		Escuela Politécnica Superior Ingeniería Informática Prácticas de Sistemas Informáticos 2			
Grupo	2401	Práctica	1A	Fecha	25/02/2018
Alumno	Alcover, Couso, Roberto				
Alumno	Aced, Fuentes, Emilio Samuel				

Práctica 1A: Arquitectura de JAVA EE (Parte 1)

Ejercicio 1:

Prepare e inicie una máquina virtual a partir de la plantilla si2srv con: 1GB de RAM asignada, 2 CPUs. A continuación:

- Modifique los ficheros que considere necesarios en el proyecto para que se despliegue tanto la aplicación web como la base de datos contra la dirección asignada a la pareja de prácticas.
- Realice un pago contra la aplicación web empleando el navegador en la ruta <http://10.1.1.1:8080/P1> Conéctese a la base de datos (usando el cliente Tora por ejemplo) y obtenga evidencias de que el pago se ha realizado.
- Acceda a la página de pruebas extendida, <http://10.1.1.1:8080/P1/testbd.jsp>. Compruebe que la funcionalidad de listado de y borrado de pagos funciona correctamente. Elimine el pago anterior.

Incluya en la memoria de prácticas todos los pasos necesarios para resolver este ejercicio así como las evidencias obtenidas.
Se pueden incluir por ejemplo capturas de pantalla.

Descomprimos el tar y cambiamos los campos as.host del fichero build.properties ,db.host del fichero postgresql.properties y db.client.host del fichero postgresql.properties poniendo como host la maquina virtual.

Realizamos el export mediante el comando :

```
export J2EE_HOME=/usr/local/glassfish-4.1.1/glassfish
```

Comprobamos que se ha abierto bien la conexion conectandonos a la maquina por el puerto 8080 al contexto P1

Probemos con la siguiente tarjeta:

```
'2347 4840 5058 7931','Gabriel Avila Locke','11/09','01/20','207'
```

Introducimos los datos de la transacción

← → 10.1.1.1:8080/P1/

Id Transacción:

Id Comercio:

Importe:

Introducimos los datos del cliente que va a hacer el pago:

← 10.1.1.1:8080/P1/comienzapago

Pago con tarjeta

Numero de visa:

Titular:

Fecha Emisión:

Fecha Caducidad:

CVV2:

Id Transacción: 3108
Id Comercion: 1997
Importe: 21.0

Prácticas de Sistemas Informáticos II

Vemos que el pago se ha realizado correctamente:

← 10.1.1.1:8080/P1/procesapago

Pago con tarjeta

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

idTransaccion: 3108
idComercio: 1997
importe: 21.0
codRespuesta: 000
idAutorizacion: 1

[Volver al comercio](#)

Prácticas de Sistemas Informáticos II

Vemos en el log del servidor que el pago se ha realizado correctamente:

Log Viewer Results (40)				
Records before 1492		Log File Record Numbers 1492 through 1535		Records after 1535
Record Number	Log Level	Message	Logger	
1535	INFO	WebModule[null] ServletContext.log():[INFO] Acceso correcto:/pago.html(details)	javax.enterprise.web	
1534	INFO	WebModule[null] ServletContext.log():[INFO] Acceso correcto:/procesapago(details)	javax.enterprise.web	
1533	INFO	WebModule[null] ServletContext.log():[INFO] Acceso correcto:/comenzapago(details)	javax.enterprise.web	
1532	INFO	WebModule[null] ServletContext.log():[INFO] Acceso correcto:/pago.html(details)	javax.enterprise.web	

También en el fichero de logs:

```
[2018-02-13T08:29:09.754-0800] [glassfish 4.0] [INFO] [] [javax.enterprise.web] [tid: _ThreadID=21 _ThreadName=http-listener-1(5)] [timeMillis: 1518539349754] [levelValue: 800] [[WebModule[null] ServletContext.log():[INFO] Acceso correcto:/pago.html]]
[2018-02-13T08:29:27.340-0800] [glassfish 4.0] [INFO] [] [javax.enterprise.web] [tid: _ThreadID=17 _ThreadName=http-listener-1(1)] [timeMillis: 1518539367340] [levelValue: 800] [[WebModule[null] ServletContext.log():[INFO] Acceso correcto:/comenzapago]]
[2018-02-13T08:30:32.493-0800] [glassfish 4.0] [INFO] [] [javax.enterprise.web] [tid: _ThreadID=18 _ThreadName=http-listener-1(2)] [timeMillis: 1518539432493] [levelValue: 800] [[WebModule[null] ServletContext.log():[INFO] Acceso correcto:/procesapago]]
[2018-02-13T08:32:32.517-0800] [glassfish 4.0] [INFO] [] [javax.enterprise.web] [tid: _ThreadID=19 _ThreadName=http-listener-1(3)] [timeMillis: 151853952517] [levelValue: 800] [[WebModule[null] ServletContext.log():[INFO] Acceso correcto:/pago.html]]
```

Accedemos a testbd.jsp y hacemos getPagos del comercio 1997

10.1.1.1:8080/P1/getpagos

Pago con tarjeta

Lista de pagos del comercio 1997

idTransaccion	Importe	codRespuesta	idAutorizacion
3108	21.0	000	1

[Volver al comercio](#)

