

MEMORIA PRÁCTICA 1A

Arquitectura Java EE

Alejandro Santorum Varela - alejandro.santorum@estudiante.uam.es

Rafael Sánchez Sánchez - rafael.sanchezs@estudiante.uam.es

Prácticas Sistemas Informáticos II

Práctica 1A Pareja 7 Grupo 2401

24 de febrero de 2020

Contents

1	Introducción	2
2	Configuración	2
3	Ejercicio 1	2
4	Ejercicio 2	11
5	Ejercicio 3	15
6	Ejercicio 4	17
7	Ejercicio 5	18
8	Ejercicio 6	25
9	Ejercicio 7	27
10	Ejercicio 8	32
11	Ejercicio 9	32
12	Ejercicio 10	33
13	Ejercicio 11	35
14	Ejercicio 12	36
15	Ejercicio 13	37
16	Cuestiones	41
16.1	Cuestión 1	41
16.2	Cuestión 2	41
16.3	Cuestión 3	41
16.4	Cuestión 4	41
17	Conclusiones	42
18	Bibliografía	42

1 Introducción

Nos encontramos en la primera práctica del curso de Sistemas Informáticos II. El objetivo fundamental de esta práctica es adentrarse en la arquitectura de JAVA EE desde el punto de vista del arquitecto de software.

A continuación se muestran los resultados obtenidos y las respuestas a las preguntas solicitadas.

2 Configuración

Antes de intentar realizar cualquier ejercicio deberemos configurar el ordenador del laboratorio para poder ejecutar los ficheros necesarios:

1. Seleccionar la imagen importada (si2srv).
2. Generar MAC *address* del Network Adapter (NAT) y Network Adapter 2 (Bridged) de la máquina virtual (en *settings*).
3. Nos *logueamos* con usuario **si2** y contraseña **2020sid0s**.
4. La primera vez que entramos en la máquina virtual deberemos configurar una dirección IP única y sin conflictos en el rango 10.X.Y.Z. En nuestro caso **10.1.7.Z** donde $Z \in \{1, 2, 3, 4\}$.
5. Ahora, para acceder de forma remota, será necesario asignar a la interfaz de red del Host la dirección IP en el rango 10.X.Y.Z. En nuestro caso **10.1.7.Z** donde $Z \in \{1, 2, 3, 4\}$.
6. Definiremos la variable de entorno de Glassfish:
`export J2EE_HOME=/opt/glassfish4/glassfish` (en una terminal del Host).
7. A continuación podremos acceder de forma remota, con un terminal del host, usando el comando **ssh si2@10.1.7.Z**.
8. Finalmente, podremos iniciar el servidor de Glassfish con el comando:
asadmin start-domain domain1.

