Universidad Autónoma

Prácticas de Sistemas Informáticos II

Práctica 1

Memoria

Autores:
Adrián Fernández
Santiago González-Carvajal

Pareja 1
Grupo 2401

25 de febrero de 2019

Índice

1. Ejercicio 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.X.Y.Z: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.X.Y.Z: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.

Además de mover el archivo **postgresql-jdbc4.jar** a la carpeta **P1-base** (con el objetivo de que funcione el comando **ant todo**), también hemos modificado los dos archivos siguientes.

Código:

En el fichero **build.properties**, línea: 24.

```
1  # Propiedades de despliegue de aplicacion de Visa
2
3  nombre=P1
4  build=${basedir}/build
5  dist=${basedir}/dist
6  src=${basedir}/src
7  web=${basedir}/web
8  paquete=ssii2
9  war=${nombre}.war
10
11 asadmin=${as.home}/bin/asadmin
12 as.home=${env.J2EE_HOME}
13 as.lib=${as.home}/lib
14 as.user=admin
15 as.host=10.1.1.1
16
17 as.port=4848
18 as.passwordfile=${basedir}/passwordfile
19 as.target=server
```

En el fichero **postgresql.properties**, líneas: 8 y 13.

```
# Propiedades de la BD postgresql

# Parametros propios de postgresql
db.name=visa
db.user=alumnodb
db.password=****
db.port=5432
db.host=10.1.1.1

# Recursos y pools asociados
db.pool.name=VisaPool
db.jdbc.resource.name=jdbc/VisaDB
db.url=jdbc:postgresql://${db.host}:${db.port}/${db.name}
db.client.host=10.1.1.1
db.client.port=4848

db.delimiter=;
db.driver=org.postgresql.Driver
db.datasource=org.postgresql.ds.PGConnectionPoolDataSource
db.vendorname=SQL92

# Herramientas
db.createdb=/usr/bin/createdb
db.dropdb=/usr/bin/dropdb

# Scripts de creacion / borrado
db.create.src=./sql/create.sql
db.insert.src=./sql/insert.sql
db.delete.src=./sql/drop.sql
```

Capturas:

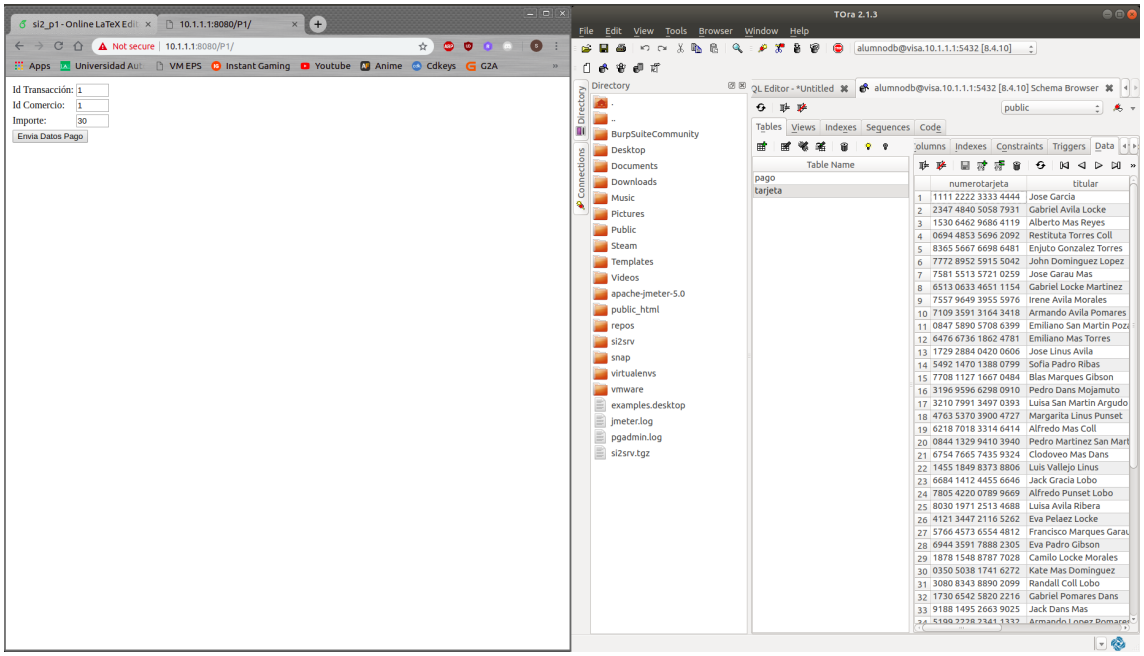


Figura 1: Accedemos al servicio de pago con tarjeta.

	numerotarjeta	titular	validadesde	validahasta	codigoverificacion
1	1111 2222 3333 4444	Jose Garcia	11/09	11/20	123

Figura 2: Usuario que vamos a utilizar en el pago.

Figura 3: Datos rellenados.

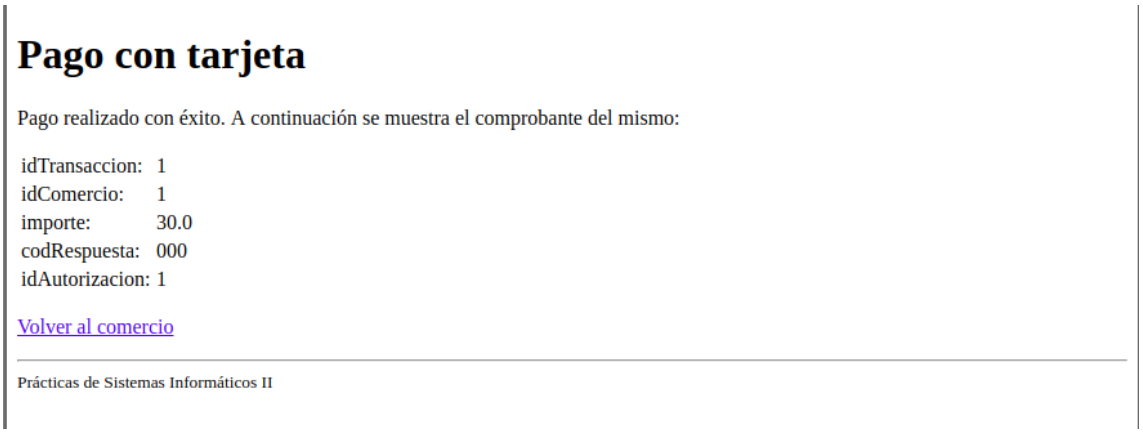


Figura 4: Pago realizado.

	idautorizacion	idtransaccion	codrespuesta	importe	idcomercio	numerotarjeta	fecha
1	1	1	000	30	1	1111 2222 3333 4444	15/02/19 08:23

Figura 5: Pago en la BD.

Figura 6: Consulta de pagos.

Figura 7: Pago consultado.

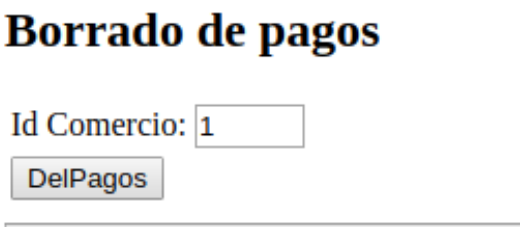


Figura 8: Borrado de pagos.

Figura 9: Borrado de pago correcto.

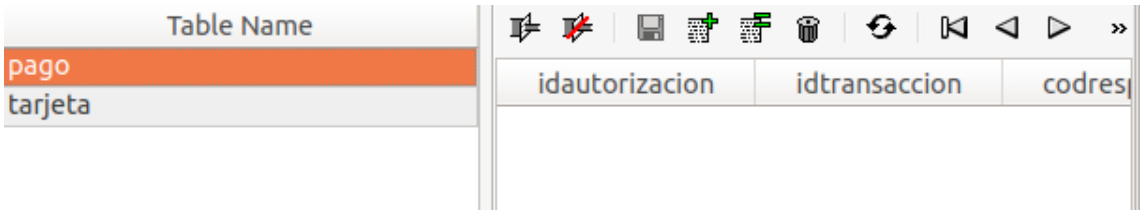


Figura 10: Borrado de pago en la BD.

2. Ejercicio 2

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 posgresql, 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.X.Y.Z: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, incluyendo capturas de pantalla.

A continuación se mostrarán unas capturas de pantalla evidenciando en correcto funcionamiento de la conexión directa a la BD **visa** al acceder a `http://10.1.1.1:8080/P1/testbd.jsp`.

