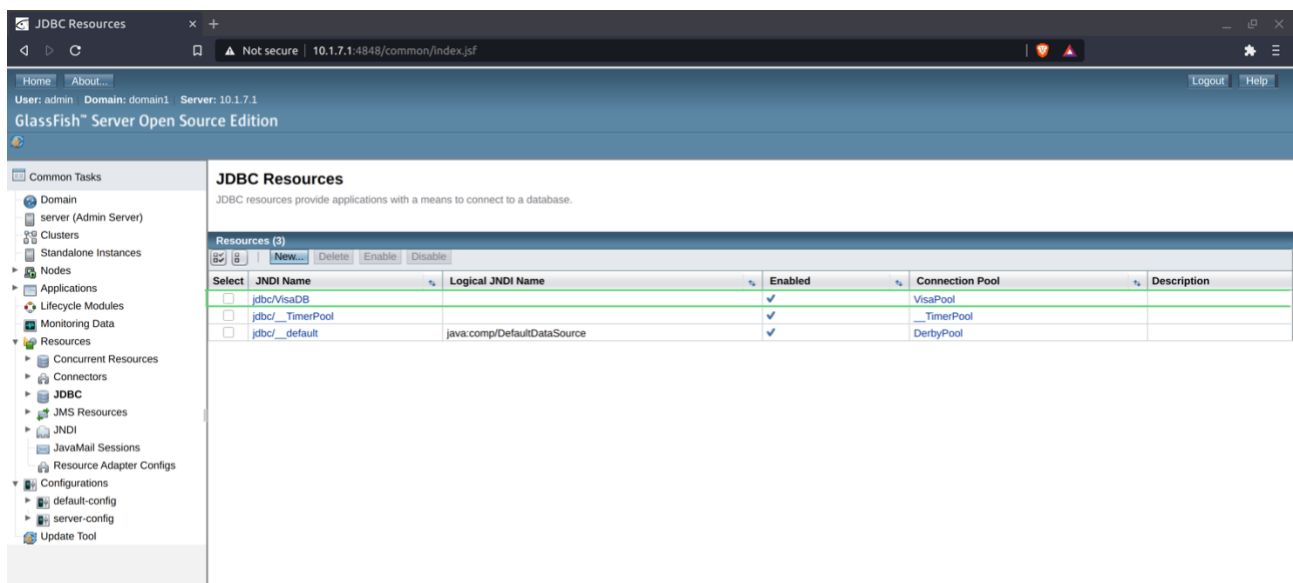
Prácticas de Sistemas Informáticos II



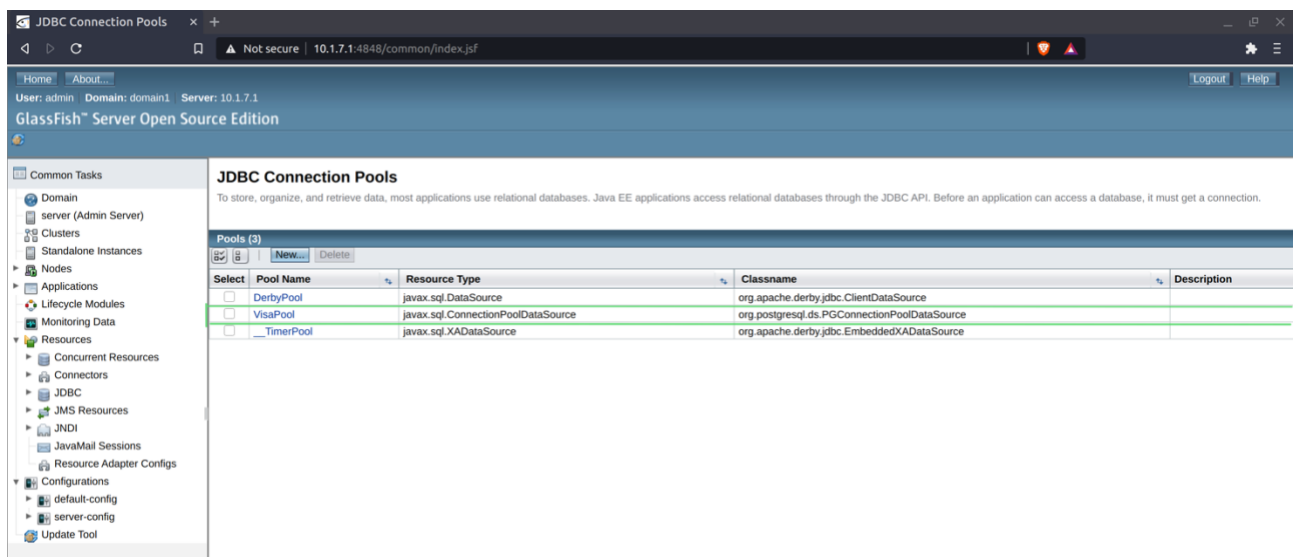
### Cuestión número 3:

Se observa del fichero postgresql.properties que el recurso es jdbc/VisaDB:  
*db.jdbc.resource.name=jdbc/VisaDB*

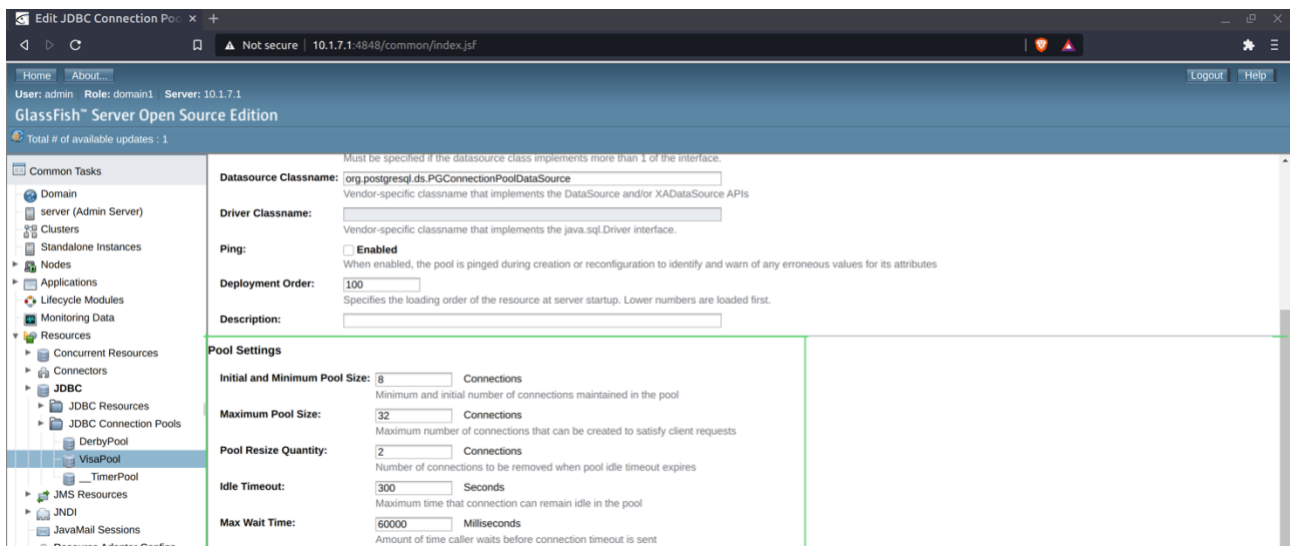
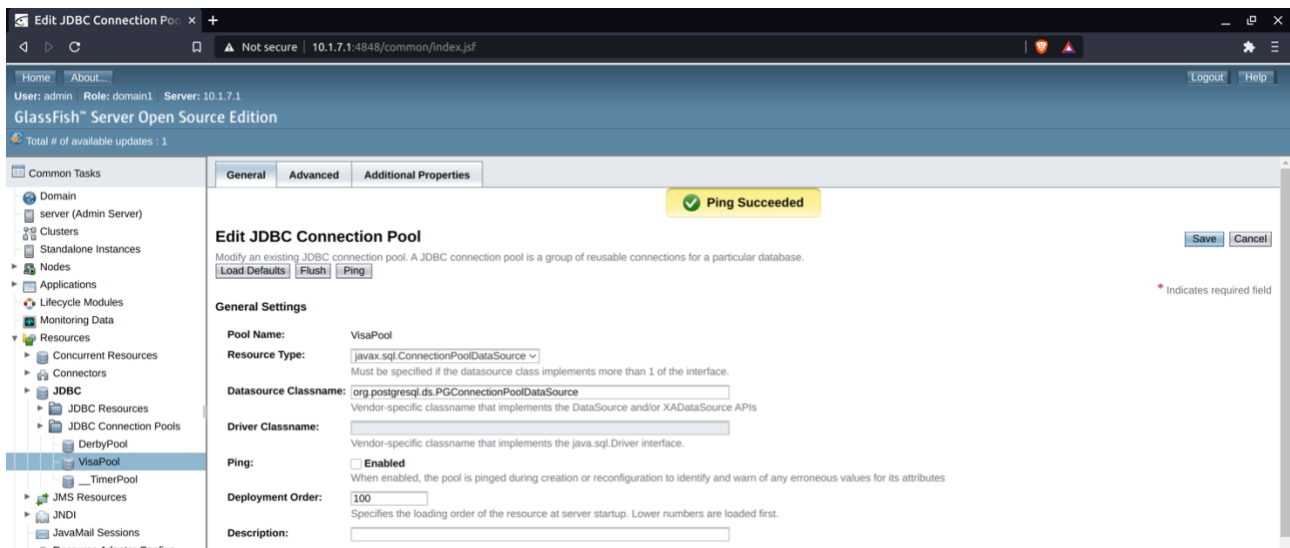
A continuación, comprobamos los recursos JDBC existentes accediendo a <http://10.1.7.1:4848/>