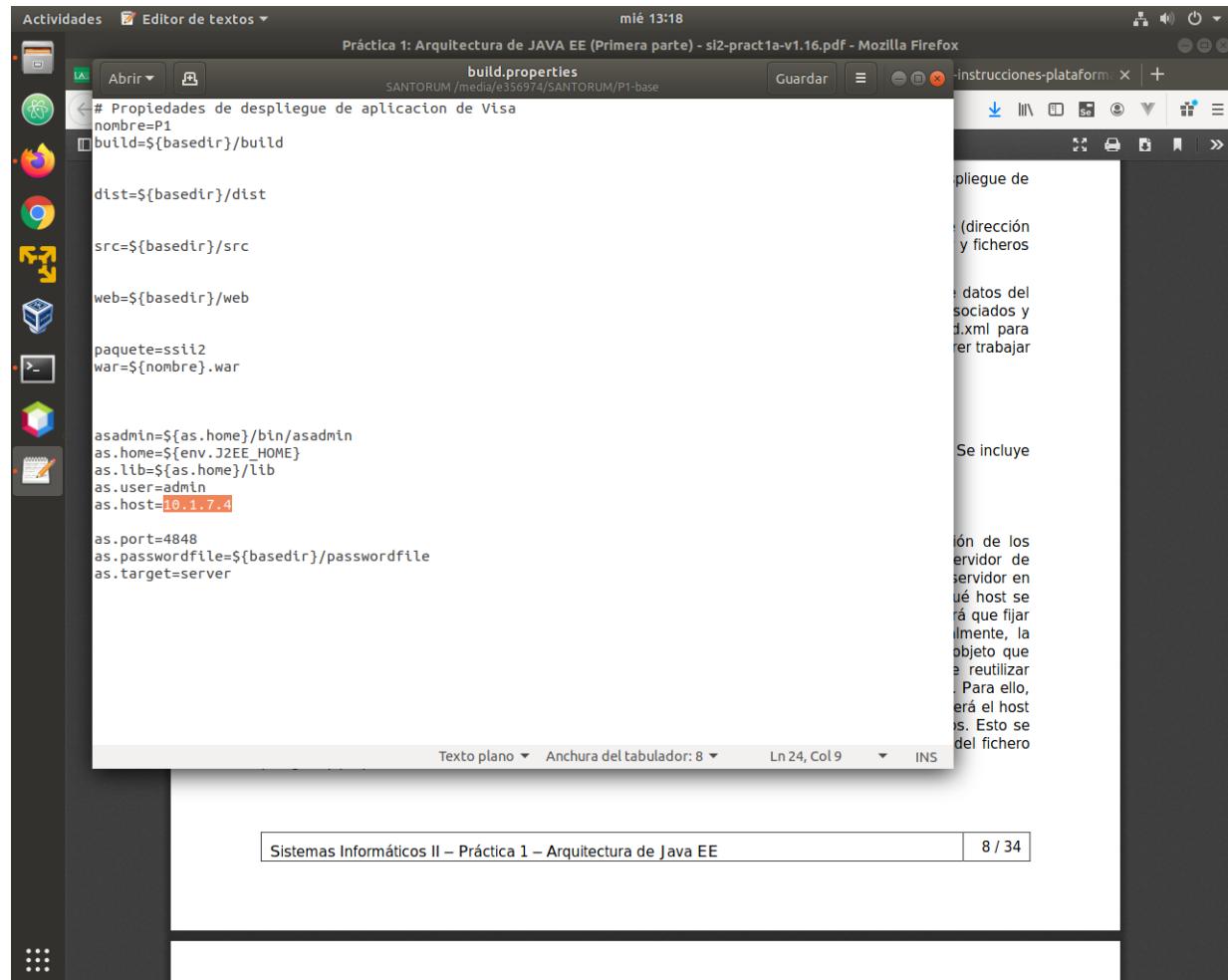
3 Ejercicio 1

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

1. 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.
2. Realice un pago contra la aplicación web empleando el navegador en la ruta:
`http://10.X.Y.Z:8080/P1`
3. Conéctese a la base de datos (usando el cliente Tora por ejemplo) y obtenga evidencias de que el pago se ha realizado.
4. 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.

Respuesta a la cuestión:

Empezamos modificando los ficheros necesarios para arrancar la aplicación web. En primer lugar modificaremos el fichero **build.properties** en la línea donde se especifica **as.host** y lo asignaremos a la IP **10.1.7.4**.



```
# Propiedades de despliegue de aplicación de Visa
nombre=1
build=${basedir}/build

dist=${basedir}/dist

src=${basedir}/src

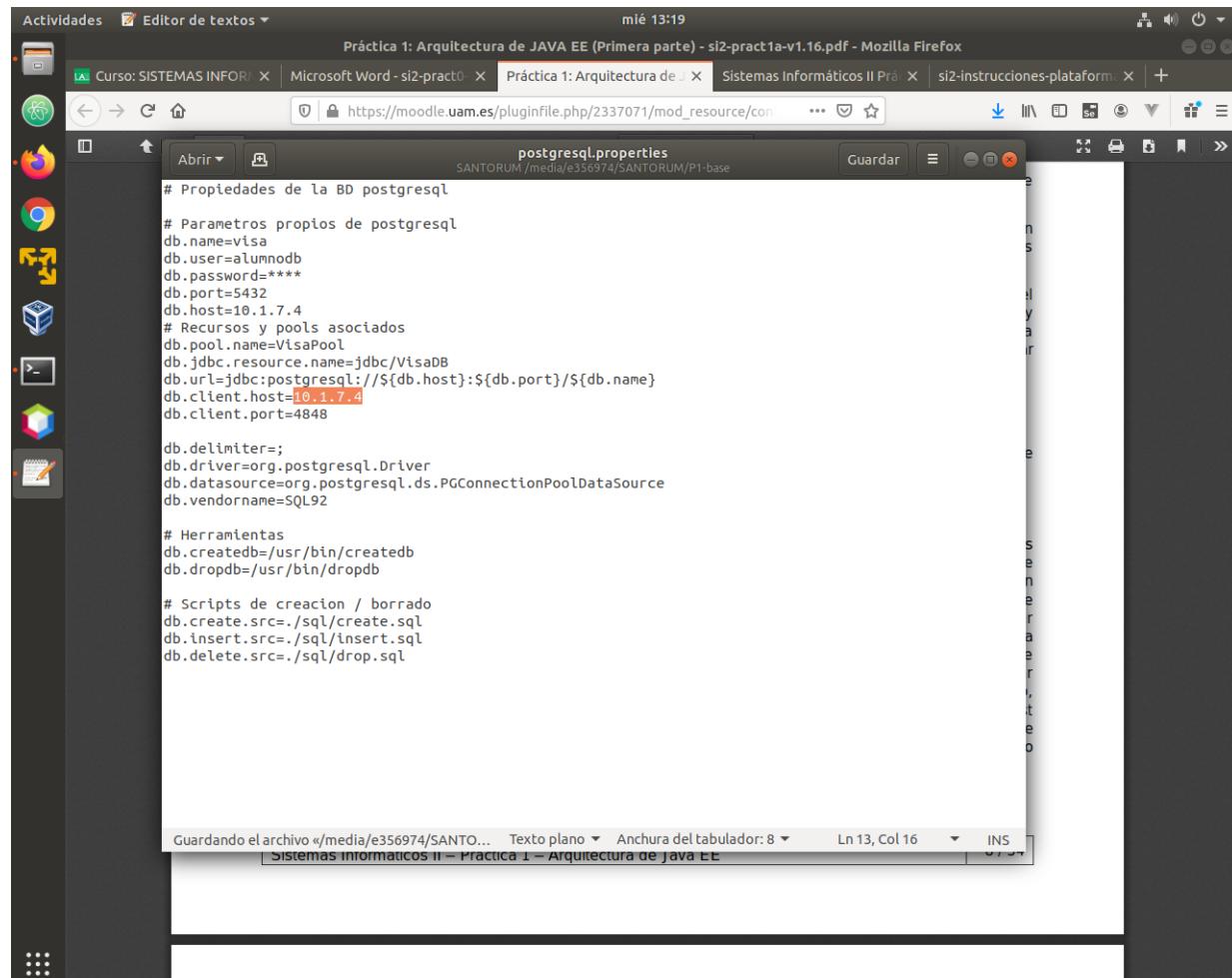
web=${basedir}/web

paquete=ssii2
war=${nombre}.war

asadmin=${as.home}/bin/asadmin
as.home=${env.J2EE_HOME}
as.lib=${as.home}/lib
as.user=admin
as.host=10.1.7.4

as.port=4848
as.passwordfile=${basedir}/passwordfile
as.target=server
```

Haremos lo mismo con el fichero **postgresql.properties**, en la variable **db.host** y en la variable **db.client.host**