Capturas:

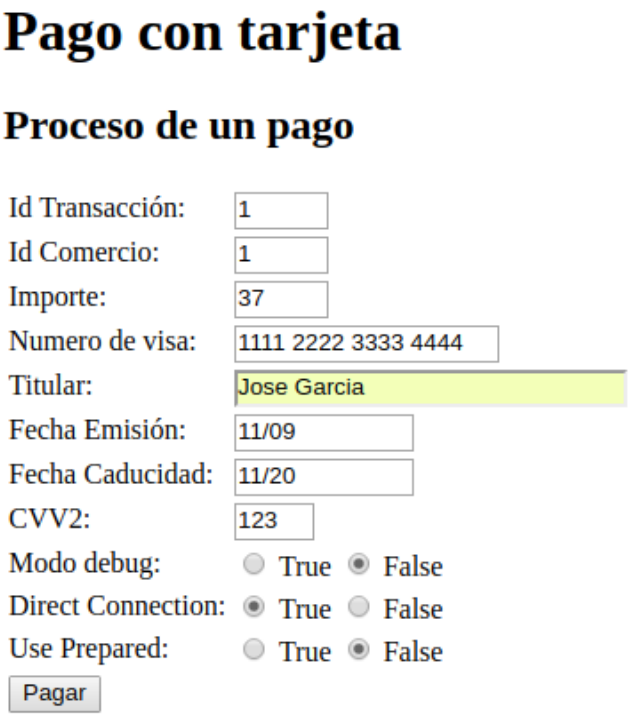


Figura 11: Formulario de pago con los datos necesarios para realizar una operación.

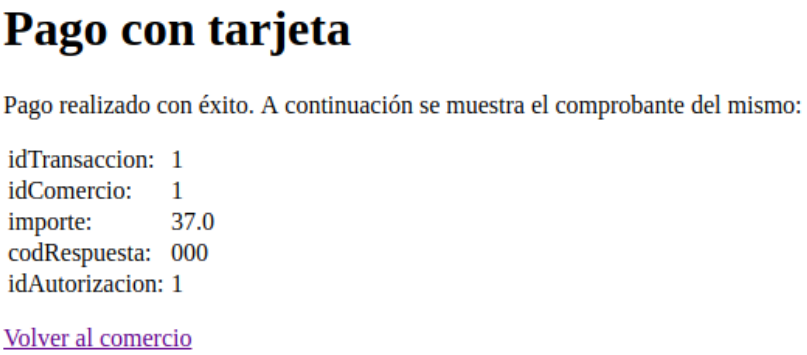


Figura 12: Respuesta del servidor.

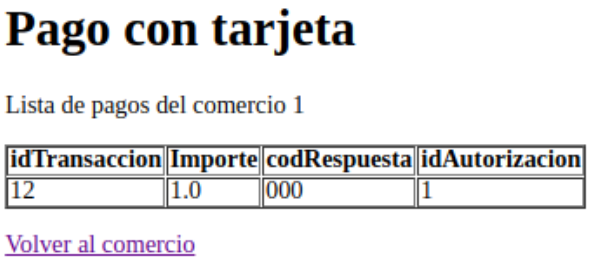


Figura 13: Respuesta del servidor a buscar los pagos con id de comercio 1.

Figura 14: Respuesta del servidor a eliminar los pagos con id de comercio 1.

3. 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.

Parámetros:

- 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

Captura:

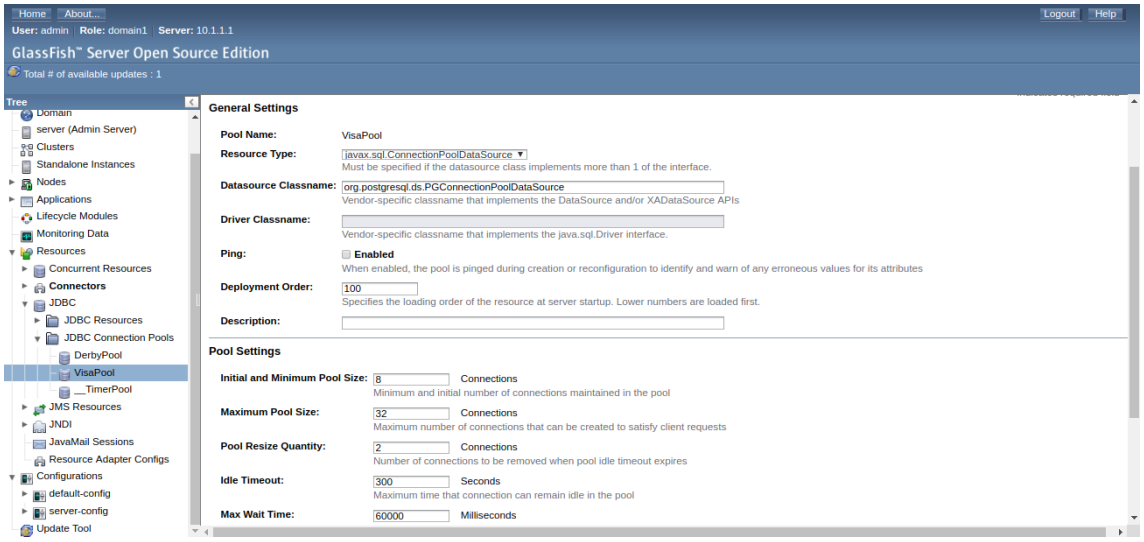


Figura 15: Interfaz de administrador mostrando las propiedades del pool de conexiones.

4. 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.

Código:

Código de validación de tarjetas en *P1-base/src/ssii2/visa/dao/visaDAO.java*:

Prepared statement:

```
private static final String SELECT_TARJETA_QRY =
                    "select * from tarjeta " +
                    "where numeroTarjeta=? " +
                    " and titular=? " +
                    " and validaDesde=? " +
                    " and validaHasta=? " +
                    " and codigoVerificacion=? ";
```

Ejecución de la consulta:

```
public boolean compruebaTarjeta(TarjetaBean tarjeta) {
    Connection con = null;
    Statement stmt = null;
    ResultSet rs = null;
    boolean ret = false;
    String qry = null;

    // TODO: Utilizar en funcion de isPrepared()
    PreparedStatement pstmt = null;

    try {

        // Crear una conexion u obtenerla del pool
        con = getConnection();

        // Se busca la ocurrencia de la tarjeta en la tabla

        /* TODO Usar prepared statement si
           isPrepared() == true */
        /*************************************************/
        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);

        } /*********************/

        /* Si hay siguiente registro, la tarjeta valido OK */
        ret = rs.next();

    } catch (Exception ee) {
```

```
45          errorLog(ee.toString());
46          ret = false;
47      } finally {
48          try {
49              if (rs != null) {
50                  rs.close(); rs = null;
51              }
52              if (stmt != null) {
53                  stmt.close(); stmt = null;
54              }
55              if (pstmt != null) {
56                  pstmt.close(); pstmt = null;
57              }
58              if (con != null) {
59                  closeConnection(con); con = null;
60              }
61          } catch (SQLException e) {
62          }
63      }
64
65      return ret;
66  }
```

Código de ejecución de pagos en *P1-base/src/ssii2/visa/dao/visaDAO.java*:

Prepared statements:

```
1  private static final String INSERT_PAGOS_QRY =
2                      "insert into pago(" +
3                      "idTransaccion,importe,idComercio,numeroTarjeta)" +
4                      " values (?,?,?,?)";
5
6  private static final String SELECT_PAGO_TRANSACCION_QRY =
7                      "select idAutorizacion, codRespuesta " +
8                      " from pago " +
9                      " where idTransaccion = ?" +
10                     " and idComercio = ?";
```

