		<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>	<b>1</b>	<b>Fecha</b>	<b>28/02/2018</b>
<b>Alumno/a</b>		Marroyo, Bouzada, Miguel Ángel			
<b>Alumno/a</b>		Navas, Ten, Raúl			

## Práctica 1: Arquitectura de Java EE – Parte 1

### Introducción

Antes de proceder a realizar cualquier ejercicio, es necesario cambiar la variable de entorno utilizando el siguiente comando: `export J2EE_HOME=/usr/local/glassfish-4.1.1/glassfish`. En el PDF informativo de pasos previos antes de realizar la práctica para configurar la máquina virtual se hace referencia a la versión de *Glassfish* 4.0, no obstante en los laboratorios (y en la versión de nuestros PCs personales) se cuenta con la versión 4.1.1.

La IP asignada a nuestra máquina virtual es 10.9.9.1. Para el cliente, 10.9.9.2.

La IP asignada a la máquina host por el script `virtualip.sh` es siempre de la forma 10.9.9.X, siendo X variable en función del ordenador que se use.

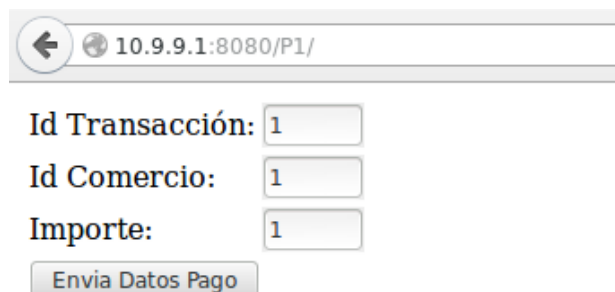
### Ejercicio número 1:

Para realizar el ejercicio ha sido necesario modificar los siguientes ficheros:

- *Build.properties*, en el que se ha introducido el valor de la IP generada en el campo `as.host` = 10.9.9.1
- *Postgresql.properties*, en el que se ha modificado `db.host` = 10.9.9.1 y `db.client.host` = 10.9.9.1

Para realizar el pago, se han utilizado los siguientes datos extraídos de la base de datos:

('2347 4840 5058 7931','Gabriel Avila Locke','11/09','01/20','207');



← 10.9.9.1:8080/P1/comienzapago

## Pago con tarjeta

Numero de visa: 2347 4840 5058 7931

Titular: Gabriel Avila Locke

Fecha Emisión: 11/09

Fecha Caducidad: 01/20

CVV2: 207

Pagar

---

Id Transacción: 1  
 Id Comercion: 1  
 Importe: 1.0

---

Prácticas de Sistemas Informáticos II

Introducimos los datos obtenidos de la base de datos y procedemos a pagar.

← 10.9.9.1:8080/P1/procesapago

## Pago con tarjeta

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

idTransaccion: 1  
 idComercio: 1  
 importe: 1.0  
 codRespuesta: 000  
 idAutorizacion: 1

[Volver al comercio](#)

---

Prácticas de Sistemas Informáticos II

alumnodb@visa.10.9.9.1:5432 [8.4.10] SQL Editor - Untitled x alumnodb@visa.10.9.9.1:5432 [8.4.10] Schema Browser x

public

Tables Views Indexes Sequences Code

Columns Indexes Constraints Triggers Data Information

Table Name  
 pago  
 tarjeta

idautorizacion	idtransaccion	codrespuesta	importe	idcomercio	numerotarjeta	fecha
1	1	000	1	1	2347 4840 5058 7931	15/02/18 09:00

Dentro de la base de datos, buscamos el nuevo pago y comprobamos que ha sido realizado con éxito y que se han guardado los datos correctamente.

Con la página de pruebas extendidas, probamos a borrar un pago.

## Pago con tarjeta

Lista de pagos del comercio 1

idTransaccion	Importe	codRespuesta	idAutorizacion
1	1.0	000	1

[Volver al comercio](#)

Prácticas de Sistemas Informáticos II

Comprobamos primero que efectivamente el pago había sido realizado con éxito.

## Pago con tarjeta

Se han borrado 1 pagos correctamente para el comercio 1

[Volver al comercio](#)

Prácticas de Sistemas Informáticos II

## Pago con tarjeta

Lista de pagos del comercio 1

idTransaccion	Importe	codRespuesta	idAutorizacion
---------------	---------	--------------	----------------

[Volver al comercio](#)

Prácticas de Sistemas Informáticos II

Al borrar el pago, ya no nos aparece en la lista.