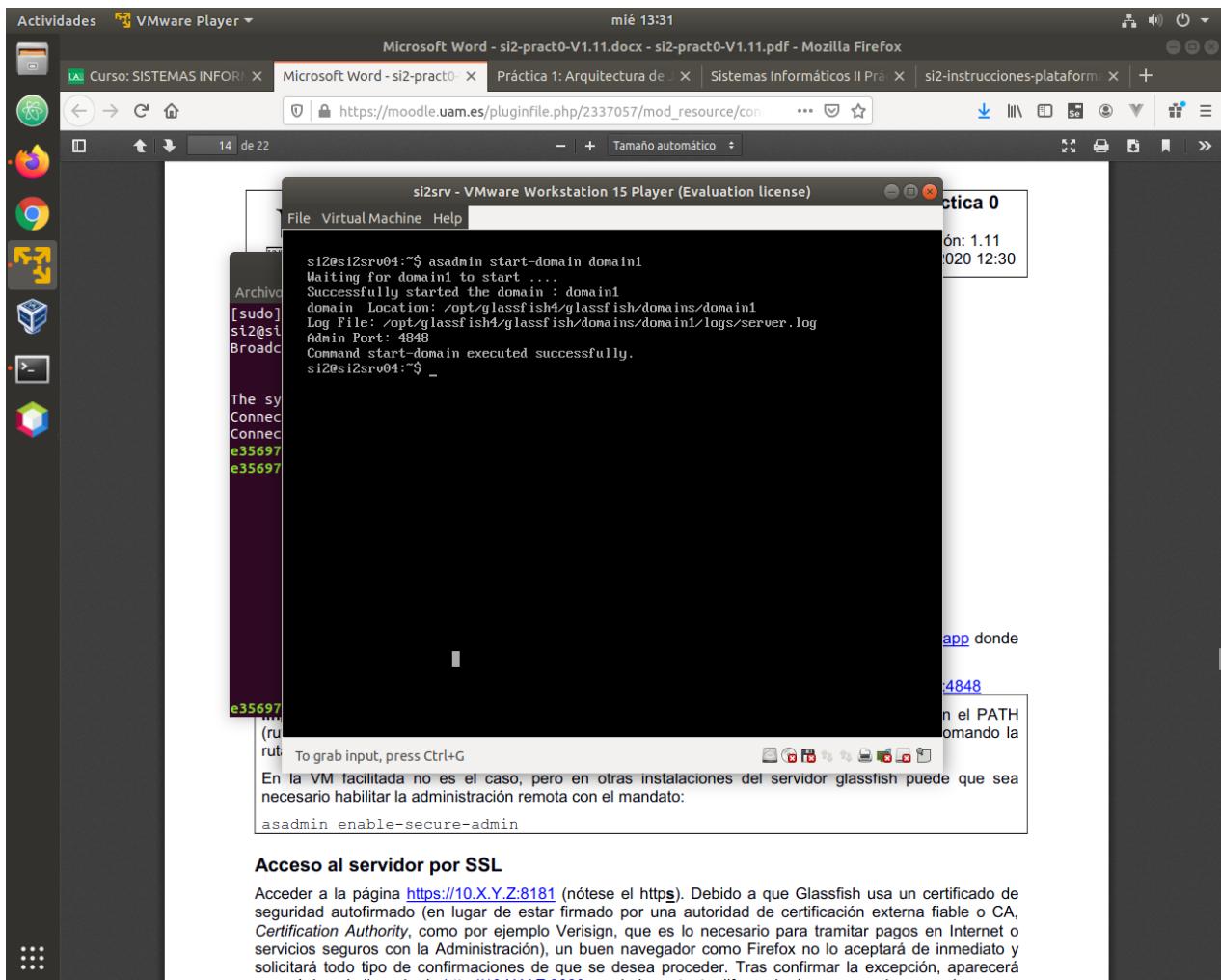
```
# Propiedades de la BD postgresql
# Parametros propios de postgresql
db.name=vista
db.user=alumnodb
db.password=*****
db.port=5432
db.host=10.1.7.4
# Recursos y pools asociados
db.pool.name=VisaPool
db.jdbc.resource.name=jdbc/VisaDB
db.url=jdbc:postgresql://${db.host}:${db.port}/${db.name}
db.client.host=10.1.7.4
db.client.port=4848

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

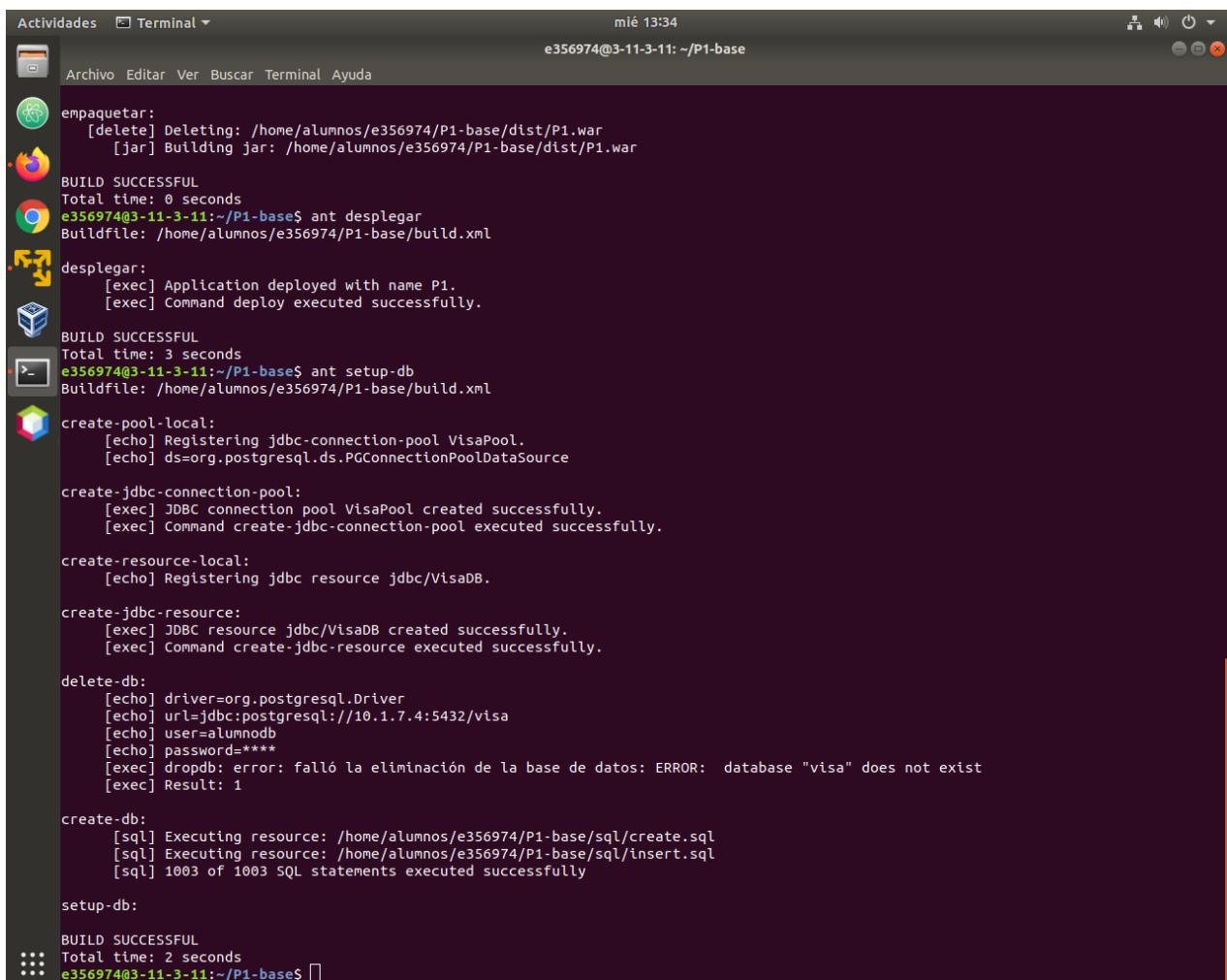
# Herramientas
db.createdb=/usr/bin/createdb
db.dropdb=/usr/bin/dropdb

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

Ahora deberemos arrancar el servidor con los nuevos cambios realizados



A continuación deberemos compilar todos los ficheros necesarios para el despliegue de la aplicación web. Empezamos iniciando la base de datos con el comando **ant setup-db** (dentro del directorio P1-base, donde se encuentran los ficheros fuente de la app).



Después, compilamos los ficheros necesarios con **ant compilar**

Actividades Terminal mié 13:34 Microsoft Word - si2-pract0-V1.11.docx - si2-pract0-V1.11.pdf - Mozilla Firefox

Curso: SISTEMAS INFOR X Microsoft Word - si2-pract0- X Práctica 1: Arquitectura de d X Sistemas Informáticos II Práct X si2-instrucciones-plataform X +