Prácticas de Sistemas Informáticos II

Confirmamos que el pago está en la base de datos con TOra

#	^	idautorizacion	idtransaccion	codrespuesta	importe	idcomercio	numerotarjeta	fecha
1	1		3108	000	21	1997	2347 4840 5058 7931	13/02/18 08:30

Borramos los pagos con testbd:

10.1.1.1:8080/P1/delpagos

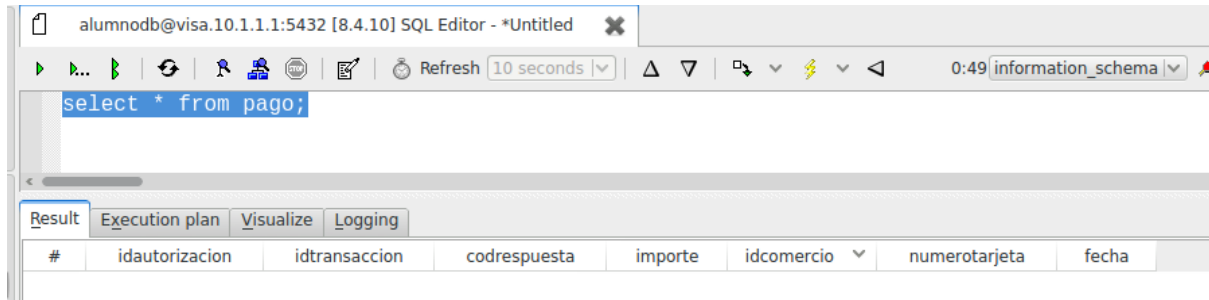
Pago con tarjeta

Se han borrado 1 pagos correctamente para el comercio 1997

[Volver al comercio](#)

Prácticas de Sistemas Informáticos II

Vemos con Tora que efectivamente el pago se ha borrado.



Ejercicio 2:

La clase VisaDAO implementa los dos tipos de conexión descritos anteriormente, los cuales son heredados de la clase DBTester. Sin embargo, la configuración de la conexión utilizando la conexión directa es incorrecta. Se pide completar la información necesaria para llevar a cabo la conexión directa de forma correcta. Para ello habrá que fijar los atributos a los valores correctos. En particular, el nombre del driver JDBC a utilizar, el JDBC connection string que se debe corresponder con el servidor postgresql, y el nombre de usuario y la contraseña. Es necesario consultar el apéndice 10 para ver los detalles de cómo se obtiene una conexión de forma correcta. Una vez completada la información, acceda a la página de pruebas extendida, <http://10.1.1.1:8080/P1/testbd.jsp> y pruebe a realizar un pago utilizando la conexión directa y pruebe a listarlo y eliminarlo. Adjunte en la memoria evidencias de este proceso

Incluimos en el constructor de la clase DBTester la siguiente sentencia:

```
Class.forName("org.postgresql.Driver").newInstance();
```

Y en la función de getConnection, en la parte de conexión directa realizamos la conexión con nuestra maquina virtual :

```
DriverManager.getConnection("jdbc:postgresql://10.1.1.1:5432/visa", "alumnodb", "");
```


Probemos a realizar una conexión directa

```
'1530 6462 9686 4119','Alberto Mas Reyes','05/09','09/20','105'
```

The screenshot shows a web form titled 'Pago con tarjeta' with the subtitle 'Proceso de un pago'. The form contains the following fields and options:

- Id Transacción: 3989
- Id Comercio: 7212
- Importe: 332
- Numero de visa: 1530 6462 9686 4119
- Titular: Alberto Mas Reyes
- Fecha Emisión: 05/09
- Fecha Caducidad: 09/20
- CVV2: 105
- Modo debug: ☐ True ☐ False
- Direct Connection: ☒ True ☐ False
- Use Prepared: ☐ True ☐ False
- Pagar button

Pago realizado con éxito:

 10.1.1.1:8080/P1/procesapago

Pago con tarjeta

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

idTransaccion: 3989
idComercio: 7212
importe: 332.0
codRespuesta: 000
idAutorizacion: 1

[Volver al comercio](#)

Prácticas de Sistemas Informáticos II

Lista de pagos del comercio 7212

Pago con tarjeta

Lista de pagos del comercio 7212

idTransaccion	Importe	codRespuesta	idAutorizacion
3989	332.0	000	1

[Volver al comercio](#)

Prácticas de Sistemas Informáticos II

Borramos los pagos del comercio 7212

Pago con tarjeta

Se han borrado 1 pagos correctamente para el comercio 7212

[Volver al comercio](#)

Prácticas de Sistemas Informáticos II

Ejercicio 3:

Ejercicio 3. Examinar el archivo `postgresql.properties` para determinar el nombre del recurso JDBC correspondiente al `DataSource` y el nombre del pool. Acceda a la Consola de Administración. Compruebe que los recursos JDBC y pool de conexiones han sido correctamente creados. Realice un Ping JDBC a la base de datos. Anote en la memoria de la práctica los valores para los parámetros Initial and Minimum Pool Size, Maximum Pool Size, Pool Resize Quantity, Idle Timeout, Max Wait Time. Comente razonadamente qué impacto considera que pueden tener estos parámetros en el rendimiento de la aplicación.