### Ejercicio número 2:

Para el siguiente ejercicio se pide utilizar el modo de pago por *Direct Connection* o conexión directa. Para implementar dicho modo es necesario realizar los siguientes cambios en el fichero *DBTester.java*:

- `private static final String JDBC_DRIVER = "org.postgresql.Driver";`
- `private static final String JDBC_CONNSTRING = "jdbc:postgresql://10.9.9.1:5432/visa";`
- `private static final String JDBC_USER = "alumnodb";`

- `private static final String JDBC_PASSWORD = "alumnodb";`

Una vez realizados los cambios procedemos a probar dicha forma de pago. Utilizaremos los siguientes datos:

`('1111 2222 3333 4444','Jose García','11/09','11/20','123');`

The screenshot shows a web browser window with the address bar displaying `10.9.9.1:8080/P1/testbd.jsp`. The page title is **Pago con tarjeta**. Below the title is a section titled **Proceso de un pago**. The form contains the following fields and options:

- 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

At the bottom of the form is a **Pagar** button. Below the form is another browser window showing the address `10.9.9.1:8080/P1/procesapago`.

## Pago con tarjeta

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

idTransaccion: 1  
idComercio: 1  
importe: 1.0  
codRespuesta: 000  
idAutorizacion: 1

[Volver al comercio](#)

### Ejercicio número 3:

Dentro del gestor de *Glassfish* al que accedemos mediante la dirección `10.9.9.1:4848` podemos realizar un *ping* a la base de datos desde la pestaña de *JDBC Connection Pool*.

General
Advanced
Additional Properties

Ping Succeeded

### Edit JDBC Connection Pool

Modify an existing JDBC connection pool. A JDBC connection pool is a group of reusable connections for a particular database.

Load Defaults
Flush
Ping

#### General Settings

**Pool Name:** VisaPool

**Resource Type:** javax.sql.ConnectionPoolDataSource  
Must be specified if the datasource class implements more than 1 of the interface.

**Datasource Classname:** org.postgresql.ds.PGConnectionPoolDataSource  
Vendor-specific classname that implements the DataSource and/or XADataSource APIs

**Driver Classname:**  
Vendor-specific classname that implements the java.sql.Driver interface.

**Ping:** ☐ Enabled  
When enabled, the pool is pinged during creation or reconfiguration to identify and warn of any erroneous values for its attributes

**Deployment Order:** 100  
Specifies the loading order of the resource at server startup. Lower numbers are loaded first.

**Description:**

---

#### Pool Settings

**Initial and Minimum Pool Size:** 8 Connections  
Minimum and initial number of connections maintained in the pool

**Maximum Pool Size:** 32 Connections  
Maximum number of connections that can be created to satisfy client requests

**Pool Resize Quantity:** 2 Connections  
Number of connections to be removed when pool idle timeout expires

**Idle Timeout:** 300 Seconds  
Maximum time that connection can remain idle in the pool

**Max Wait Time:** 60000 Milliseconds  
Amount of time caller waits before connection timeout is sent

Los valores de los parámetros son los siguientes:

- *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*

Estos parámetros nos indican que nuestra aplicación sólo puede dar servicio a un máximo de 32 clientes, por lo que no sirve para sistemas de alta concurrencia/demanda. En cuanto a tiempo de espera antes de desconectarse, los valores actuales parecen un poco excesivos para un servidor de compra con tarjeta. Quizás sería necesario reducirlos ya que a la hora de realizar un pago *online* se espera que se realice de forma rápida sobretodo por temas de seguridad y satisfacción del cliente.

### Ejercicio número 4:

El primer método *getQryCompruebaTarjeta* comprueba si la tarjeta es válida haciendo una consulta SQL de tipo SELECT en la base de datos.