Ejecución de las consultas:

```
1  public synchronized boolean realizaPago(PagoBean pago) {
2      Connection con = null;
3      Statement stmt = null;
4      ResultSet rs = null;
5      boolean ret = false;
6      String codRespuesta = "999"; // En principio, denegado
7
8      // TODO: Utilizar en funcion de isPrepared()
9      PreparedStatement pstmt = null;
10
11     // Calcular pago.
12     // Comprobar id.transaccion - si no existe,
13     // es que la tarjeta no fue comprobada
14     if (pago.getIdTransaccion() == null) {
15         return false;
16     }
17
18     // Registrar el pago en la base de datos
19     try {
```

```
20
21          // Obtener conexion
22          con = getConnection();
23
24          // Insertar en la base de datos el pago
25
26          /* TODO Usar prepared statement si
27             isPrepared() == true */
28          /*************************************************/
29          if (isPrepared() == true) {
30             String insert  = INSERT_PAGOS_QRY;
31             errorLog(insert);
32             pstmt = con.prepareStatement(insert);
33             pstmt.setString(1, pago.getIdTransaccion());
34             pstmt.setDouble(2, pago.getImporte());
35             pstmt.setString(3, pago.getIdComercio());
36             pstmt.setString(4, pago.getTarjeta().getNumero());
37             ret = false;
38             if (!pstmt.execute()
39                     && pstmt.getUpdateCount() == 1) {
40               ret = true;
41             }
42
43          } else {
44          /*************************************************/
45          stmt = con.createStatement();
46          String insert = getQryInsertPago(pago);
47          errorLog(insert);
48          ret = false;
49          if (!stmt.execute(insert)
50                  && stmt.getUpdateCount() == 1) {
51              ret = true;
52                  }
53          }/***************/
54
55          // Obtener id.autorizacion
56          if (ret) {
57
58              /* TODO Permitir usar prepared statement si
59               * isPrepared() = true */
60              /*************************************************/
61              if (isPrepared() == true) {
62                  String select = SELECT_PAGO_TRANSACCION_QRY;
63                  errorLog(select);
64                  pstmt = con.prepareStatement(select);
65                  pstmt.setString(1, pago.getIdTransaccion());
66                  pstmt.setString(2, pago.getIdComercio());
67                  rs = pstmt.executeQuery();
68              } else {
69              /*************************************************/
70
71                  String select = getQryBuscaPagoTransaccion(pago);
72                  errorLog(select);
73                  rs = stmt.executeQuery(select);
74
75              }/*********************************/
76              if (rs.next()) {
77                  pago.setIdAutorizacion(String.valueOf(rs.getInt("
```

```
                    idAutorizacion")));
78                  pago.setCodRespuesta(rs.getString("codRespuesta"));
79              } else {
80                  ret = false;
81              }
82
83          }
84
85      } catch (Exception e) {
86          errorLog(e.toString());
87          ret = false;
88      } finally {
89          try {
90              if (rs != null) {
91                  rs.close(); rs = null;
92              }
93              if (stmt != null) {
94                  stmt.close(); stmt = null;
95              }
96              if (pstmt != null) {
97                  pstmt.close(); pstmt = null;
98              }
99              if (con != null) {
100                 closeConnection(con); con = null;
101             }
102         } catch (SQLException e) {
103         }
104     }
105
106     return ret;
107 }
```

5. Ejecicio 5

Ejercicio 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.

La información de la aplicación se genera como mensajes de nivel **SEVERE**.

Capturas:

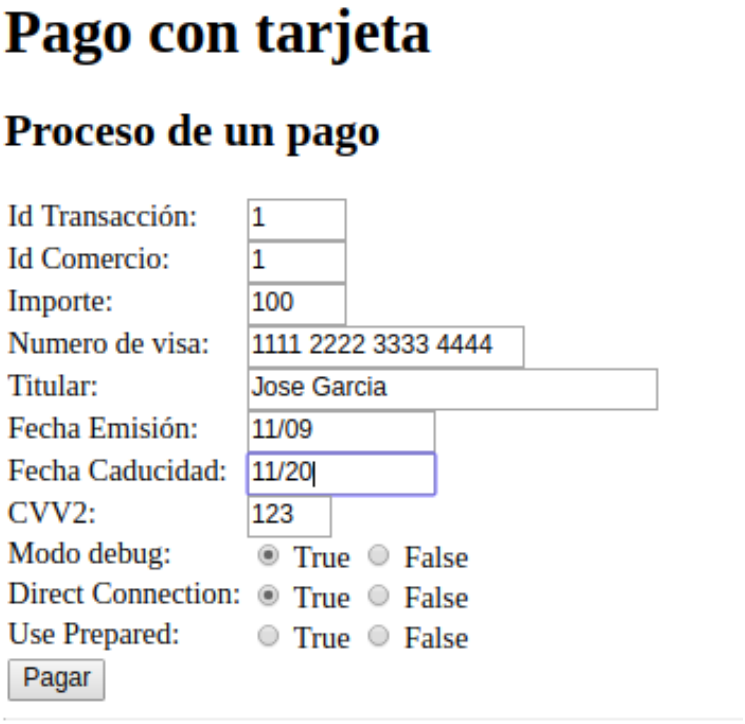
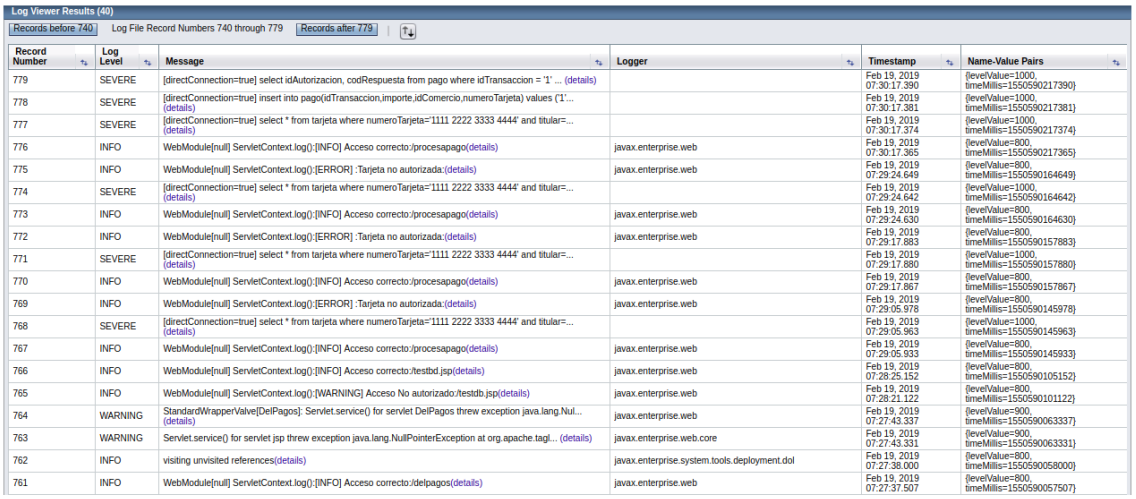


Figura 16: Formulario de pago con los datos necesarios para realizar una operación.



Log Viewer Results (40)

Records before 740 | Log File Record Numbers 740 through 779 | Records after 779

Record Number	Log Level	Message	Logger	Timestamp	Name-Value Pairs
779	SEVERE	[directConnection=true] select idAutorizacion, codRespuesta from pago where idTransaccion = '1' ... (details)		Feb 19, 2019 07:30:17.390	{levelValue=1000, timeMillis=1550590217390}
778	SEVERE	[directConnection=true] insert into pago(idTransaccion,importe,idComercio,numeroTarjeta) values ('1'... (details)		Feb 19, 2019 07:30:17.381	{levelValue=1000, timeMillis=1550590217381}
777	SEVERE	[directConnection=true] select * from tarjeta where numeroTarjeta='1111 2222 3333 4444' and titular=... (details)		Feb 19, 2019 07:30:17.374	{levelValue=1000, timeMillis=1550590217374}
776	INFO	WebModule[null] ServletContext.log():[INFO] Acceso correcto:/procesapago(details)	javax.enterprise.web	Feb 19, 2019 07:30:17.365	{levelValue=800, timeMillis=1550590217365}
775	INFO	WebModule[null] ServletContext.log():[ERROR] :Tarjeta no autorizada:(details)	javax.enterprise.web	Feb 19, 2019 07:29:24.649	{levelValue=800, timeMillis=1550590164649}
774	SEVERE	[directConnection=true] select * from tarjeta where numeroTarjeta='1111 2222 3333 4444' and titular=... (details)		Feb 19, 2019 07:29:24.642	{levelValue=1000, timeMillis=1550590164642}
773	INFO	WebModule[null] ServletContext.log():[INFO] Acceso correcto:/procesapago(details)	javax.enterprise.web	Feb 19, 2019 07:29:24.630	{levelValue=800, timeMillis=1550590164630}
772	INFO	WebModule[null] ServletContext.log():[ERROR] :Tarjeta no autorizada:(details)	javax.enterprise.web	Feb 19, 2019 07:29:17.883	{levelValue=800, timeMillis=1550590157883}
771	SEVERE	[directConnection=true] select * from tarjeta where numeroTarjeta='1111 2222 3333 4444' and titular=... (details)		Feb 19, 2019 07:29:17.880	{levelValue=1000, timeMillis=1550590157880}
770	INFO	WebModule[null] ServletContext.log():[INFO] Acceso correcto:/procesapago(details)	javax.enterprise.web	Feb 19, 2019 07:29:17.867	{levelValue=800, timeMillis=1550590157867}
769	INFO	WebModule[null] ServletContext.log():[ERROR] :Tarjeta no autorizada:(details)	javax.enterprise.web	Feb 19, 2019 07:29:05.978	{levelValue=800, timeMillis=1550590145978}
768	SEVERE	[directConnection=true] select * from tarjeta where numeroTarjeta='1111 2222 3333 4444' and titular=... (details)		Feb 19, 2019 07:29:05.963	{levelValue=1000, timeMillis=1550590145963}
767	INFO	WebModule[null] ServletContext.log():[INFO] Acceso correcto:/procesapago(details)	javax.enterprise.web	Feb 19, 2019 07:29:05.933	{levelValue=800, timeMillis=1550590145933}
766	INFO	WebModule[null] ServletContext.log():[INFO] Acceso correcto:/testbd.jsp(details)	javax.enterprise.web	Feb 19, 2019 07:28:25.152	{levelValue=800, timeMillis=1550590105152}
765	INFO	WebModule[null] ServletContext.log():[WARNING] Acceso No autorizado:/testdb.jsp(details)	javax.enterprise.web	Feb 19, 2019 07:28:21.122	{levelValue=800, timeMillis=1550590101122}
764	WARNING	StandardWrapperValve[DelPagos]: Servlet.service() for servlet DelPagos threw exception java.lang.Nul... (details)	javax.enterprise.web	Feb 19, 2019 07:27:43.337	{levelValue=900, timeMillis=1550590063337}
763	WARNING	Servlet.service() for servlet jsp threw exception java.lang.NullPointerException at org.apache.tagl... (details)	javax.enterprise.web.core	Feb 19, 2019 07:27:43.331	{levelValue=900, timeMillis=1550590063331}
762	INFO	visiting unvisited references(details)	javax.enterprise.system.tools.deployment.dol	Feb 19, 2019 07:27:38.000	{levelValue=800, timeMillis=1550590058000}
761	INFO	WebModule[null] ServletContext.log():[INFO] Acceso correcto:/delpagos(details)	javax.enterprise.web	Feb 19, 2019 07:27:37.507	{levelValue=800, timeMillis=1550590057507}