https://moodle.uam.es/pluginfile.php/2337057/mod_resource/con ... Tamano automático

14 de 22

File Virtual Machine Help

e356974@3-11-3-11: ~/P1-base

Archivo Editar Ver Buscar Terminal Ayuda

Total time: 0 seconds
e356974@3-11-3-11:~/P1-base\$ ant limpiar
Buildfile: /home/alumnos/e356974/P1-base/build.xml

limpiar:
[delete] Deleting directory /home/alumnos/e356974/P1-base/build

BUILD SUCCESSFUL
Total time: 0 seconds
e356974@3-11-3-11:~/P1-base\$ ant compilar
Buildfile: /home/alumnos/e356974/P1-base/build.xml

montar-jerarquia:
[mkdir] Created dir: /home/alumnos/e356974/P1-base/build
[mkdir] Created dir: /home/alumnos/e356974/P1-base/build/WEB-INF/classes
[mkdir] Created dir: /home/alumnos/e356974/P1-base/build/WEB-INF/lib

compilar:
[javac] Compiling 17 source files to /home/alumnos/e356974/P1-base/build/WEB-INF/classes

BUILD SUCCESSFUL
Total time: 0 seconds
e356974@3-11-3-11:~/P1-base\$

To grab input, press Ctrl+G

En la VM facilitada no es el caso, pero en otras instalaciones del servidor glassfish puede que sea necesario habilitar la administración remota con el mandato:

asadmin enable-secure-admin

Y preparamos del paquete con la aplicación web (fichero .war) con **ant empaquetar**.