El segundo método *getQryInsertPago* efectúa el pago haciendo una consulta SQL de tipo INSERT en la base de datos.

```
String getOryCompruebaTarjeta(TarjetaBean tarjeta) {
    String qry = "select * from tarjeta "
        + "where numeroTarjeta='" + tarjeta.getNumero()
        + "' and titular='" + tarjeta.getTitular()
        + "' and validaDesde='" + tarjeta.getFechaEmision()
        + "' and validaHasta='" + tarjeta.getFechaCaducidad()
        + "' and codigoVerificacion='" + tarjeta.getCodigoVerificacion() + "'";
    return qry;
}

/**
 * getQryInsertPago
 */
String getOryInsertPago(PagoBean pago) {
    String qry = "insert into pago("
        + "idTransaccion,"
        + "importe,idComercio,"
        + "numeroTarjeta)"
        + " values ("
        + "'" + pago.getIdTransaccion() + "',"
        + pago.getImporte() + ","
        + "'" + pago.getIdComercio() + "',"
        + "'" + pago.getTarjeta().getNumero() + "'"
        + ")";
    return qry;
}
```

## Ejercicio número 5:

Dentro de *VisaDAO.java* basta con buscar el término *errorLog* para encontrar en qué partes del código se realiza la acción de escribir datos en el fichero de *log*. Dichas partes son a la hora de comprobar si la tarjeta es válida, a la hora de obtener el registro de los pagos realizados y al borrar pagos.

A continuación se muestra la forma que tiene el log del servidor, viendo en detalle algunas de sus partes cuando se ha intentado hacer una compra de forma incorrecta.

Log Viewer - Mozilla Firefox

https://10.9.9.1:4848/common/logViewer/logViewer.js?instanceName=server&logLevel=INFO&viewResults=true

**Log Viewer**

View search, and filter a server log file using basic and advanced options. Refer to the Log Levels page for information about log levels you can filter here.

[Advanced Search](#)

**Search Criteria**

Text search:

Only log entries containing the specified text will be displayed. Search is case sensitive.

Timestamp: ☒ Most Recent ☐ Specific Range:

Log Level:  ☐ Do not include more severe messages

Log entries are limited to those stored in the log file. Set appropriate log level in the Log Level page to ensure data is logged.

[Search](#) [Close](#)

**Modify Search**

Instance:

Log File:

**Log Viewer Results (40)**

Records before 1289 | Log File: Record Numbers 1289 through 1328 | Records after 1328 | [Full](#)