Figura 17: Log del servidor de aplicaciones.

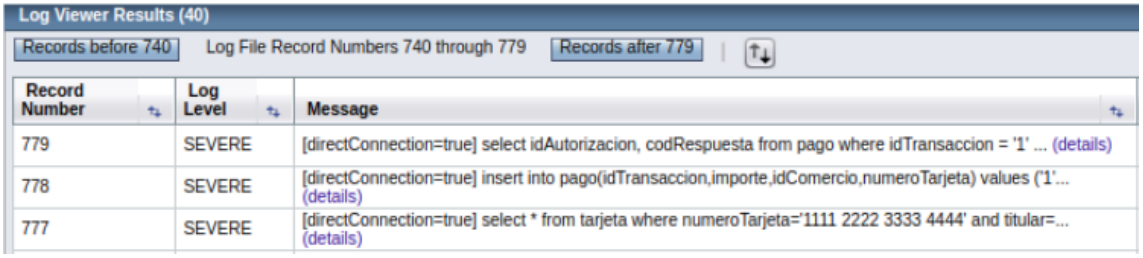


Figura 18: Ampliación de los 3 primeros mensajes.

6. Ejecicio 6

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

Incluye en la memoria cada fragmento de código donde se han ido añadiendo las modificaciones requeridas.

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?

Cuestión:

Porque ahora el servicio está en una máquina distinta a la del cliente (hemos dividido la aplicación en dos máquinas virtuales), y por ello no basta con modificar el pago que se recibe, ya que sería un paso por referencia que no puede modificarse desde otra máquina.

Código:

Modificaciones de código (ponemos //MODIFICADA en las líneas modificadas en este ejercicio):

```
1     /**
2      *
3      * Comprobacion de la tarjeta
4      * @param tarjeta Objeto con toda la informacion de la tarjeta
5      * @return true si la comprobacion contra las tarjetas contenidas
           en
6      *         en la tabla TARJETA fue satisfactoria, false en caso
           contrario      */
7     @WebMethod(operationName = "compruebaTarjeta") //MODIFICADA
8     public boolean compruebaTarjeta(@WebParam(name="tarjeta")
          TarjetaBean tarjeta) {//MODIFICADA
9         Connection con = null;
10        Statement stmt = null;
11        ResultSet rs = null;
12        boolean ret = false;
13        String qry = null;
14
15        // TODO: Utilizar en funcion de isPrepared()
16        PreparedStatement pstmt = null;
17
18        try {
19
20            // Crear una conexion u obtenerla del pool
21            con = getConnection();
22
23            // Se busca la ocurrencia de la tarjeta en la tabla
24
25            /* TODO Usar prepared statement si
26               isPrepared() == true */
27            /*************************************************/
28            if (isPrepared() == true) {
29               String select  = SELECT_TARJETA_QRY;
30               errorLog(select);
31               pstmt = con.prepareStatement(select);
32               pstmt.setString(1, tarjeta.getNumero());
33               pstmt.setString(2, tarjeta.getTitular());
34               pstmt.setString(3, tarjeta.getFechaEmision());
35               pstmt.setString(4, tarjeta.getFechaCaducidad());
36               pstmt.setString(5, tarjeta.getCodigoVerificacion());
37               rs = pstmt.executeQuery();
38
39            } else {
40            /*************************************************/
41            stmt = con.createStatement();
42            qry = getQryCompruebaTarjeta(tarjeta);
43            errorLog(qry);
44            rs = stmt.executeQuery(qry);
45
46            } /*********************/
47
48            /* Si hay siguiente registro, la tarjeta valido OK */
49            ret = rs.next();
50
51        } catch (Exception ee) {
```

```
52              errorLog(ee.toString());
53              ret = false;
54          } finally {
55              try {
56                  if (rs != null) {
57                      rs.close(); rs = null;
58                  }
59                  if (stmt != null) {
60                      stmt.close(); stmt = null;
61                  }
62                  if (pstmt != null) {
63                      pstmt.close(); pstmt = null;
64                  }
65                  if (con != null) {
66                      closeConnection(con); con = null;
67                  }
68              } catch (SQLException e) {
69              }
70          }
71
72          return ret;
73
74      }
75
76      /**
77       * Realiza el pago
78       * @param pago
79       * @return
80       */
81      @WebMethod(operationName = "realizaPago") //MODIFICADA
82      public synchronized PagoBean realizaPago(@WebParam(name="pago")
            PagoBean pago) {//MODIFICADA
83          Connection con = null;
84          Statement stmt = null;
85          PagoBean returned_pago = null;
86          ResultSet rs = null;
87          boolean ret = false;
88          String codRespuesta = "999"; // En principio, denegado
89
90          // TODO: Utilizar en funcion de isPrepared()
91          PreparedStatement pstmt = null;
92
93          // Calcular pago.
94          // Comprobar id.transaccion - si no existe,
95          // es que la tarjeta no fue comprobada
96          if (pago.getIdTransaccion() == null) {
97              return null;//MODIFICADA
98          }
99
100         // Registrar el pago en la base de datos
101         try {
102
103             // Obtener conexion
104             con = getConnection();
105
106             // Insertar en la base de datos el pago
107
108             /* TODO Usar prepared statement si
```

```
109                 isPrepared() == true */
110             /*********************************************/
111             if (isPrepared() == true) {
112                 String insert  = INSERT_PAGOS_QRY;
113                 errorLog(insert);
114                 pstmt = con.prepareStatement(insert);
115                 pstmt.setString(1, pago.getIdTransaccion());
116                 pstmt.setDouble(2, pago.getImporte());
117                 pstmt.setString(3, pago.getIdComercio());
118                 pstmt.setString(4, pago.getTarjeta().getNumero());
119                 ret = false;
120                 if (!pstmt.execute()
121                         && pstmt.getUpdateCount() == 1) {
122                   ret = true;
123                 }
124
125             } else {
126             /*********************************************/
127             stmt = con.createStatement();
128             String insert = getQryInsertPago(pago);
129             errorLog(insert);
130             ret = false;
131             if (!stmt.execute(insert)
132                     && stmt.getUpdateCount() == 1) {
133                 ret = true;
134                         }
135             }/***************/
136
137             // Obtener id.autorizacion
138             if (ret) {
139
140                 /* TODO Permitir usar prepared statement si
141                  * isPrepared() = true */
142                 /*********************************************/
143                 if (isPrepared() == true) {
144                     String select = SELECT_PAGO_TRANSACCION_QRY;
145                     errorLog(select);
146                     pstmt = con.prepareStatement(select);
147                     pstmt.setString(1, pago.getIdTransaccion());
148                     pstmt.setString(2, pago.getIdComercio());
149                     rs = pstmt.executeQuery();
150                 } else {
151                 /*********************************************/
152
153                     String select = getQryBuscaPagoTransaccion(pago);
154                     errorLog(select);
155                     rs = stmt.executeQuery(select);
156
157                 }/***********************************/
158                 if (rs.next()) {
159                     pago.setIdAutorizacion(String.valueOf(rs.getInt("
                            idAutorizacion")));
160                     pago.setCodRespuesta(rs.getString("codRespuesta"));
161                 } else {
162                     ret = false;
163                 }
164
165             }
```

```
166
167         } catch (Exception e) {
168             errorLog(e.toString());
169             ret = false;
170         } finally {
171             try {
172                 if (rs != null) {
173                     rs.close(); rs = null;
174                 }
175                 if (stmt != null) {
176                     stmt.close(); stmt = null;
177                 }
178                 if (pstmt != null) {
179                     pstmt.close(); pstmt = null;
180                 }
181                 if (con != null) {
182                     closeConnection(con); con = null;
183                 }
184             } catch (SQLException e) {
185             }
186         }
187         //MODIFICADO HASTA FIN
188         if(ret == false){
189           return null;
190         } else{
191           returned_pago = new PagoBean();
192           returned_pago.setIdTransaccion(returned_pago.getIdTransaccion
                ());
193           returned_pago.setIdComercio(pago.getIdComercio());
194           returned_pago.setImporte(pago.getImporte());
195           returned_pago.setRutaRetorno(pago.getRutaRetorno());
196           returned_pago.setTarjeta(pago.getTarjeta());
197           returned_pago.setIdAutorizacion(pago.getIdAutorizacion());
198           returned_pago.setCodRespuesta(pago.getCodRespuesta());
199
200           return returned_pago;
201         }
202         //FIN
203     }
204
205     /**
206      * TODO: Metodos isPrepared() y setPrepared()
207      */
208     /*****************************************************/
209     @WebMethod(operationName = "isPrepared")//MODIFICADA
210     public boolean isPrepared() {
211         return prepared;
212     }
213
214     @WebMethod(operationName = "setPrepared")//MODIFICADA
215     public void setPrepared(@WebParam(name="prepared") boolean prepared
            ) {//MODIFICADA
216         this.prepared = prepared;
217     }
218     /*****************************************************/
219
220     /**
221      * @return the debug
```

```
222         */
223        @WebMethod(operationName = "isDebug")//MODIFICADA
224        public boolean isDebug() {
225            return debug;
226        }
227
228        /**
229         * @param debug the debug to set
230         */
231        @WebMethod(operationName = "setDebug")//MODIFICADA
232        public void setDebug(@WebParam(name="debug") boolean debug) {//
               MODIFICADA
233            this.debug = debug;
234        }
235
236        /**
237         * @param debug the debug to set
238         */
239        @WebMethod(exclude=true)//MODIFICADA
240        public void setDebug(String debug) {
241            this.debug = (debug.equals("true"));
242        }
243
244        /**
245         * @return the debug
246         */
247        @Override//MODIFICADA
248        @WebMethod(operationName = "isDirectConnection")//MODIFICADA
249        public boolean isDirectConnection() {
250            return super.isDirectConnection();
251        }
252
253        /**
254         * @param debug the debug to set
255         */
256        @Override//MODIFICADA
257        @WebMethod(operationName = "setDirectConnection")//MODIFICADA
258        public void setDirectConnection(@WebParam(name = "directConnection"
               ) boolean directConnection) {//MODIFICADA
259          super.setDirectConnection(directConnection);
260        }
```