Después, accedemos al listado de pools para así acceder después al de VisaDB (VisaPool)



Hacemos el ping y miramos los valores de los parámetros descritos en el enunciado:



### Cuestión número 4:

Consulta de si una tarjeta es válida:

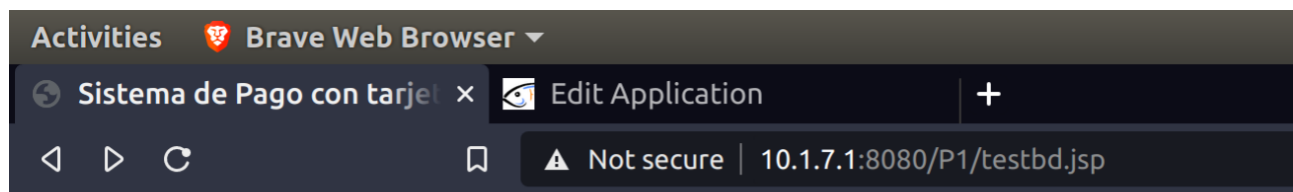
```
"select * from tarjeta "  
+ "where numeroTarjeta=" + tarjeta.getNumero()  
+ " and titular=" + tarjeta.getTitular()  
+ " and validaDesde=" + tarjeta.getFechaEmision()  
+ " and validaHasta=" + tarjeta.getFechaCaducidad()  
+ " and codigoVerificacion=" + tarjeta.getCodigoVerificacion() + ""
```

Ejecución del pago:

```
"insert into pago("
+ "idTransaccion,"
+ "importe,idComercio,"
+ "numeroTarjeta)"
+ "values ("
+ "'' + pago.getIdTransaccion() + ',' +
+ pago.getImporte() + ',' +
+ "'' + pago.getIdComercio() + ',' +
+ "'' + pago.getTarjeta().getNumero() + ''"
+ ")"
```

### Cuestión número 5:

Para este ejercicio, realizamos un pago con el modo **debug** activado, como se muestra en la captura, y a continuación comprobamos el log del servidor para ver que se han realizado correctamente las acciones.