Record Number	Log Level	Message	Logger	Timestamp	Name-Value Pairs
1328	INFO	WebModule[null] ServletContext.log(): [ERROR] Page incorrecto(details)	javaax.enterprise.web	Feb 21, 2018 08:28:13.952	[levelValue=800, timeMills=1519230493952]
1327	SEVERE	[directConnectionFailed] org.postgresql.util.PSQLException: ERROR: duplicate key value violates unique constraint "tarjeta_pkey" (details)	javaax.enterprise.web	Feb 21, 2018 08:28:13.952	[levelValue=1000, timeMills=1519230493952]
1326	SEVERE	[directConnectionFailed] insert into pago(idTransaccion,importe,idComercio,numeroTarjeta) values (1, ... (details)	javaax.enterprise.web	Feb 21, 2018 08:28:13.950	[levelValue=1000, timeMills=1519230493950]
1325	SEVERE	[directConnectionFailed] select * from tarjeta where numeroTarjeta=1111.2222.3333.4444 and titular= ... (details)	javaax.enterprise.web	Feb 21, 2018 08:28:13.942	[levelValue=1000, timeMills=1519230493942]
1324	INFO	WebModule[null] ServletContext.log(): [INFO] Acceso correcto(procesapago)(details)	javaax.enterprise.web	Feb 21, 2018 08:28:13.834	[levelValue=800, timeMills=1519230493834]
1323	INFO	WebModule[null] ServletContext.log(): [INFO] Acceso correcto(festibol.jpg)(details)	javaax.enterprise.web	Feb 21, 2018 08:27:43.580	[levelValue=800, timeMills=1519230463580]
1322	WARNING	StandardWrapperValve[javax.facesServlet] Servlet.service() for servlet FacesServlet threw exception java. ... (details)	javaax.enterprise.web	Feb 21, 2018 07:50:53.690	[levelValue=900, timeMills=1519228253690]
1321	INFO	Exception when handling error trying to reset the response. java.io.IOException: Broken pipe at sun. ... (details)	javaax.enterprise.resource.webcontainer.jsf.context	Feb 21, 2018 07:50:53.688	[levelValue=800, timeMills=1519228253688]
1320	INFO	Admin Console: Initializing Session Attributes... (details)	org.glassfish.admingui	Feb 21, 2018 07:50:47.904	[levelValue=800, timeMills=1519228247904]
1319	INFO	Redirecting to / (details)	org.glassfish.admingui	Feb 21, 2018 07:50:47.631	[levelValue=800, timeMills=1519228247631]
1318	INFO	Listening to REST requests at context: /management/domain (details)	javaax.enterprise.adm.rest	Feb 21, 2018 07:50:47.596	[levelValue=800, timeMills=1519228247596]
1317	INFO	Initializing Jersey application, version Jersey: 2.0 2013-05-03 14:50:15. (details)	org.glassfish.jersey.server.ApplicationHandler	Feb 21, 2018 07:50:47.405	[levelValue=800, timeMills=1519228247405]
1316	SEVERE	SEC9054: Certificate has expired. ([ Version: V3 Subject: CN=GTCE CyberTrust Root 5, OU=GTCE Cyb... (details)	javaax.enterprise.system.ssl.security.com.sun.enterprise.security.ssl.impl	Feb 21, 2018 07:50:46.399	[levelValue=1000, timeMills=1519228246399]
1315	SEVERE	SEC9054: Certificate has expired. ([ Version: V3 Subject: CN=GTCE CyberTrust Root 5, OU=GTCE Cyb... (details)	javaax.enterprise.system.ssl.security.com.sun.enterprise.security.ssl.impl	Feb 21, 2018 07:50:38.900	[levelValue=1000, timeMills=1519228238900]
1314	SEVERE	SEC9054: Certificate has expired. ([ Version: V3 Subject: CN=GTCE CyberTrust Root 5, OU=GTCE Cyb... (details)	javaax.enterprise.system.ssl.security.com.sun.enterprise.security.ssl.impl	Feb 21, 2018 07:50:29.062	[levelValue=1000, timeMills=1519228229062]
1313	WARNING	Content path from ServletContext: differs from path from bundle. (details)	javaax.enterprise.system.container.web.com.sun.web.security	Feb 21, 2018 07:50:28.615	[levelValue=900, timeMills=1519228228615]
1312	INFO	Loading application ..._admingui done in 2.368 ms(details)	javaax.enterprise.system.core	Feb 21, 2018 07:50:27.809	[levelValue=800, timeMills=1519228227809]
1311	INFO	Loading application ..._admingui at / (details)	javaax.enterprise.web	Feb 21, 2018 07:50:27.808	[levelValue=800, timeMills=1519228227808]
1310	INFO	Installing Mojarra 2.2.0 (20130902-211) https://www.java.net/forums/mojara-servletapi/2.2.0(11930) to... (details)	javaax.enterprise.web	Feb 21, 2018 07:50:26.482	[levelValue=800, timeMills=1519228226482]
1309	INFO	this.makeModuleFor(org.glassfish.main.admingui.console.common.full.plugin, 4.0.0) returned OSGModule... (details)	javaax.enterprise.resource.webcontainer.jsf.config	Feb 21, 2018 07:50:26.006	[levelValue=800, timeMills=1519228226006]
1308	INFO	this.makeModuleFor(org.glassfish.main.admingui.console.common.full.plugin, 4.0.0) returned OSGModule... (details)	javaax.enterprise.resource.webcontainer.jsf.config	Feb 21, 2018 07:50:26.006	[levelValue=800, timeMills=1519228226006]
1307	INFO	this.makeModuleFor(org.glassfish.main.admingui.console.common.full.plugin, 4.0.0) returned OSGModule... (details)	javaax.enterprise.resource.webcontainer.jsf.config	Feb 21, 2018 07:50:26.006	[levelValue=800, timeMills=1519228226006]
1306	INFO	this.makeModuleFor(org.glassfish.main.admingui.console.common.full.plugin, 4.0.0) returned OSGModule... (details)	javaax.enterprise.resource.webcontainer.jsf.config	Feb 21, 2018 07:50:26.005	[levelValue=800, timeMills=1519228226005]
1305	INFO	this.makeModuleFor(org.glassfish.main.admingui.console.common.full.plugin, 4.0.0) returned OSGModule... (details)	javaax.enterprise.resource.webcontainer.jsf.config	Feb 21, 2018 07:50:26.005	[levelValue=800, timeMills=1519228226005]
1304	INFO	this.makeModuleFor(org.glassfish.main.admingui.console.common.full.plugin, 4.0.0) returned OSGModule... (details)	javaax.enterprise.resource.webcontainer.jsf.config	Feb 21, 2018 07:50:26.005	[levelValue=800, timeMills=1519228226005]
1303	INFO	this.makeModuleFor(org.glassfish.main.admingui.console.common.full.plugin, 4.0.0) returned OSGModule... (details)	javaax.enterprise.resource.webcontainer.jsf.config	Feb 21, 2018 07:50:26.004	[levelValue=800, timeMills=1519228226004]
1302	INFO	this.makeModuleFor(org.glassfish.main.admingui.console.common.full.plugin, 4.0.0) returned OSGModule... (details)	javaax.enterprise.resource.webcontainer.jsf.config	Feb 21, 2018 07:50:26.004	[levelValue=800, timeMills=1519228226004]
1301	INFO	this.makeModuleFor(org.glassfish.main.admingui.console.common.full.plugin, 4.0.0) returned OSGModule... (details)	javaax.enterprise.resource.webcontainer.jsf.config	Feb 21, 2018 07:50:26.004	[levelValue=800, timeMills=1519228226004]
1300	INFO	this.makeModuleFor(org.glassfish.main.admingui.console.common.full.plugin, 4.0.0) returned OSGModule... (details)	javaax.enterprise.resource.webcontainer.jsf.config	Feb 21, 2018 07:50:26.004	[levelValue=800, timeMills=1519228226004]
1299	INFO	this.makeModuleFor(org.glassfish.main.admingui.console.common.full.plugin, 4.0.0) returned OSGModule... (details)	javaax.enterprise.resource.webcontainer.jsf.config	Feb 21, 2018 07:50:26.003	[levelValue=800, timeMills=1519228226003]