7. Ejecicio 7

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 (el documento aparece después de las preguntas):

- ¿En qué fichero están definidos los tipos de datos intercambiados con el web-service?
- ¿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?
- ¿Qué etiqueta describe los mensajes intercambiados en la invocación de los métodos del webservice?
- ¿En qué etiqueta se especifica el protocolo de comunicación con el webservice?
- ¿En qué etiqueta se especifica la URL a la que se deberá conectar un cliente para acceder al webservice?

Cuestiones:

Cuestión 1. En el fichero WSDL.

Cuestión 2. Se usan los datos que importa en las líneas 3, 4 y 5 (son de la forma xs:tipodedato, por ejemplo xs:String).

Cuestión 3. Los tipos de datos que se definen son todos los que aparecen con el prefijo tns que dirigen a la URL del servidor.

Cuestión 4. La etiqueta operation.

Cuestión 5. La etiqueta message, (también aparece en output message=).

Cuestión 6. En la etiqueta soap:binding transport=.

Cuestión 7. En la etiqueta service, concretamente en soap:address.

Código:

Fichero **WSDL** pedido:

```
<?xml version='1.0' encoding='UTF-8'?><!-- Published by JAX-WS RI (http://jax-ws.java.net). RI's version is Metro/2.3.2-b608 (trunk-7979; 2015-01-21T12:50:19+0000) JAXWS-RI/2.2.11-b150120.1832 JAXWS-API/2.2.12 JAXB-RI/2.2.12-b141219.1637 JAXB-API/2.2.13-b141020.1521 svn-revision#unknown. --><!-- Generated by JAX-WS RI (http://jax-ws.java.net). RI's version is Metro/2.3.2-b608 (trunk-7979; 2015-01-21T12:50:19+0000) JAXWS-RI/2.2.11-b150120.1832 JAXWS-API/2.2.12 JAXB-RI/2.2.12-b141219.1637 JAXB-API/2.2.13-b141020.1521 svn-revision#unknown. --><definitions xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" xmlns:wsp="http://www.w3.org/ns/ws-policy" xmlns:wsp1_2="http://schemas.xmlsoap.org/ws/2004/09/policy" xmlns:wsam="http://www.w3.org/2007/05/addressing/metadata" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tns="http://dao.visa.ssii2/" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="http://schemas.xmlsoap.org/wsdl/" targetNamespace="http://dao.visa.ssii2/" name="VisaDAOWSService">
<types>
<xsd:schema>
<xsd:import namespace="http://dao.visa.ssii2/" schemaLocation="http://10.1.1.1:8080/P1-ws-ws/VisaDAOWSService?xsd=1"/>
</xsd:schema>
</types>
<message name="isDebug">
<part name="parameters" element="tns:isDebug"/>
</message>
<message name="isDebugResponse">
<part name="parameters" element="tns:isDebugResponse"/>
</message>
<message name="setDebug">
<part name="parameters" element="tns:setDebug"/>
</message>
<message name="setDebugResponse">
<part name="parameters" element="tns:setDebugResponse"/>
</message>
<message name="compruebaTarjeta">
<part name="parameters" element="tns:compruebaTarjeta"/>
</message>
<message name="compruebaTarjetaResponse">
<part name="parameters" element="tns:compruebaTarjetaResponse"/>
</message>
<message name="realizaPago">
<part name="parameters" element="tns:realizaPago"/>
</message>
<message name="realizaPagoResponse">
<part name="parameters" element="tns:realizaPagoResponse"/>
</message>
<message name="getPagos">
<part name="parameters" element="tns:getPagos"/>
</message>
<message name="getPagosResponse">
<part name="parameters" element="tns:getPagosResponse"/>
</message>
<message name="delPagos">
<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="isDirectConnection">
56 <part name="parameters" element="tns:isDirectConnection"/>
57 </message>
58 <message name="isDirectConnectionResponse">
59 <part name="parameters" element="tns:isDirectConnectionResponse"/>
60 </message>
61 <message name="setDirectConnection">
62 <part name="parameters" element="tns:setDirectConnection"/>
63 </message>
64 <message name="setDirectConnectionResponse">
65 <part name="parameters" element="tns:setDirectConnectionResponse"/>
66 </message>
67 <message name="errorLog">
68 <part name="parameters" element="tns:errorLog"/>
69 </message>
70 <message name="errorLogResponse">
71 <part name="parameters" element="tns:errorLogResponse"/>
72 </message>
73 <portType name="VisaDAOWS">
74 <operation name="isDebug">
75 <input wsam:Action="http://dao.visa.ssii2/VisaDAOWS/isDebugRequest"
      message="tns:isDebug"/>
76 <output wsam:Action="http://dao.visa.ssii2/VisaDAOWS/isDebugResponse"
      message="tns:isDebugResponse"/>
77 </operation>
78 <operation name="setDebug">
79 <input wsam:Action="http://dao.visa.ssii2/VisaDAOWS/setDebugRequest"
      message="tns:setDebug"/>
80 <output wsam:Action="http://dao.visa.ssii2/VisaDAOWS/setDebugResponse"
      message="tns:setDebugResponse"/>
81 </operation>
82 <operation name="compruebaTarjeta">
83 <input wsam:Action="http://dao.visa.ssii2/VisaDAOWS/
      compruebaTarjetaRequest" message="tns:compruebaTarjeta"/>
84 <output wsam:Action="http://dao.visa.ssii2/VisaDAOWS/
      compruebaTarjetaResponse" message="tns:compruebaTarjetaResponse"/>
85 </operation>
86 <operation name="realizaPago">
87 <input wsam:Action="http://dao.visa.ssii2/VisaDAOWS/realizaPagoRequest"
       message="tns:realizaPago"/>
88 <output wsam:Action="http://dao.visa.ssii2/VisaDAOWS/
      realizaPagoResponse" message="tns:realizaPagoResponse"/>
```

```
89 </operation>
90 <operation name="getPagos">
91 <input wsam:Action="http://dao.visa.ssii2/VisaDAOWS/getPagosRequest"
       message="tns:getPagos"/>
92 <output wsam:Action="http://dao.visa.ssii2/VisaDAOWS/getPagosResponse"
       message="tns:getPagosResponse"/>
93 </operation>
94 <operation name="delPagos">
95 <input wsam:Action="http://dao.visa.ssii2/VisaDAOWS/delPagosRequest"
       message="tns:delPagos"/>
96 <output wsam:Action="http://dao.visa.ssii2/VisaDAOWS/delPagosResponse"
       message="tns:delPagosResponse"/>
97 </operation>
98 <operation name="isPrepared">
99 <input wsam:Action="http://dao.visa.ssii2/VisaDAOWS/isPreparedRequest"
       message="tns:isPrepared"/>
100 <output wsam:Action="http://dao.visa.ssii2/VisaDAOWS/isPreparedResponse
       " message="tns:isPreparedResponse"/>
101 </operation>
102 <operation name="setPrepared">
103 <input wsam:Action="http://dao.visa.ssii2/VisaDAOWS/setPreparedRequest"
        message="tns:setPrepared"/>
104 <output wsam:Action="http://dao.visa.ssii2/VisaDAOWS/
       setPreparedResponse" message="tns:setPreparedResponse"/>
105 </operation>
106 <operation name="isDirectConnection">
107 <input wsam:Action="http://dao.visa.ssii2/VisaDAOWS/
       isDirectConnectionRequest" message="tns:isDirectConnection"/>
108 <output wsam:Action="http://dao.visa.ssii2/VisaDAOWS/
       isDirectConnectionResponse" message="tns:isDirectConnectionResponse
       "/>
109 </operation>
110 <operation name="setDirectConnection">
111 <input wsam:Action="http://dao.visa.ssii2/VisaDAOWS/
       setDirectConnectionRequest" message="tns:setDirectConnection"/>
112 <output wsam:Action="http://dao.visa.ssii2/VisaDAOWS/
       setDirectConnectionResponse" message="
       tns:setDirectConnectionResponse"/>
113 </operation>
114 <operation name="errorLog">
115 <input wsam:Action="http://dao.visa.ssii2/VisaDAOWS/errorLogRequest"
       message="tns:errorLog"/>
116 <output wsam:Action="http://dao.visa.ssii2/VisaDAOWS/errorLogResponse"
       message="tns:errorLogResponse"/>
117 </operation>
118 </portType>
119 <binding name="VisaDAOWSPortBinding" type="tns:VisaDAOWS">
120 <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="
       document"/>
121 <operation name="isDebug">
122 <soap:operation soapAction=""/>
123 <input>
124 <soap:body use="literal"/>
125 </input>
126 <output>
127 <soap:body use="literal"/>
128 </output>
129 </operation>
```

```
<operation name="setDebug">
<soap:operation soapAction=""/>
<input>
<soap:body use="literal"/>
</input>
<output>
<soap:body use="literal"/>
</output>
</operation>
<operation name="compruebaTarjeta">
<soap:operation soapAction=""/>
<input>
<soap:body use="literal"/>
</input>
<output>
<soap:body use="literal"/>
</output>
</operation>
<operation name="realizaPago">
<soap:operation soapAction=""/>
<input>
<soap:body use="literal"/>
</input>
<output>
<soap:body use="literal"/>
</output>
</operation>
<operation name="getPagos">
<soap:operation soapAction=""/>
<input>
<soap:body use="literal"/>
</input>
<output>
<soap:body use="literal"/>
</output>
</operation>
<operation name="delPagos">
<soap:operation soapAction=""/>
<input>
<soap:body use="literal"/>
</input>
<output>
<soap:body use="literal"/>
</output>
</operation>
<operation name="isPrepared">
<soap:operation soapAction=""/>
<input>
<soap:body use="literal"/>
</input>
<output>
<soap:body use="literal"/>
</output>
</operation>
<operation name="setPrepared">
<soap:operation soapAction=""/>
<input>
<soap:body use="literal"/>
```

```
188 </input>
189 <output>
190 <soap:body use="literal"/>
191 </output>
192 </operation>
193 <operation name="isDirectConnection">
194 <soap:operation soapAction=""/>
195 <input>
196 <soap:body use="literal"/>
197 </input>
198 <output>
199 <soap:body use="literal"/>
200 </output>
201 </operation>
202 <operation name="setDirectConnection">
203 <soap:operation soapAction=""/>
204 <input>
205 <soap:body use="literal"/>
206 </input>
207 <output>
208 <soap:body use="literal"/>
209 </output>
210 </operation>
211 <operation name="errorLog">
212 <soap:operation soapAction=""/>
213 <input>
214 <soap:body use="literal"/>
215 </input>
216 <output>
217 <soap:body use="literal"/>
218 </output>
219 </operation>
220 </binding>
221 <service name="VisaDAOWSService">
222 <port name="VisaDAOWSPort" binding="tns:VisaDAOWSPortBinding">
223 <soap:address location="http://10.1.1.1:8080/P1-ws-ws/VisaDAOWSService"
      />
224 </port>
225 </service>
226 </definitions>
```