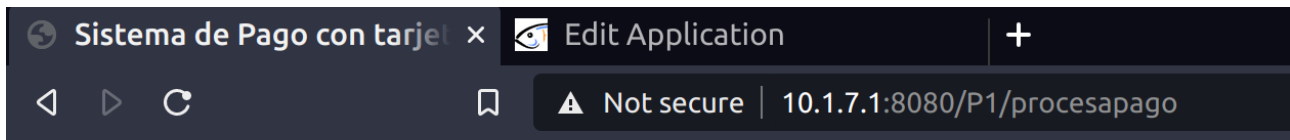


## Pago con tarjeta

### Proceso de un pago

Id Transacción:	<input type="text" value="3"/>
Id Comercio:	<input type="text" value="1"/>
Importe:	<input type="text" value="60"/>
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 checked="" type="radio"/> True <input type="radio"/> False
Direct Connection:	<input type="radio"/> True <input checked="" type="radio"/> False
Use Prepared:	<input type="radio"/> True <input type="radio"/> False
<input type="button" value="Pagar"/>	





# Pago con tarjeta

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

idTransaccion: 3  
idComercio: 1  
importe: 60.0  
codRespuesta: 000  
idAutorizacion: 5

[Volver al comercio](#)

Prácticas de Sistemas Informáticos II

## Log:

Record Number	Log Level	Message	Logger	Timestamp	Name-Value Pairs
642	SEVERE	The SSL certificate has expired: ([ Version: V3 Subject: CN=Entrust.net Secure Server Certifica... (details)	javax.enterprise.system.security.ssl	Mar 5, 2021 04:55:09.988	(levelValue=1000, timeMillis=1614948909988)
643	INFO	Listening to REST requests at context: /management/domain (details)	javax.enterprise.admin.rest	Mar 5, 2021 04:55:12.356	(levelValue=800, timeMillis=1614948912356)
644	INFO	Redirecting to /index.jsf(details)	org.glassfish.admingui	Mar 5, 2021 04:55:12.378	(levelValue=800, timeMillis=1614948912378)
645	INFO	Admin Console: Initializing Session Attributes... (details)	org.glassfish.admingui	Mar 5, 2021 04:55:12.536	(levelValue=800, timeMillis=1614948912536)
646	INFO	WebModule[null] ServletContext.log() [INFO] Acceso correcto /testbd.jsp(details)	javax.enterprise.web	Mar 5, 2021 05:05:08.418	(levelValue=800, timeMillis=1614949508418)
647	INFO	WebModule[null] ServletContext.log() [INFO] Acceso correcto /procesapago(details)	javax.enterprise.web	Mar 5, 2021 05:05:47.143	(levelValue=800, timeMillis=1614949547143)
648	SEVERE	[directConnection=true] select * from tarjeta where numeroTarjeta="1111 2222 3333 4444" and titular=... (details)		Mar 5, 2021 05:05:47.199	(levelValue=1000, timeMillis=1614949547199)
649	SEVERE	[directConnection=true] insert into pago(idTransaccion,importe,idComercio,numeroTarjeta) values (3,... (details)		Mar 5, 2021 05:05:47.223	(levelValue=1000, timeMillis=1614949547223)
650	SEVERE	[directConnection=true] select idAutorizacion, codRespuesta from pago where idTransaccion = 3 ... (details)		Mar 5, 2021 05:05:47.233	(levelValue=1000, timeMillis=1614949547233)

## Cuestión número 6:

En este ejercicio hemos modificado el fichero VisaDAOWS.java añadiendo las importaciones de las librerías para el *WebService*:

```
import javax.jws.WebMethod;  
import javax.jws.WebParam;  
import javax.jws.WebService;
```

Además, se han añadido las anotaciones dadas:

*@WebService()*: indica que la clase Java implementa un servicio web.  
*@WebMethod(operationName = "nombreMetodo")*: indica que el método Java que le sigue será exportado como un método público del servicio.  
*@WebParam(name = "nombreArgumento")*: indica que el argumento que viene a continuación también es un argumento del método equivalente del servicio.