## Log Entry Detail

**Timestamp** Feb 21, 2018 08:28:13.952  
**Log Level** SEVERE  
**Logger**  
**Name-Value Pairs** {levelValue=1000, timeMillis=1519230493952}  
**Record Number** 1327  
**Message ID**  
**Complete Message** [directConnection=false] org.postgresql.util.PSQLException: ERROR: duplicate key value violates unique constraint "pago\_uc"

## Log Entry Detail

**Timestamp** Feb 21, 2018 08:28:13.950  
**Log Level** SEVERE  
**Logger**  
**Name-Value Pairs** {levelValue=1000, timeMillis=1519230493950}  
**Record Number** 1326  
**Message ID**  
**Complete Message** [directConnection=false] insert into pago(idTransaccion,importe,idComercio,numeroTarjeta) values ('1',1.0,'1','1111 2222 3333 4444')

## Log Entry Detail

**Timestamp** Feb 21, 2018 08:28:13.942  
**Log Level** SEVERE  
**Logger**  
**Name-Value Pairs** {levelValue=1000, timeMillis=1519230493942}  
**Record Number** 1325  
**Message ID**  
**Complete Message** [directConnection=false] select \* from tarjeta where numeroTarjeta='1111 2222 3333 4444' and titular='Jose Garcia' and validaDesde='11/09' and validaHasta='11/20' and codigoVerificacion='123'

## Ejercicio número 6:

¿Por qué se ha de alterar el parámetro de retorno del método `realizaPago()` para que devuelva el pago el lugar de un boolean?

Es necesario alterar el parámetro de retorno del método *realizaPago* porque ya no basta con comprobar que el pago sea *true* o correcto, si no que es necesario obtener algunos de sus atributos para poder realizar operaciones posteriores.

## Ejercicio número 7:

The screenshot shows the GlassFish Server Open Source Edition web console. The left sidebar displays a tree view of the server's configuration, with 'P1-ws-ws' selected under the 'Applications' section. The main content area displays the 'Web Service Endpoint Information' for this service. The information includes the Application Name, Tester, WSDL, Endpoint Name, Service Name, Port Name, Deployment Type, Implementation Type, Implementation Class Name, Endpoint Address URI, and Namespace.