`org.postgresql.ds.PGConnectionPoolDataSource`

Pool:

The screenshot shows the GlassFish Server Open Source Edition administration console. The left sidebar displays a tree view of the server's configuration, with 'JDBC' and 'JDBC Connection Pools' expanded. The 'VisaPool' is selected. The main panel shows the 'Edit JDBC Connection Pool' configuration for 'VisaPool'. The 'General Settings' tab is active, showing the following values:

- Pool Name:** VisaPool
- Resource Type:** `javax.sql.ConnectionPoolDataSource`
- Datasource Classname:** `org.postgresql.ds.PGConnectionPoolDataSource`
- Driver Classname:** (empty)
- Ping:** ☒ Enabled
- Deployment Order:** 100
- Description:** (empty)

The 'Pool Settings' section shows the following values:

- 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

Vemos que el ping se realiza:

The screenshot shows the same 'Edit JDBC Connection Pool' configuration page for 'VisaPool'. A yellow banner with a green checkmark and the text 'Ping Succeeded' is displayed at the top. Below the banner, the 'General Settings' section is visible, showing the same configuration values as the previous screenshot.

Initial and Minimum Pool Size : 8

Maximum Pool Size: 32

Pool Resize Quantity: 2

Idle Timeout: 300

Max Wait Time :60000

Ejercicio 4:

Ejercicio 4. Localice los siguientes fragmentos de código SQL dentro del proyecto proporcionado (P1-base) correspondientes a los siguientes procedimientos:

- Consulta de si una tarjeta es válida.

- Ejecución del pago. Incluya en la memoria de prácticas dichas consultas.

Comprobar si una tarjeta es valida:

Inicialización de la consulta:

```
private static final String INSERT_PAGOS_QRY =  
    "insert into pago(" +  
    "idTransaccion,importe,idComercio,numeroTarjeta)" +  
    " values (?, ?, ?, ?)";
```

Binding de los parámetros:

```
if (isPrepared() == true) {  
    String select = SELECT_TARJETA_QRY;  
    errorLog(select);  
    pstmt = con.prepareStatement(select);  
    pstmt.setString(1, tarjeta.getNumero());  
    pstmt.setString(2, tarjeta.getTitular());  
    pstmt.setString(3, tarjeta.getFechaEmision());  
    pstmt.setString(4, tarjeta.getFechaCaducidad());  
    pstmt.setString(5, tarjeta.getCodigoVerificacion());  
    rs = pstmt.executeQuery();  
  
} else {  
    /*****/  
    stmt = con.createStatement();  
    qry = getQryCompruebaTarjeta(tarjeta);  
    errorLog(qry);  
    rs = stmt.executeQuery(qry);  
  
}
```

Ejecución del pago:

Inicialización de la consulta:

```
private static final String INSERT_PAGOS_QRY =  
    "insert into pago(" +  
    "idTransaccion,importe,idComercio,numeroTarjeta)" +  
    " values (?, ?, ?, ?)";
```

Binding de los parámetros:

```

try {

    // Obtener conexion
    con = getConnection();

    // Insertar en la base de datos el pago

    /* TODO Usar prepared statement si
       isPrepared() == true */
    /***/
    if (isPrepared() == true) {
        String insert = INSERT_PAGOS_QRY;
        errorLog(insert);
        pstmt = con.prepareStatement(insert);
        pstmt.setString(1, pago.getIdTransaccion());
        pstmt.setDouble(2, pago.getImporte());
        pstmt.setString(3, pago.getIdComercio());
        pstmt.setString(4, pago.getTarjeta().getNumero());
        ret = false;
        if (!pstmt.execute()
            && pstmt.getUpdateCount() == 1) {
            ret = true;
        }
    }

    } else {
        /***/
        stmt = con.createStatement();
        String insert = getQryInsertPago(pago);
        errorLog(insert);
        ret = false;
        if (!stmt.execute(insert)
            && stmt.getUpdateCount() == 1) {
            ret = true;
        }
    }
}
}

```


Cuestión número 5:

Edite el fichero VisaDAO.java y localice el método errorLog. Compruebe en qué partes del código se escribe en log utilizando dicho método. Realice un pago utilizando la página testbd.jsp con la opción de debug activada. Visualice el log del servidor de aplicaciones y compruebe que dicho log contiene información adicional sobre las acciones llevadas a cabo en VisaDAO.java. Incluya en la memoria una captura de pantalla del log del servidor.

Creemos un pago con el método debug activado

Go forward one page
Right-click or pull down to show history

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

Vemos que se ha realizado con éxito:

Pago con tarjeta

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

idTransaccion: 4
idComercio: 1
importe: 30.0
codRespuesta: 000
idAutorizacion: 3

[Volver al comercio](#)

Prácticas de Sistemas Informáticos II

Vamos al Log Files de Glassfish para ver qué cambios ha habido en el log. Como podemos observar vemos las distintas operaciones que se han llevado a cabo en la base de datos para la creación del nuevo pago

Log Viewer - Mozilla Firefox

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.