A los métodos dictados en el enunciado.

Por último, se ha modificado la función *realizaPago()* de tal forma que devuelva el pago realizado en vez de un boolean. Esto sirve para tanto publicar las funciones de VisaDAO con servicio web como para tener una

mayor modularización que nos permita utilizar el método realiza pago en otra aplicación completamente distinta a la nuestra.

## Cuestión número 7:

Para la realización de este ejercicio tenemos que modificar el fichero *build.properties* cambiando los parámetros:

- *as.host.client* = **10.1.7.2**
- *as.host.server* = **10.1.7.1**

La definición de los tipos de datos intercambiados con el **webservice** se encuentran en:  
<http://10.1.7.1:8080/P1-ws-ws/VisaDAOWSService?wsdl>

El fichero obtenido se encuentra en el directorio P1-ws bajo el nombre *VisaDAOWSService.xml*

Los datos predefinidos que se usan son int, double, y boolean.

Los tipos de datos que se definen son pagoBean y tarjetaBean

La etiqueta asociada a los métodos invocados es *<operation>*

La etiqueta asociada a los mensajes intercambiados es *<message>*

La etiqueta en la que se especifica el protocolo de comunicación es *<soap:binding>*

La etiqueta en la que se especifica la URL es *<soap:address>*

## Cuestión número 8:

En este ejercicio incluimos en todos los ficheros que accedan al servicio de forma remota las librerías para generación de *stubs*:

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

Además, cambiamos la declaración del dao a:

```
// Nueva declaracion VisaDAOWS
VisaDAOWSService service = new VisaDAOWSService();
VisaDAOWS dao = service.getVisaDAOWSPort();
```

Así, con esta declaración, se hace la instanciación de la clase remota.

Por último, modificamos la llamada a la función procesaPago ya que ahora devuelve el objeto pago:

```
// realizaPago ahora devuelve un objeto no un boolean
pagoObj = dao.realizaPago(pago);
if (pagoObj == null) {
    enviaError(new Exception("Pago incorrecto"), request, response);
    return;
}

request.setAttribute(ComienzaPago.ATTR_PAGO, pagoObj);
if (sesion != null) sesion.invalidate();
reenvia("/pagoexito.jsp", request, response);
return;
```

### Cuestión número 9:

En primer lugar añado al fichero web.xml el parámetro de la ruta al servicio:

```
<!-- RUTA SERVICIO PARA HACER LA LLAMADA DESDE LE FICHERO DE CONFIGURACION -->
<context-param>
  <param-name>rutaServicio</param-name>
  <param-value>http://10.1.7.1:8080/P1-ws-ws/VisaDAOWSService</param-value>
</context-param>
```

A continuación, añado a todas las clases que hacen uso del servicio, las mencionadas en el ejercicio anterior, añadimos el código de obtención de ruta:

```
// Obtener ruta del fichero de configuracion
```

```
String rutaServicio = getServletContext().getInitParameter("rutaServicio");
```

```
BindingProvider bp = (BindingProvider) dao;
```

```
bp.getRequestContext().put(BindingProvider.ENDPOINT_ADDRESS_PROPERTY, rutaServicio);
```

### Cuestión número 10:

Para este ejercicio, realizamos las mismas modificaciones que en el ejercicio 8 y 9 en los ficheros DelPagos.java y GetPagos.java

Por otro lado, modificamos la función getPagos para que devuelva un ArrayList en lugar de una Array como se especifica en el enunciado.