Property	Value
Application Name	P1-ws-ws
Tester	/P1-ws-ws/VisaDAOWSService?Tester
WSDL	/P1-ws-ws/VisaDAOWSService?wsdl
Endpoint Name	VisaDAOWS
Service Name	VisaDAOWSService
Port Name	VisaDAOWSPort
Deployment Type	109
Implementation Type	SERVLET
Implementation Class Name	ssii2.visa.dao.VisaDAOWS
Endpoint Address URI	/P1-ws-ws/VisaDAOWSService
Namespace	http://dao.visa.ssii2/

---

```

1 <?xml version='1.0' encoding='UTF-8'?><!-- Published by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is Metro/2.3 (tags/2.3-75)
2 <types>
3 <xsd:schema>
4 <xsd:import namespace="http://dao.visa.ssi2/" schemaLocation="http://10.9.9.1:8080/P1-ws-ws/VisaDAOWSService?xsd=1"/>
5 </xsd:schema>
6 </types>
7 <message name="isDebug">
8 <part name="parameters" element="tns:isDebug"/>
9 </message>
10 <message name="isDebugResponse">
11 <part name="parameters" element="tns:isDebugResponse"/>
12 </message>
13 <message name="setDebug">
14 <part name="parameters" element="tns:setDebug"/>
15 </message>
16 <message name="setDebugResponse">
17 <part name="parameters" element="tns:setDebugResponse"/>
18 </message>
19 <message name="compruebaTarjeta">
20 <part name="parameters" element="tns:compruebaTarjeta"/>
21 </message>
22 <message name="compruebaTarjetaResponse">
23 <part name="parameters" element="tns:compruebaTarjetaResponse"/>
24 </message>
25 <message name="realizaPago">
26 <part name="parameters" element="tns:realizaPago"/>
27 </message>
28 <message name="realizaPagoResponse">
29 <part name="parameters" element="tns:realizaPagoResponse"/>
30 </message>
31 <message name="getPagos">
32 <part name="parameters" element="tns:getPagos"/>
33 </message>
34 <message name="getPagosResponse">
35 <part name="parameters" element="tns:getPagosResponse"/>
36 </message>
37 <message name="delPagos">
38 <part name="parameters" element="tns:delPagos"/>
39 </message>
40 <message name="delPagosResponse">
41 <part name="parameters" element="tns:delPagosResponse"/>
42 </message>
43 <message name="isPrepared">
44 <part name="parameters" element="tns:isPrepared"/>
45 </message>
46 <message name="isPreparedResponse">
47 <part name="parameters" element="tns:isPreparedResponse"/>
48 </message>
49 <message name="setPrepared">
50 <part name="parameters" element="tns:setPrepared"/>
51 </message>
52 <message name="setPreparedResponse">
53 <part name="parameters" element="tns:setPreparedResponse"/>
54 </message>
55 <message name="errorLog">
56 <part name="parameters" element="tns:errorLog"/>

```

- **¿En qué fichero están definidos los tipos de datos intercambiados con el webservice?**  
`schemaLocation="http://10.9.9.1:8080/P1-ws-ws/VisaDAOWSService?xsd=1"`
- **¿Qué tipos de datos predefinidos se usan?**
- **¿Cuáles son los tipos de datos que se definen?**
- **¿Qué etiqueta está asociada a los métodos invocados en el webservice?**  
`<operation name = "..."> </operation>`
- **¿Qué etiqueta describe los mensajes intercambiados en la invocación de los métodos del webservice?**  
`<message name = "nombre_del_metodo"> </message>`
- **¿En qué etiqueta se especifica el protocolo de comunicación con el webservice?**  
 En las primeras líneas de código, HTML 1.0.
- **¿En qué etiqueta se especifica la URL a la que se deberá conectar un cliente para acceder al webservice?**  
`<service name="VisaDAOWSService">`  
`<port name="VisaDAOWSPort" binding="tns:VisaDAOWSPortBinding">`  
`<soap:address location="http://10.9.9.1:8080/P1-ws-ws/VisaDAOWSService"/>`  
`</port>`  
`</service>`



### Ejercicio número 8:

Los cambios realizados en *ProcesaPago.java* han sido los siguientes:

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

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

Es importante que en el fichero se hayan realizado los siguientes *imports*:

```
import ssii2.visa.VisaDAOWSService;
import ssii2.visa.VisaDAOWS;
import javax.xml.ws.WebServiceRef;
import javax.xml.ws.*;
```

### Ejercicio número 9:

Dentro de *web.xml* es necesario eliminar los comentarios y modificar el siguiente fragmento de código:

```
<context-param>
    <param-name>webmaster</param-name>
    <param-value>http://10.9.9.1:8080/P1-ws-ws/VisaDAOWSService</param-value>
</context-param>
```

La URL se obtiene desde el fichero XML de WSDL, en el apartado de *<service>*.