Modify Search

Instance:

Log File:

Log Viewer Results (40)

Record Number	Log Level	Message	Logger	Timestamp	Name-Value Pairs
1462	SEVERE	[DirectConnection=false] select idAutorizacion, codRespuesta from pago where idTransaccion = '4...' (details)		Feb 16, 2018 13:05:03.098	[DevValue=1000, DevValue=151880779030982]
1461	SEVERE	[DirectConnection=false] insert into pago(idTransaccion,importe,idComercio,numeroTarjeta) values ('4...', (details)		Feb 16, 2018 13:05:03.097	[DevValue=1000, DevValue=151880779030977]
1460	SEVERE	[DirectConnection=false] select * from tarjeta where numeroTarjeta='2347 4840 5058 7931' and titular... (details)		Feb 16, 2018 13:05:03.096	[DevValue=1000, DevValue=151880779030962]
1459	INFO	WebModule(runt) ServletContext.log(): [INFO] Acceso correcto a base de datos (details)	javax.enterprise.web	Feb 16, 2018 13:05:03.094	[DevValue=1000, DevValue=151880779030942]
1458	SEVERE	[DirectConnection=false] select idAutorizacion, codRespuesta from pago where idTransaccion = '3...' (details)		Feb 16, 2018 13:05:03.090	[DevValue=1000, DevValue=151880772008902]
1457	SEVERE	[DirectConnection=false] insert into pago(idTransaccion,importe,idComercio,numeroTarjeta) values ('3...', (details)		Feb 16, 2018 13:05:03.089	[DevValue=1000, DevValue=151880772008892]
1456	SEVERE	[DirectConnection=false] select * from tarjeta where numeroTarjeta='2347 4840 5058 7931' and titular... (details)		Feb 16, 2018 13:05:03.084	[DevValue=1000, DevValue=151880772008842]
1455	INFO	WebModule(runt) ServletContext.log(): [INFO] Acceso correcto a base de datos (details)	javax.enterprise.web	Feb 16, 2018 13:05:03.089	[DevValue=1000, DevValue=151880772008892]
1454	INFO	WebModule(runt) ServletContext.log(): [INFO] Acceso correcto a base de datos (details)	javax.enterprise.web	Feb 16, 2018 13:05:03.084	[DevValue=1000, DevValue=151880772008842]
1453	INFO	WebModule(runt) ServletContext.log(): [WARNING] Acceso No autorizado a base de datos (details)	javax.enterprise.web	Feb 16, 2018 13:05:03.114	[DevValue=1000, DevValue=151880776451142]
1452	INFO	WebModule(runt) ServletContext.log(): [WARNING] Acceso No autorizado a base de datos (details)	javax.enterprise.web	Feb 16, 2018 13:05:03.395	[DevValue=1000, DevValue=151880776423952]
1451	INFO	Admin Console: Initializing Session Attributes... (details)	org.glassfish.adminqui	Feb 16, 2018 10:46:48.229	[DevValue=1000, DevValue=15188068052292]
1450	INFO	Admin Console: Initializing Session Attributes... (details)	org.glassfish.adminqui	Feb 16, 2018	[DevValue=1000, DevValue=15188068052292]

También hacemos lo propio abriendo con vim el fichero:
/opt/glassfish4/glassfish/domains/domain1/logs/server.log