8. Ejecicio 8

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.

Incluye en la memoria una captura con dichas modificaciones.

Código:

```
1     @Override
2     protected void processRequest(HttpServletRequest request,
          HttpServletResponse response)
3     throws ServletException, IOException {
4
5         TarjetaBean tarjeta = creaTarjeta(request);
6         ValidadorTarjeta val = new ValidadorTarjeta();
7         PagoBean pago = null;
8
9         // printAddresses(request,response);
10        if (! val.esValida(tarjeta)) {
11            request.setAttribute(val.getErrorName(), val.getErrorVisa()
                  );
12            reenvia("/formdatosvisa.jsp", request, response);
13            return;
14        }
15
16        VisaDAOWSService service = new VisaDAOWSService();
17        VisaDAOWS dao = service.getVisaDAOWSPort();
18                //VisaDAO dao = new VisaDAO();
19                try{
20                  HttpSession sesion = request.getSession(false);
21                  if (sesion != null) {
22                          pago = (PagoBean) sesion.getAttribute(
                              ComienzaPago.ATTR_PAGO);
23                  }
24                  if (pago == null) {
25                          pago = creaPago(request);
26                          boolean isdebug = Boolean.valueOf(request.
                              getParameter("debug"));
27                          dao.setDebug(isdebug);
28                          boolean isdirectConnection = Boolean.valueOf(
                              request.getParameter("directConnection"))
                              ;
29                          dao.setDirectConnection(isdirectConnection);
30                          boolean usePrepared = Boolean.valueOf(request
                              .getParameter("usePrepared"));
31                          dao.setPrepared(usePrepared);
32            }
33
34            // Almacenamos la tarjeta en el pago
35            pago.setTarjeta(tarjeta);
36
37            if (! dao.compruebaTarjeta(tarjeta)) {
38                enviaError(new Exception("Tarjeta no autorizada:"),
                      request, response);
39                return;
40            }
41        pago = dao.realizaPago(pago);
42              if (!pago) {
43                enviaError(new Exception("Pago incorrecto"), request,
                      response);
44                return;
45            }
46
47            request.setAttribute(ComienzaPago.ATTR_PAGO, pago);
```

```
48              if (sesion != null) sesion.invalidate();
49              reenvia("/pagoexito.jsp", request, response);
50              return;
51          }catch(Exception e){
52            enviaError(new Exception("Pago incorrecto"), request, response)
                  ;
53          }
54      }
```

9. Ejecicio 9

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.

Código:

```
1  <context-param>
2      <param-name>webmaster</param-name>
3      <param-value>http://10.1.1.1:8080/P1-ws-ws/VisaDAOWSService</param-
           value>
4  </context-param>
```

10. Ejecicio 10

Ejercicio 10. Siguiendo el patrón de los cambios anteriores, adaptar las siguientes clases cliente para que toda la funcionalidad de la página de pruebas testbd.jsp se realice a través del servicio web. Esto afecta al menos a los siguientes recursos:

- Servlet DelPagos.java: la operación dao.delPagos() debe implementarse en el servicio web.
- Servlet GetPagos.java: la operación dao.getPagos() debe implementarse en el servicio web.

Tenga en cuenta que no todos los tipos de datos son compatibles con JAXB (especifica como codificar clases java como documentos XML), por lo que es posible que tenga que modificar el valor de retorno de alguno de estos métodos. Los apéndices contienen más información. Más específicamente, se tiene que modificar la declaración actual del método getPagos(), que devuelve un PagoBean[], por:

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

Hay que tener en cuenta que la página listapagos.jsp espera recibir un array del tipo PagoBean[]. Por ello, es conveniente, una vez obtenida la respuesta, convertir el ArrayList a un array de tipo PagoBean[] utilizando el método toArray() de la clase ArrayList.

Incluye en la memoria una captura con las adaptaciones realizadas.