### Ejercicio número 10:

Para *DelPagos.java* sólo es necesario realizar el mismo cambio que para *ProcesaPago.java*:

```
//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"));
```

Para *GetPagos.java* hacemos el mismo cambio que para *DelPagos.java* con el añadido de que es necesario transformar el tipo de dato *ArrayList<PagoBean>* a *PagoBean[]*.

```

//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"));

/* Se recoge de la petici&oacute;n el par&aacute;metro idComercio*/
String idComercio = request.getParameter(PARAM_ID_COMERCIO);

/* Petici&oacute;n de los pagos para el comercio */
List<PagoBean> listapagos = new ArrayList<PagoBean>();
listapagos = dao.getPagos(idComercio);

PagoBean[] pagos = new PagoBean[listapagos.size()];
pagos = listapagos.toArray(pagos);

request.setAttribute(ATTR_PAGOS, pagos);
reenvia("/listapagos.jsp", request, response);
return;

```

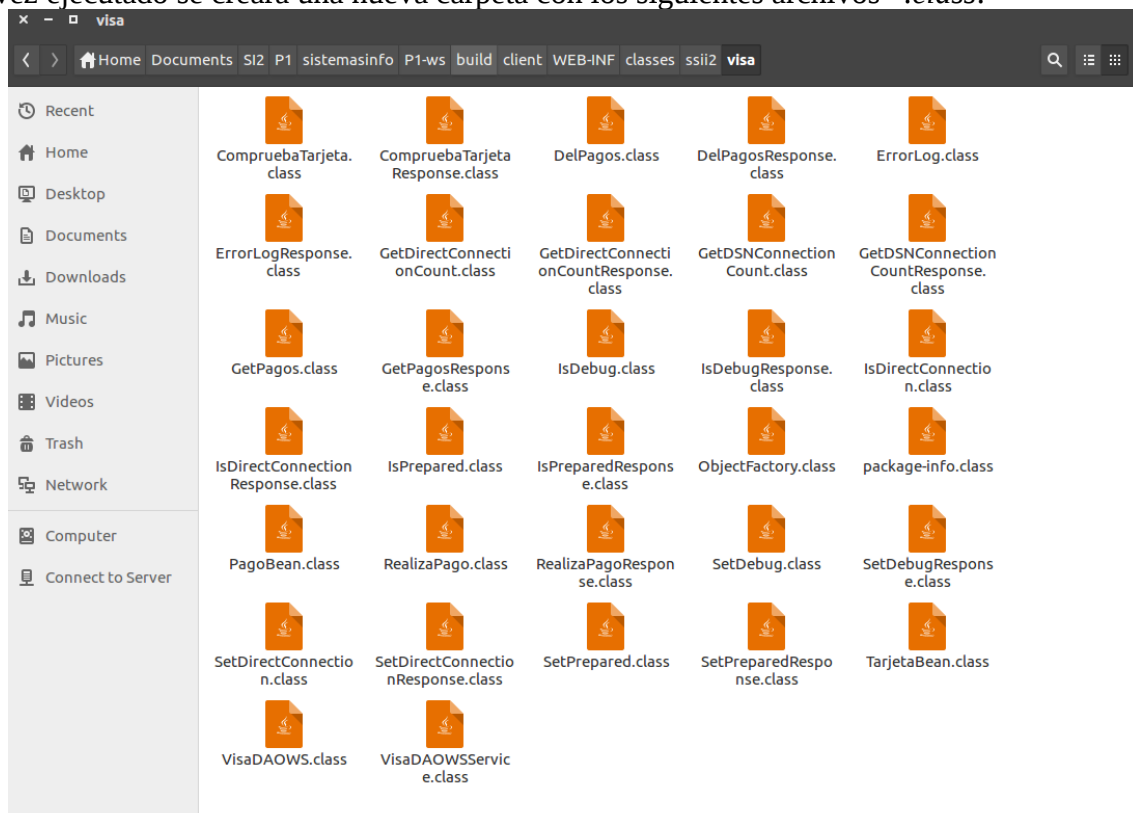
De nuevo, es importante hacer los *imports* mencionados en el ejercicio 8 dentro de los ficheros *DelPagos.java* y *GetPagos.java*.