y yendo al final del mismo. Nótese que la información está en orden inverso al del log suministrado por glassfish.

```
[2018-02-16T11:00:42.355-0800] [glassfish 4.0] [INFO] [] [javax.enterprise.web] [tid: _ThreadID=21 _ThreadName=http-listener-1(5)] [timeMillis: 1518807642355] [levelValue: 800] [[
  WebModule[null] ServletContext.log():[WARNING] Acceso No autorizado:/testdb.jsp]]

[2018-02-16T11:00:45.114-0800] [glassfish 4.0] [INFO] [] [javax.enterprise.web] [tid: _ThreadID=18 _ThreadName=http-listener-1(2)] [timeMillis: 1518807645114] [levelValue: 800] [[
  WebModule[null] ServletContext.log():[WARNING] Acceso No autorizado:/testdb.jsp]]

[2018-02-16T11:01:40.924-0800] [glassfish 4.0] [INFO] [] [javax.enterprise.web] [tid: _ThreadID=17 _ThreadName=http-listener-1(1)] [timeMillis: 1518807700924] [levelValue: 800] [[
  WebModule[null] ServletContext.log():[INFO] Acceso correcto:/testdb.jsp]]

[2018-02-16T11:02:00.069-0800] [glassfish 4.0] [INFO] [] [javax.enterprise.web] [tid: _ThreadID=21 _ThreadName=http-listener-1(5)] [timeMillis: 1518807720069] [levelValue: 800] [[
  WebModule[null] ServletContext.log():[INFO] Acceso correcto:/procesapago]]

[2018-02-16T11:02:00.084-0800] [glassfish 4.0] [SEVERE] [] [] [tid: _ThreadID=21 _ThreadName=Thread-4] [timeMillis: 1518807720084] [levelValue: 1000] [[
  [directConnection=false] select * from tarjeta where numeroTarjeta='2347 4840 5058 7931' and titular='Gabriel Avila Locke' and validaDesde='11/09' and validaHasta='01/20' and codigoVerificacion='207']]

[2018-02-16T11:02:00.089-0800] [glassfish 4.0] [SEVERE] [] [] [tid: _ThreadID=21 _ThreadName=Thread-4] [timeMillis: 1518807720089] [levelValue: 1000] [[
  [directConnection=false] insert into pago(idTransaccion,importe,idComercio,numeroTarjeta) values ('3',30.0,'1','2347 4840 5058 7931')]]

[2018-02-16T11:02:00.090-0800] [glassfish 4.0] [SEVERE] [] [] [tid: _ThreadID=21 _ThreadName=Thread-4] [timeMillis: 1518807720090] [levelValue: 1000] [[
  [directConnection=false] select idAutorizacion, codRespuesta from pago where idTransaccion = '3' and idComercio = '1']]

[2018-02-16T11:05:03.094-0800] [glassfish 4.0] [INFO] [] [javax.enterprise.web] [tid: _ThreadID=18 _ThreadName=http-listener-1(2)] [timeMillis: 1518807903094] [levelValue: 800] [[
  WebModule[null] ServletContext.log():[INFO] Acceso correcto:/procesapago]]

[2018-02-16T11:05:03.096-0800] [glassfish 4.0] [SEVERE] [] [] [tid: _ThreadID=18 _ThreadName=Thread-4] [timeMillis: 1518807903096] [levelValue: 1000] [[
  [directConnection=false] select * from tarjeta where numeroTarjeta='2347 4840 5058 7931' and titular='Gabriel Avila Locke' and validaDesde='11/09' and validaHasta='01/20' and codigoVerificacion='207']]

[2018-02-16T11:05:03.097-0800] [glassfish 4.0] [SEVERE] [] [] [tid: _ThreadID=18 _ThreadName=Thread-4] [timeMillis: 1518807903097] [levelValue: 1000] [[
  [directConnection=false] insert into pago(idTransaccion,importe,idComercio,numeroTarjeta) values ('4',30.0,'1','2347 4840 5058 7931')]]

[2018-02-16T11:05:03.098-0800] [glassfish 4.0] [SEVERE] [] [] [tid: _ThreadID=18 _ThreadName=Thread-4] [timeMillis: 1518807903098] [levelValue: 1000] [[
  [directConnection=false] select idAutorizacion, codRespuesta from pago where idTransaccion = '4' and idComercio = '1']]
~
```

Las partes del código de VisaDAO que escriben en el log son las que llaman a errorLog, a saber: borrar Pagos, buscar Pagos, realizar Pago, comprobacion Tarjeta.

Ejercicio 6:

Realícense las modificaciones necesarias en VisaDAOWS.java para que implemente de manera correcta un servicio web. Los siguientes métodos y todos sus parámetros deberán ser publicados como métodos del servicio.

compruebaTarjeta()

realizaPago()

sDebug() / setDebug()

(Nota: VisaDAO.java contiene dos métodos setDebug que reciben distintos argumentos. Solo uno de ellos podrá ser exportado como servicio web)3 .

isPrepared() / setPrepared() En la clase DBTester, de la que hereda VisaDAOWS.java, deberemos publicar así mismo:

isDirectConnection() / setDirectConnection()

Para ello, implemente estos métodos también en la clase hija. Es decir, haga un override de Java, implementando estos métodos en VisaDAOWS mediante invocaciones a la clase padre (super). Modifique así mismo el método realizaPago() para que éste devuelva el pago modificado tras la correcta o incorrecta realización del pago:

Con identificador de autorización y código de respuesta correcto en caso de haberse realizado. Con null en caso de no haberse podido realizar. Por último, conteste a la siguiente pregunta:

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

Necesitamos la instancia del objeto en la maquina cliente ya que ahora la aplicación está distribuida.

Ejercicio 7:

Despliegue el servicio con la regla correspondiente en el build.xml. Acceda al WSDL remotamente con el navegador e inclúyalo en la memoria de la práctica (habrá que asegurarse que la URL contiene la dirección IP de la máquina virtual donde se encuentra el servidor de aplicaciones). Comente en la memoria aspectos relevantes del código XML del fichero WSDL y su relación con los métodos Java del objeto del servicio, argumentos recibidos y objetos devueltos. Conteste a las siguientes preguntas:

View WSDL

If the server or listener is not running, the link may not work. In this case, check the status of the server instance. After launching the we return to this screen

Application Name: P1-ws-ws

Links: [server] <http://si2srv02:8080/P1-ws-ws/VisaDAOWSService?wsdl>
[server] <https://si2srv02:8181/P1-ws-ws/VisaDAOWSService?wsdl>

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

En el tag types del WSDL vienen los ficheros donde se definirán los tipos de datos:

<types>

<xsd:schema>

<xsd:import namespace="http://dao.visa.ssii2/" schemaLocation="<http://10.1.1.2:8080/P1-ws-ws/VisaDAOWSService?xsd=1>"/>

</xsd:schema>

</types>

En este fichero <http://10.1.1.2:8080/P1-ws-ws/VisaDAOWSService?xsd=1> se definen los tipos de datos a usar:

```
<!--
  Published by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is Metro/2.3 (tags/2.3-7528; 2013-04-29T19:34:10+0000) JAXWS-RI/2.2.8 JAXWS/2.2 svn-revision#unknown.
-->
<xs:schema version="1.0" targetNamespace="http://dao.visa.ssi2/">
  <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="errorLog" type="tns:errorLog"/>
  <xs:element name="errorLogResponse" type="tns:errorLogResponse"/>
  <xs:element name="getDSNConnectionCount" type="tns:getDSNConnectionCount"/>
  <xs:element name="getDSNConnectionCountResponse" type="tns:getDSNConnectionCountResponse"/>
  <xs:element name="getDirectConnectionCount" type="tns:getDirectConnectionCount"/>
  <xs:element name="getDirectConnectionCountResponse" type="tns:getDirectConnectionCountResponse"/>
  <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"/>
  <xs:complexType name="getDirectConnectionCount">
    <xs:sequence/>
  </xs:complexType>
  <xs:complexType name="getDirectConnectionCountResponse">
    <xs:sequence>
      <xs:element name="return" type="xs:int"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="realizaPago">
    <xs:sequence>
      <xs:element name="nacho" type="tns:nachoBean" minOccurs="0"/>
    </xs:sequence>
  </xs:complexType>
```

¿Qué tipos de datos predefinidos se usan?

los que vienen con el tipo precedido por xs, a saber,
xs:int, xs:string, xs:double, xs:boolean

¿Cuáles son los tipos de datos que se definen?

los que vienen precedido por tns, a saber,
tns:tarjetaBean , tns:pagoBean etc...

¿Qué etiqueta está asociada a los métodos invocados en el webservice?

<Operation>

¿Qué etiqueta describe los mensajes intercambiados en la invocación de los métodos del webservice?

<message>

¿En qué etiqueta se especifica el protocolo de comunicación con el webservice?

Sabemos que binding nos dice el modo en el que se transmiten los mensajes sobre un protocolo de RPC.

<binding name="VisaDAOWSPortBinding" type="tns:VisaDAOWS"><soap:binding
transport="http://schemas.xmlsoap.org/soap/http" style="document"/>

¿En qué etiqueta se especifica la URL a la que se deberá conectar un cliente para acceder al

```

webservice?
soap:address
<service name="VisaDAOWSService">
    <port name="VisaDAOWSPort" binding="tns:VisaDAOWSPortBinding">
        <soap:address location="http://10.1.1.2:8080/P1-ws-ws/VisaDAOWSService"/>
    </port>
</service>

```

Ejercicio 8:

Realícese las modificaciones necesarias en `ProcesaPago.java` para que implemente de manera correcta la llamada al servicio web mediante stubs estáticos. Téngase en cuenta que:
 □ El nuevo método `realizaPago()` ahora no devuelve un boolean, sino el propio objeto `Pago` modificado.
 □ Las llamadas remotas pueden generar nuevas excepciones que deberán ser tratadas en el código cliente.

Realizado en el código. Ver el fichero: `P1-ws/src/client/ssii2/controlador/ProcesaPago.java`

Ejercicio 9:

Modifique la llamada al servicio para que la ruta al servicio remoto se obtenga del fichero de configuración `web.xml`. Para saber cómo hacerlo consulte el apéndice 15.1 para más información y edite el fichero `web.xml` y analice los comentarios que allí se incluyen.

Realizado en el código. Ver el fichero: `P1-ws/src/client/ssii2/controlador/ProcesaPago.java`

y añadido :

```

<context-param>
    <param-name>url_servidor</param-name>
    <param-value>http://10.1.1.2:8080/P1-ws-ws/VisaDAOWSService</param-value>
</context-param>

```

a `web.xml`

Ejercicio 11:

Realice una importación manual del WSDL del servicio sobre el directorio de clases local. Anote en la memoria qué comando ha sido necesario ejecutar en la línea de comandos, qué clases han sido generadas y por qué. Téngase en cuenta que el servicio debe estar previamente desplegado. comando:

```

wsimport -d build/client/WEB-INF/classes/ -p ssii2.visa http://10.1.1.2:8080/P1-ws-
ws/VisaDAOWSService?wsdl

```

Clases generadas

En la ruta `build/cliente/WEB_INF/classes` se han generado todas las clases que se implementan en el servidor, esto sirve para generar el proxy para el cliente dándole a conocer las clases y métodos disponibles con el objetivo de que pueda usar la funcionalidad del servidor.