Código:

Cambios en VisaDAOWS.java:

```
1  /**
2       * Buscar los pagos asociados a un comercio
3       * @param idComercio
4       * @return
5       */
6      @WebMethod(operationName = "getPagos")
7      public ArrayList<PagoBean> getPagos(@WebParam(name="idComercio")
           String idComercio) {
8
9          PreparedStatement pstmt = null;
10         Connection pcon = null;
11         ResultSet rs = null;
12         PagoBean[] ret = null;
13         ArrayList<PagoBean> pagos = null;
14         String qry = null;
15
16         try {
17
18             // Crear una conexion u obtenerla del pool
19             pcon = getConnection();
20             qry = SELECT_PAGOS_QRY;
```

```
21              errorLog(qry + "[idComercio=" + idComercio + "]");
22
23              // La preparacion del statement
24              // es automaticamente tomada de un pool en caso
25              // de que ya haya sido preparada con anterioridad
26              pstmt = pcon.prepareStatement(qry);
27
28              pstmt.setString(1, idComercio);
29              rs = pstmt.executeQuery();
30
31              pagos = new ArrayList<PagoBean>();
32
33              while (rs.next()) {
34                  TarjetaBean t = new TarjetaBean();
35                  PagoBean p = new PagoBean();
36                  p.setIdTransaccion(rs.getString("idTransaccion"));
37                  p.setIdComercio(rs.getString("idComercio"));
38                  p.setImporte(rs.getFloat("importe"));
39                  t.setNumero(rs.getString("numeroTarjeta"));
40                  p.setTarjeta(t);
41                  p.setCodRespuesta(rs.getString("codRespuesta"));
42                  p.setIdAutorizacion(String.valueOf(rs.getInt("
                        idAutorizacion")));
43
44                  pagos.add(p);
45              }
46
47              ret = new PagoBean[pagos.size()];
48              ret = pagos.toArray(ret);
49
50              // Cerramos / devolvemos la conexion al pool
51              pcon.close();
52
53          } catch (Exception e) {
54              errorLog(e.toString());
55
56          } finally {
57              try {
58                  if (rs != null) {
59                      rs.close(); rs = null;
60                  }
61                  if (pstmt != null) {
62                      pstmt.close(); pstmt = null;
63                  }
64                  if (pcon != null) {
65                      closeConnection(pcon); pcon = null;
66                  }
67              } catch (SQLException e) {
68              }
69          }
70
71          ArrayList<PagoBean> retorn = new ArrayList<PagoBean>(Arrays.
                asList(ret));
72
73          return retorn;
74      }
75
76      // Borrar los pagos asociados a un comercio
```

```
77      /**
78       *
79       * @param idComercio
80       * @return numero de registros afectados
81       */
82      @WebMethod(operationName = "delPagos")
83      public int delPagos(@WebParam(name="idComercio") String idComercio)
            {
84
85          PreparedStatement pstmt = null;
86          Connection pcon = null;
87          ResultSet rs = null;
88          int ret = 0;
89          String qry = null;
90
91          try {
92
93              // Crear una conexion u obtenerla del pool
94              pcon = getConnection();
95              qry = DELETE_PAGO_QRY;
96              errorLog(qry + "[idComercio=" + idComercio + "]");
97
98              // La preparacion del statement
99              // es automaticamente tomada de un pool en caso
100             // de que ya haya sido preparada con anterioridad
101             pstmt = pcon.prepareStatement(qry);
102             pstmt.setString(1, idComercio);
103             ret = pstmt.executeUpdate();
104
105             // Cerramos / devolvemos la conexion al pool
106             pcon.close();
107
108         } catch (Exception e) {
109             errorLog(e.toString());
110
111         } finally {
112             try {
113                 if (rs != null) {
114                     rs.close(); rs = null;
115                 }
116                 if (pstmt != null) {
117                     pstmt.close(); pstmt = null;
118                 }
119                 if (pcon != null) {
120                     closeConnection(pcon); pcon = null;
121                 }
122             } catch (SQLException e) {
123             }
124         }
125
126         return ret;
127     }
```

Cambios en GetPagos.java:

```
1   /**
2       * Procesa una petici&oacute;n HTTP tanto <code>GET</code> como <
            code>POST</code>.
```

```
3       * @param request objeto de petici&oacute;n
4       * @param response objeto de respuesta
5       */
6       protected void processRequest(HttpServletRequest request,
            HttpServletResponse response)
7       throws ServletException, IOException {
8
9           VisaDAOWSService service = new VisaDAOWSService();
10      VisaDAOWS dao = service.getVisaDAOWSPort();
11      String url_from_xml;
12      url_from_xml=getServletContext().getInitParameter("webmaster");
13
14      BindingProvider bp = (BindingProvider) dao;
15      bp.getRequestContext().put(BindingProvider.
            ENDPOINT_ADDRESS_PROPERTY, url_from_xml);
16
17                  /* Se recoge de la petici&oacute;n el par&aacute;metro
                        idComercio*/
18                  String idComercio = request.getParameter(
                        PARAM_ID_COMERCIO);
19
20                  /* Petici&oacute;n de los pagos para el comercio */
21                  List<PagoBean> aux = dao.getPagos(idComercio);
22                  PagoBean[] pagos = new PagoBean[aux.size()];
23      pagos = aux.toArray(pagos);
24
25          request.setAttribute(ATTR_PAGOS, pagos);
26          reenvia("/listapagos.jsp", request, response);
27          return;
28      }
```

Cambios en DelPagos.java:

```
1   /**
2       * Procesa una petici&oacute;n HTTP tanto <code>GET</code> como <
            code>POST</code>.
3       * @param request objeto de petici&oacute;n
4       * @param response objeto de respuesta
5       */
6       protected void processRequest(HttpServletRequest request,
            HttpServletResponse response)
7       throws ServletException, IOException {
8
9                   VisaDAOWSService service = new VisaDAOWSService();
10      VisaDAOWS dao = service.getVisaDAOWSPort();
11      String url_from_xml;
12      url_from_xml=getServletContext().getInitParameter("webmaster");
13
14      BindingProvider bp = (BindingProvider) dao;
15      bp.getRequestContext().put(BindingProvider.
            ENDPOINT_ADDRESS_PROPERTY, url_from_xml);
16
17                  /* Se recoge de la petici&oacute;n el par&aacute;metro
                        idComercio*/
18                  String idComercio = request.getParameter(
                        PARAM_ID_COMERCIO);
19
20                  /* Petici&oacute;n de los pagos para el comercio */
```

```
21                  int ret = dao.delPagos(idComercio);
22
23                  if (ret != 0) {
24                          request.setAttribute(ATTR_BORRADOS, ret);
25                          reenvia("/borradook.jsp", request, response);
26                  }
27                  else {
28                          reenvia("/borradoerror.jsp", request, response)
                                ;
29                  }
30          return;
31      }
```

11. Ejecicio 11

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://10.1.1.1:8080/P1-ws-ws/VisaDAOWSService?wsdl

Clases: Las clases correspondientes a los métodos del .xml. Se han creado para que sirvan de interfaces públicas.

12. Ejecicio 12

Ejercicio 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}

Código:

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

13. Ejecicio 13

Ejercicio 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.

Incluye evidencias en la memoria de la realización del ejercicio.

Capturas:

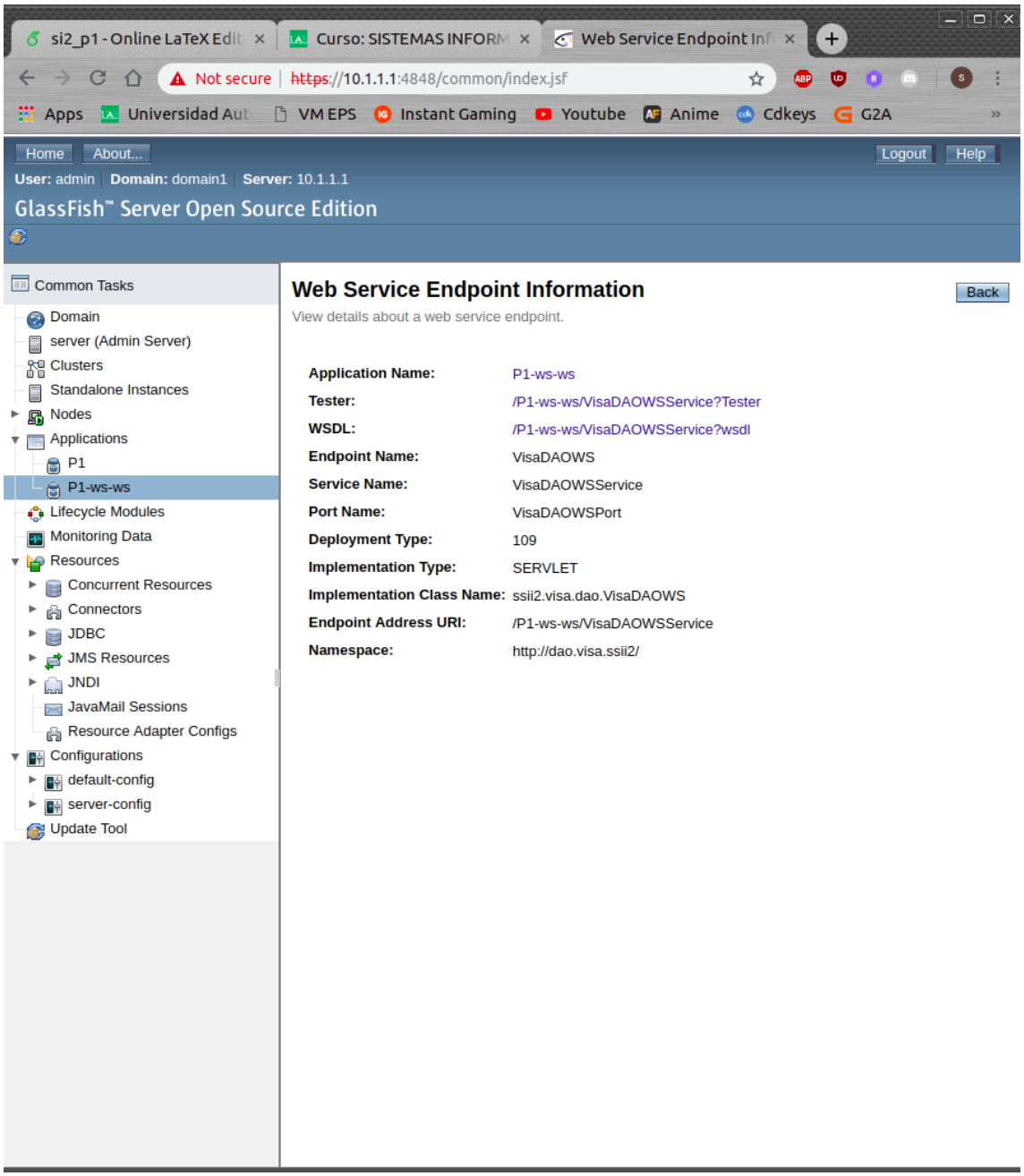


Figura 19: Servicio desplegado.

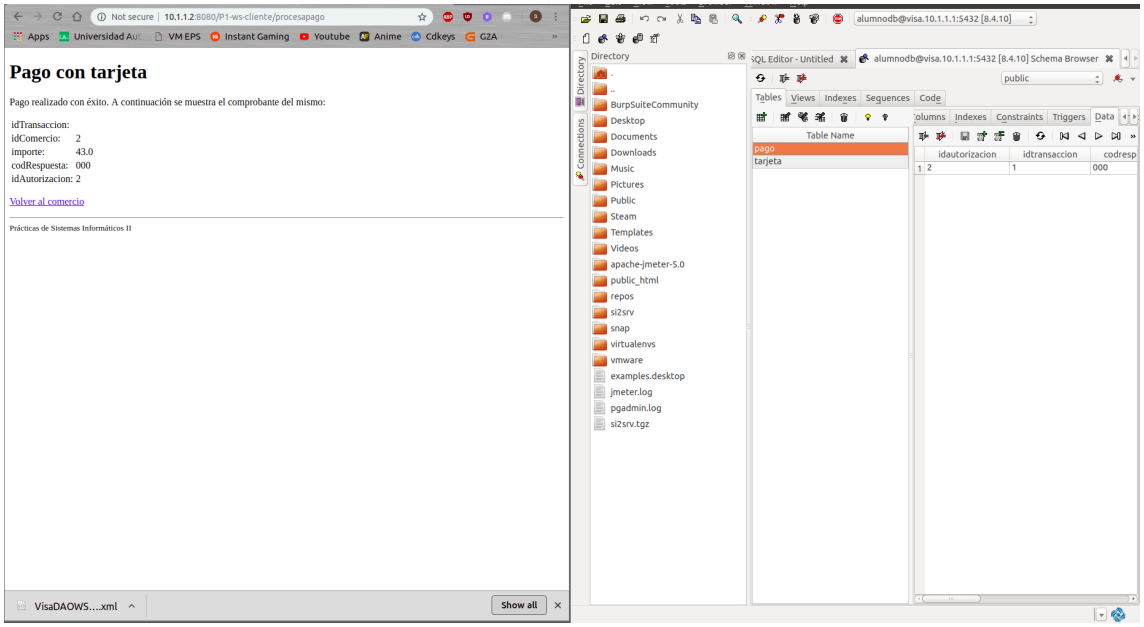


Figura 20: Cliente utilizando el servicio para realizar un pago.

```
as.host.client=10.1.1.2
as.host.server=10.1.1.1
as.port=4848
```

Figura 21: Direcciones IP.

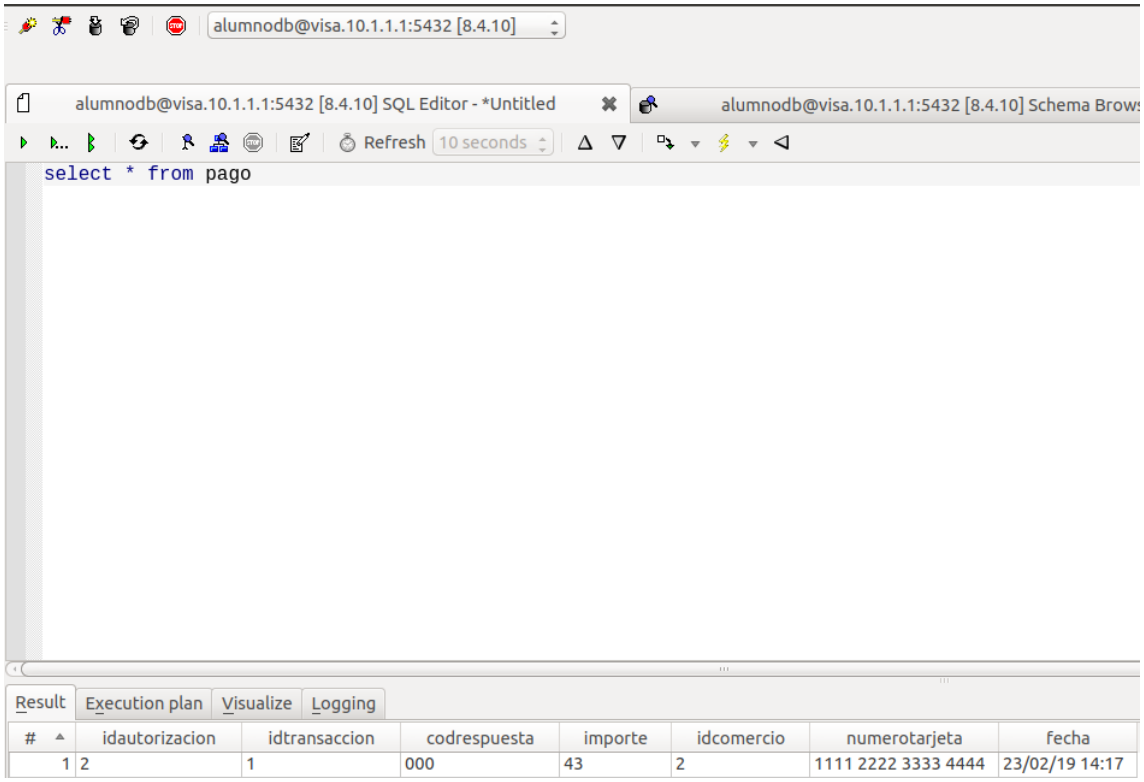


Figura 22: Consulta SQL con resultado.

14. Cuestiones

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.

Primero pasa por pago.html, donde las peticiones de pago son recibidas por el servlet ComienzaPago. Este servlet redirige a formdatosvisa.jsp que recoge los datos de la VISA y redirige al servlet ProcesaPago, que se encarga de validar los datos de la tarjeta, y como hay uno que produce un fallo (fecha de caducidad), se vuelve a llamar a formdatosvisa.jsp para que muestre el error.

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?

Los servlets encargado de ello es el ComienzaPago y el ProcesaPago.

Cuestion 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 ComienzaPago solicita idTransaccion, idComercio e importe, y el ProcesaPago se encarga de solicitar los datos de la tarjeta.

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?

Cuando se utiliza pago.html la invocación de servlets es la que hemos comentado anteriormente (ComienzaPago y ProcesaPago), debido a que pasa por dos formularios, mientras que cuando se utiliza la página de pruebas extendidas testbd.jsp, solo se pasa por un formulario, y la llamada se hace al servlet de ProcesaPago. El pago funciona correctamente porque la funcionalidad es la misma que cuando se usa pago.html nada más que toda la información se obtiene en un mismo formulario y es recibida por un mismo servlet, mientras que en pago.html la información del primer formulario se le pasa al segundo servlet (además de la que obtiene del segundo formulario), por lo tanto una vez rellenos los dos formularios los datos obtenidos son equivalentes a los obtenidos en un formulario de testbd.jsp.