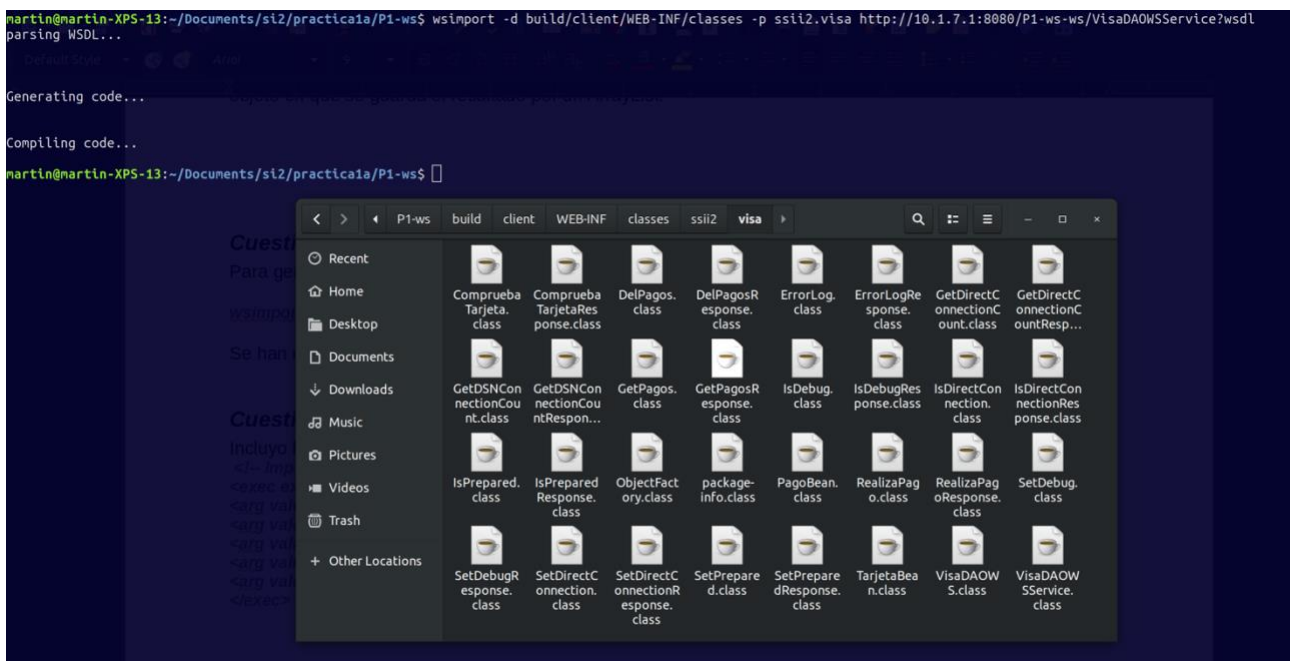
Por último, hay que modificar las llamadas a esta función, cambiando el tipo del objeto en que se guarda el resultado por un ArrayList.

### Cuestión número 11:

Para generar los stubs se ejecutamos el siguiente comando:

```
wsimport -d build/client/WEB-INF/classes -p ssl2.visa http://10.1.7.1:8080/P1-ws-ws/VisaDAOWSService?wsdl
```

Se han obtenido las siguientes clases:



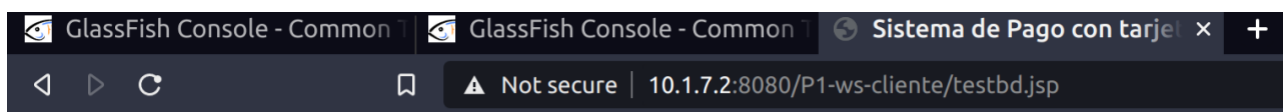
### Cuestión número 12:

Incluyo la generación de stubs en el build.xml de la siguiente forma:

```
<!-- Implementacion de wsimport en generar-stubs -->
<exec executable="wsimport">
  <arg value="-d" />
  <arg value="${build.client}/WEB-INF/classes" />
  <arg value="-p" />
  <arg value="${paquete}.visa" />
  <arg value="${wsdl.url}" />
</exec>
```

### Cuestión número 13:

Tras realizar todo lo anterior, compilamos, empaquetamos y desplegamos el servicio. A continuación, generamos los stubs, compilamos, empaquetamos y desplegamos el cliente. Finalmente, realizamos un pago, consultamos la lista de pagos y borramos el pago para observar que todo esta correctamente programado:



## 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="100"/>
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 checked="" 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"/>	