```
File Virtual Machine Help
e356974@3-11-3-11: ~/P1-base
Archivo Editar Ver Buscar Terminal Ayuda
montar jerarquía:
[mkdir] Created dir: /home/alumnos/e356974/P1-base/build
[mkdir] Created dir: /home/alumnos/e356974/P1-base/build/WEB-INF/classes
[mkdir] Created dir: /home/alumnos/e356974/P1-base/build/WEB-INF/lib

compilar:
[javac] Compiling 17 source files to /home/alumnos/e356974/P1-base/build/WEB-INF/classes

BUILD SUCCESSFUL
Total time: 0 seconds
e356974@3-11-3-11:~/P1-base$ ant empaquetar
Buildfile: /home/alumnos/e356974/P1-base/build.xml

preparar-web-inf:
[copy] Copying 11 files to /home/alumnos/e356974/P1-base/build

empaquetar:
[delete] Deleting: /home/alumnos/e356974/P1-base/dist/P1.war
[jar] Building jar: /home/alumnos/e356974/P1-base/dist/P1.war

BUILD SUCCESSFUL
Total time: 0 seconds
e356974@3-11-3-11:~/P1-base$ [run]
(run)
run: To grab input, press Ctrl+G
En la VM facilitada no es el caso, pero en otras instalaciones del servidor glassfish puede que sea necesario habilitar la administración remota con el mandato:
asadmin enable-secure-admin
```

Finalmente, podemos desplegar la aplicación con **ant desplegar**

```

Actividades Terminal
Microsoft Word - si2-pract0-V1.11.docx - si2-pract0-V1.11.pdf - Mozilla Firefox
Curso: SISTEMAS INFOR... Microsoft Word - si2-pract0-V1.11.pdf - Práctica 1: Arquitectura de... | Sistemas Informáticos II Práct... | si2-instrucciones-plataform...
mié 13:34
https://moodle.uam.es/pluginfile.php/2337057/mod_resource/con...
14 de 22
File Virtual Machine Help
e356974@3-11-3-11:~/P1-base
Archivo Editar Ver Buscar Terminal Ayuda
BUILD SUCCESSFUL
Total time: 0 seconds
e356974@3-11-3-11:~/P1-base$ ant empaquetar
Buildfile: /home/alumnos/e356974/P1-base/build.xml

preparar-web-inf:
[copy] Copying 11 files to /home/alumnos/e356974/P1-base/build

empaquetar:
[delete] Deleting: /home/alumnos/e356974/P1-base/dist/P1.war
[jar] Building jar: /home/alumnos/e356974/P1-base/dist/P1.war

BUILD SUCCESSFUL
Total time: 0 seconds
e356974@3-11-3-11:~/P1-base$ ant desplegar
Buildfile: /home/alumnos/e356974/P1-base/build.xml

desplegar:
[exec] Application deployed with name P1.
[exec] Command deploy executed successfully.