Ejercicio número 12:

Complete el target generar-stubs definido en build.xml para que invoque a wsimport (utilizar la funcionalidad de ant exec para ejecutar aplicaciones). Téngase en cuenta que:

- El raíz del directorio de salida del compilador para la parte cliente ya está definido en build.properties como \${build.client}/WEB-INF/classes
- El paquete Java raíz (ssii2) ya está definido como \${paquete}
- La URL ya está definida como \${wsdl.url}

Complete el target generar-stubs definido en build.xml para que invoque a wsimport (utilizar la funcionalidad de ant exec para ejecutar aplicaciones). Téngase en cuenta que:

- El raíz del directorio de salida del compilador para la parte cliente ya está definido en build.properties como \${build.client}/WEB-INF/classes
- El paquete Java raíz (ssii2) ya está definido como \${paquete}
- La URL ya está definida como \${wsdl.url}

<!-- TODO - Implementar llamada wsimport →

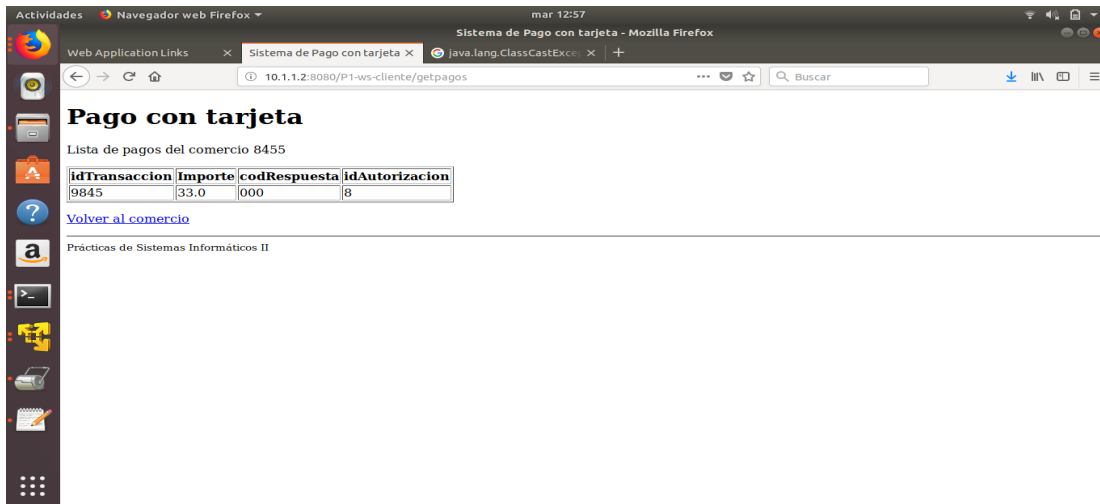
```
<exec executable = "${wsimport}">
  <arg line = "-d ${build.client}/WEB-INF/classes"/>
  <arg line = "-p ${paquete}.visa"/>
  <arg line = "${wsdl.url}"/>
</exec>
```

Cuestión número 13:

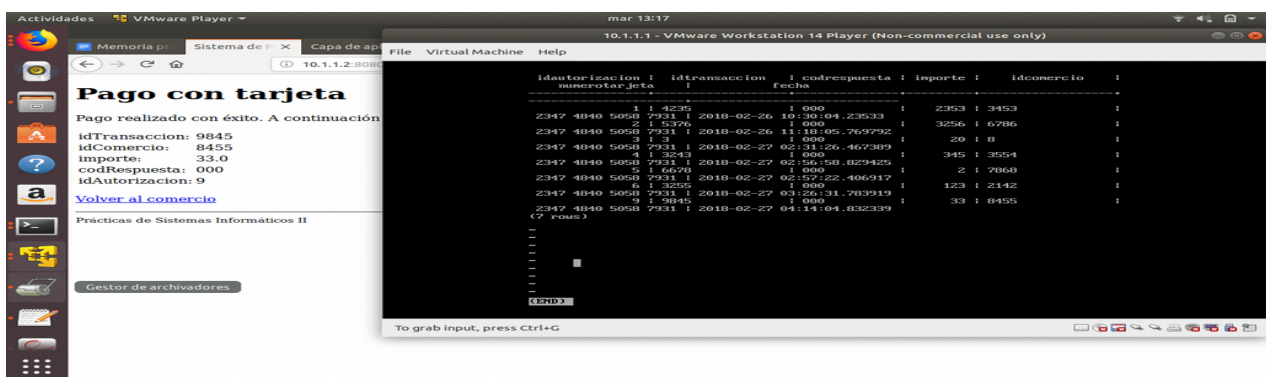
Realice un despliegue de la aplicación completo en dos nodos tal y como se explica en la Figura 8. Habrá que tener en cuenta que ahora en el fichero build.properties hay que especificar la dirección IP del servidor de aplicaciones donde se desplegará la parte del cliente de la aplicación y la dirección IP del servidor de aplicaciones donde se desplegará la parte del servidor. Las variables as.host.client y as.host.server deberán contener esta información.