## Pago con tarjeta

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

idTransaccion: 1  
 idComercio: 1  
 importe: 100.0  
 codRespuesta: 000  
 idAutorizacion: 1

[Volver al comercio](#)

Prácticas de Sistemas Informáticos II

alumnodb@visa.10.1.7.1:5432 [8.4.10] SQL Editor - Untitled x | alumnodb@visa.10.1.7.1:5432 [8.4.10] Schema Browser x

public

Table Name	Columns	Indexes	Constraints	Triggers	Data	Information
pago	idautorizacion	idtransaccion	codrespuesta	importe	idcomercio	numerotarjeta
tarjeta	1	1	000	100	1	1111 2222 3333 ...

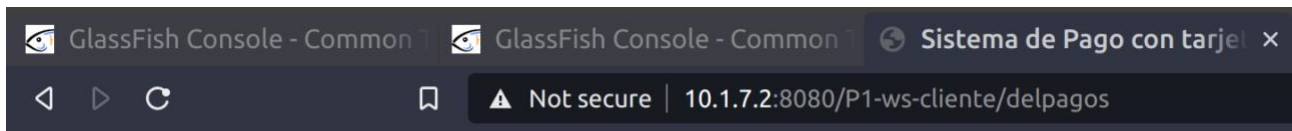
## Pago con tarjeta

Lista de pagos del comercio 1

idTransaccion	Importe	codRespuesta	idAutorizacion
1	100.0	000	1

[Volver al comercio](#)

Prácticas de Sistemas Informáticos II

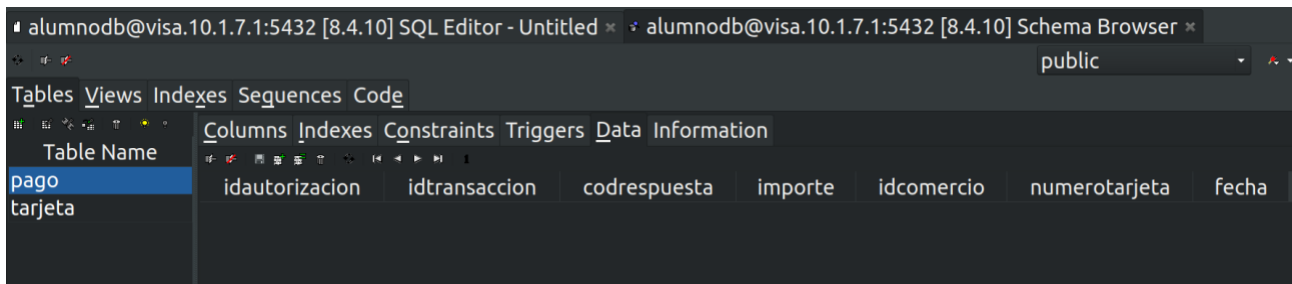


## Pago con tarjeta

Se han borrado 1 pagos correctamente para el comercio 1

[Volver al comercio](#)

Prácticas de Sistemas Informáticos II



**Cuestión 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.*

pago.html → ComienzaPago → formdatosvisa.jsp → ProcesaPag → formdatosvisa.jsp → error/muestraerror.jsp

**Cuestión 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, y cuáles son los encargados de procesarla?*

El encargado de solicitar esta información es ComienzaPago.

**Question 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?*

Los datos solicitados son siempre:

- Número de tarjeta
- Titular
- Fecha de emisión
- Fecha de caducidad

Estos datos se almacenan en una instancia de PagoBean y se utilizan en ProcesaPago

**Cuestión 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?**

La principal diferencia es la forma y orden de introducir los datos,

En la página de pruebas extendidas *testbd.jsp*, se introducen todos los datos necesarios en la misma página y luego se invoca al *servelet ComienzaPago*.

En la página *pago.html*, primero se introducen los datos:

- *idTransaccion*
- *idComercio*
- *importe*

Después se llama al *servelet* *comienza pago* con los datos solicitados y se rellenan los que faltan en otra plantilla *html*.

Ambos funcionan a la perfección porque ambos utilizan el *servelet ComienzaPago*.