### Ejercicio número 11:

Para realizar la importación del WSDL es necesario ejecutar el siguiente comando:

`wsimport -d build/client/WEB-INF/classes -p ssii2.visa http://10.9.9.1:8080/P1-ws-ws/VisaDAOWSService?wsdl`

Una vez ejecutado se creará una nueva carpeta con los siguientes archivos *\*.class*:



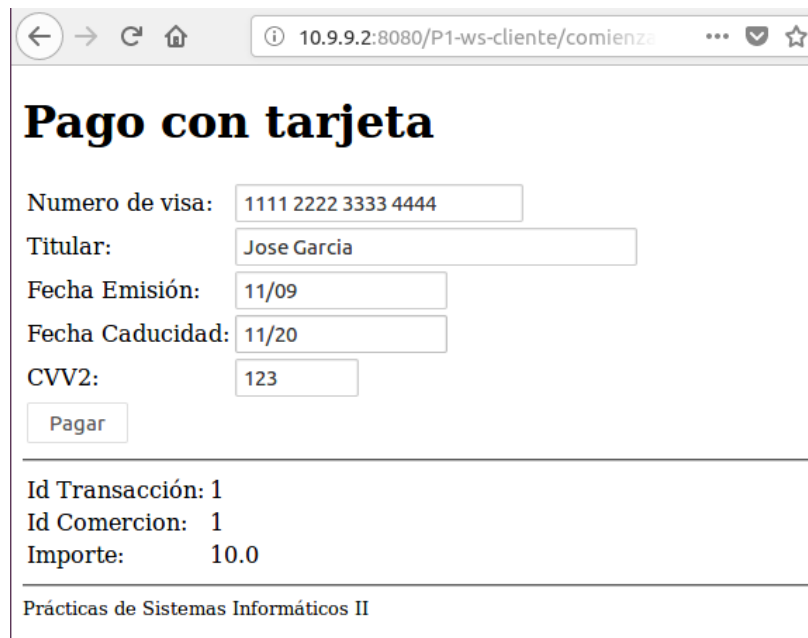
## Ejercicio número 12:

Para generar los *stubs* desde *build.xml* añadimos al *target generar-stubs* la siguiente información, que no es más que el comando del ejercicio anterior con el formato adecuado para que se ejecute con la herramienta *ant*.

```
<target name="generar-stubs" depends="montar-jerarquia" description="Genera los stubs del cliente a partir del archivo WSDL">
  <!-- TODO - Implementar llamada wsimport -->
  <delete file="${build}/${tmpvisaclientjar}" />
  <jar jarfile="${build}/${tmpvisaclientjar}" >
    <fileset dir="${build.client}/WEB-INF/classes" />
  </jar>
  <move file="${build}/${tmpvisaclientjar}" todir="${build.client}/WEB-INF/lib" />
  <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>
</target>
```

## Ejercicio número 13:

Una vez compilado, empaquetado y desplegado tanto servicio como cliente, podemos acceder a la siguiente página de pagos:



← → ↻ 🏠 ⓘ 10.9.9.2:8080/P1-ws-cliente/comienza ... 📧 ☆

## Pago con tarjeta

Numero de visa:

Titular:

Fecha Emisión:

Fecha Caducidad:

CVV2:

---

Id Transacción: 1  
Id Comercion: 1  
Importe: 10.0

---

Prácticas de Sistemas Informáticos II

En nuestro caso, incluso introduciendo los datos de forma correcta el sistema nos dice que el pago es incorrecto. Es posible que sea debido a algún fallo dentro de *ProcesaPago.java*.

## 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.

Se pasaría por los siguientes pasos:

- Servlet *ComienzaPago*
- *Formdatosvisa.jsp*
- Servlet *ProcesaPago*
- *JavaBean VisaDAO*
- *Error/muestraerror.jsp*

### ***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?**

Los servlets encargados son: *ProcesaPago* para procesar la petición y *GetPagos* para obtener los datos sobre los pagos hechos en un comercio.

### ***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?**

El servlet *ProcesaPago* solicita todos los campos necesarios de la tarjeta (Número de tarjeta, titular, fecha de emisión, fecha de caducidad y CCV).

Los datos se guardan en la base de datos VisaDB, donde se guarda la información de las tarjetas y de los pagos.

### ***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?**

La página de pruebas extendida cuenta con una serie *flags* como *debugging* o conexión directa. No obstante dichos *flags* no afectan directamente al funcionamiento de la página en cuanto a funcionalidad.