Probar a realizar pagos correctos a través de la página testbd.jsp. Ejecutar las consultas SQL necesarias para comprobar que se realiza el pago. Anotar en la memoria práctica los resultados en forma de consulta SQL y resultados sobre la tabla de pagos





mediante la consulta `select * from pagos;` vemos que se ha realizado el pago

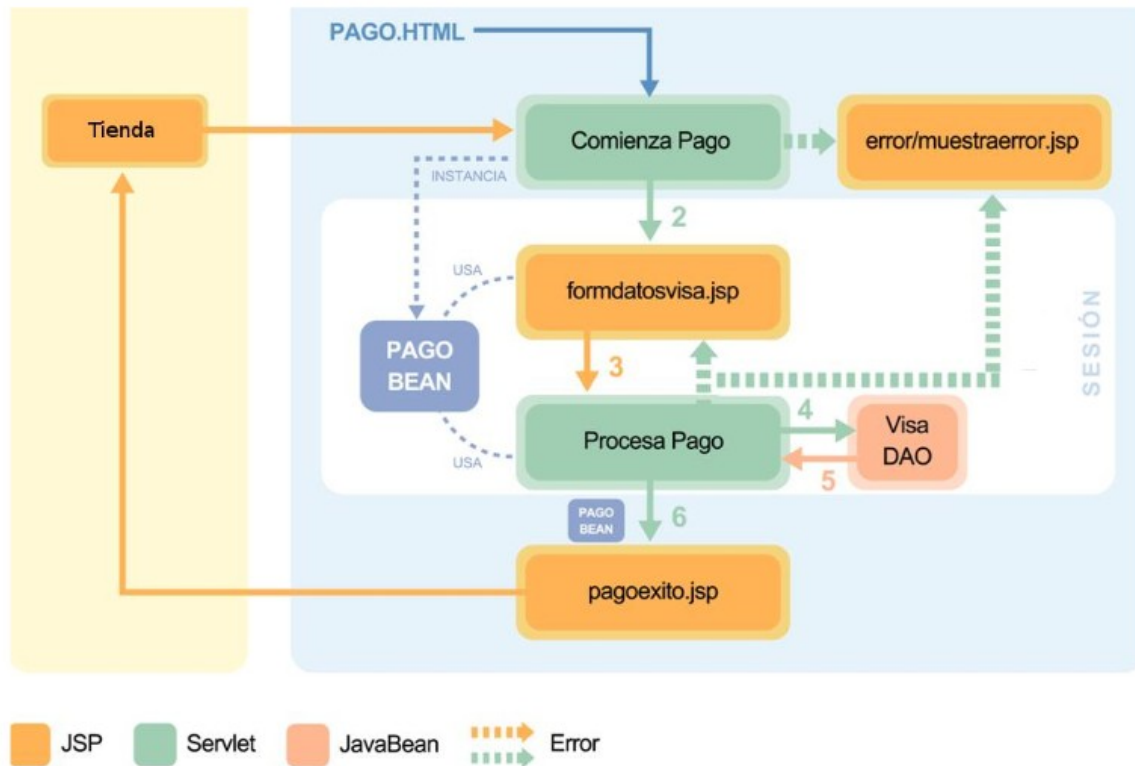


Cuestión número 1:

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

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



html:

pago.html -> comienza pago -> formatosvisa.jsp -> procesa pago -> pago exito -> tienda
-> (si hay un error) error/muestraerror.jsp
-> (error) error/muestraerror.jsp

Pago.html de el pasa al servlet de Comienza Pago que te dirige al servlet formatosvisa.jsp, después al servlet de Procesa Pago el cual detecta el error mediante formatosvisa.jsp, llevandote a error/muestraerror.jsp que muestra un mensaje de pago incorrecto.

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?
Comienza pago

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?

Copienza pago:

IdTransaccion , IdComercio ,Importe

Los guarda en la session asociada mediante:

```
request.getSession(true).setAttribute(ATTR_PAGO, pago);
```

formvisa.jsp:

Numero, Titular, FechaEmision, FechaCaducidad,CodigoVerificacion

Los cuales los envía a Procesa pago mediante un post.

Procesa pago:

Recoge toda la información guardandola en una instancia de Pago Bean recogiendo los campos de comienza pago de la session del sistema si esta existe, si no genera un pago nuevo:

```
if (sesion != null) {  
    pago = (PagoBean) sesion.getAttribute(ComienzaPago.ATTR_PAGO);  
}  
if (pago == null) {  
    pago = creaPago(request);  
    boolean isdebug = Boolean.valueOf(request.getParameter("debug"));  
    dao.setDebug(isdebug);  
    boolean isdirectConnection = Boolean.valueOf(request.getParameter("directConnection"));  
    dao.setDirectConnection(isdirectConnection);  
    boolean usePrepared = Boolean.valueOf(request.getParameter("usePrepared"));  
    dao.setPrepared(usePrepared);  
}
```

Los campos asociados a la tarjeta los recoge siempre mediante un post:

```
TarjetaBean tarjeta = creaTarjeta(request);
```

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?

Las diferencias es que cuando se accede por pago.html los datos se piden en dos tandas, usando "Comienza Pago" y "formvisa" como hemos comentado en el ejercicio anterior, en cambio testbd.jsp pide los datos a la vez.

Ambos, envían los datos a "Procesa Pago" por eso funciona igualmente.