BUILD SUCCESSFUL
Total time: 3 seconds
e356974@3-11-3-11:~/P1-base$ 
[run]
[run] To grab input, press Ctrl+G
[run] En la VM facilitada no es el caso, pero en otras instalaciones del servidor glassfish puede que sea necesario habilitar la administración remota con el mandato:
[run] asadmin enable-secure-admin

```

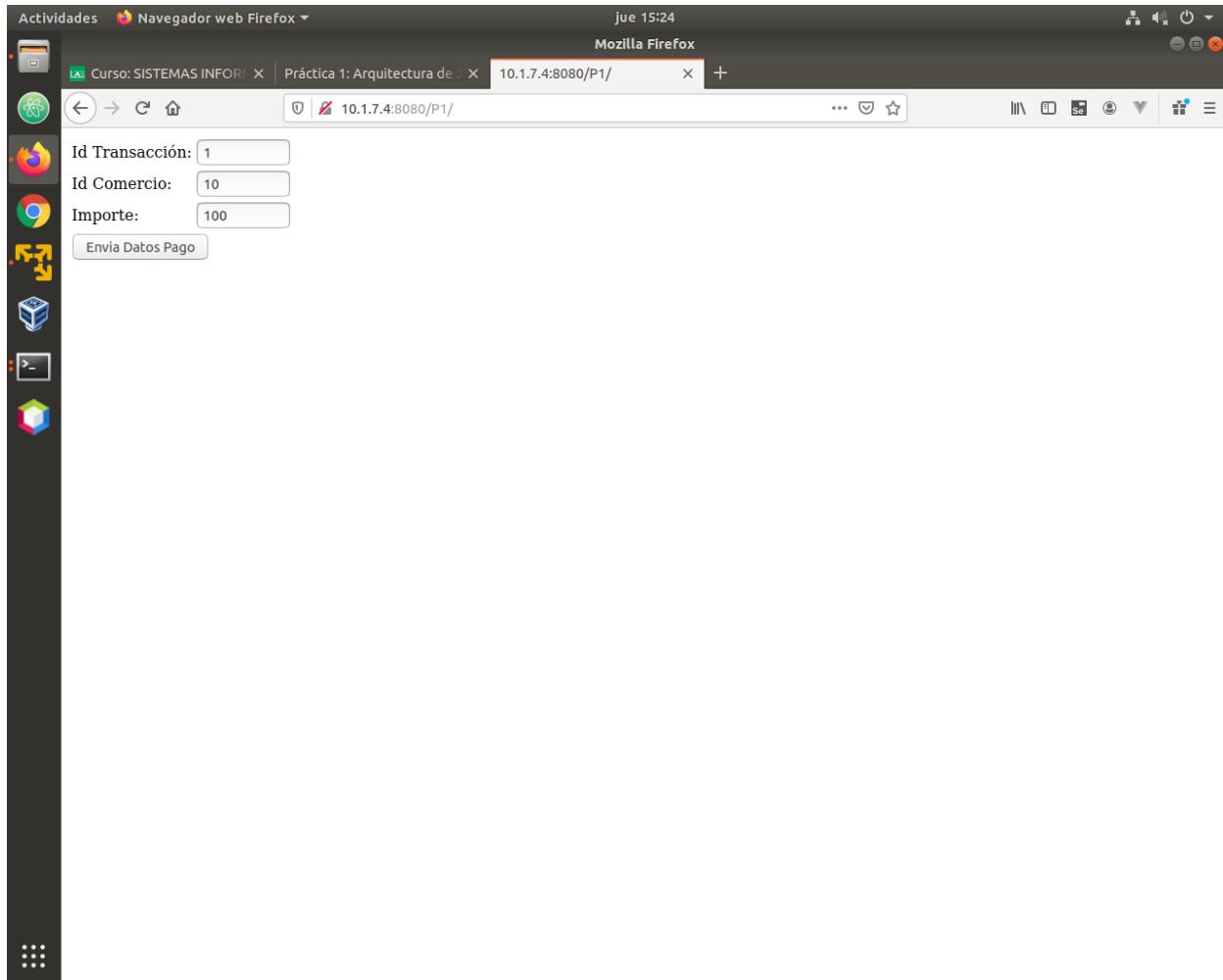
Comentar que toda esta secuencia de comandos se puede abreviar por el comando **ant todo**, que se encarga de ejecutar secuencialmente los comandos **ant** descritos anteriormente. Ahora, con la aplicación lanzada, es el momento de intentar realizar un pago en ella. Para ello deberemos conocer de mano una tarjeta de crédito válida registrada en la base de datos, así como el resto de datos del cliente. Esta información la podemos obtener de dos maneras básicas:

1. Con la aplicación Tora, conectándose a la base de datos de forma remota y comprobando la tabla **tarjeta**.

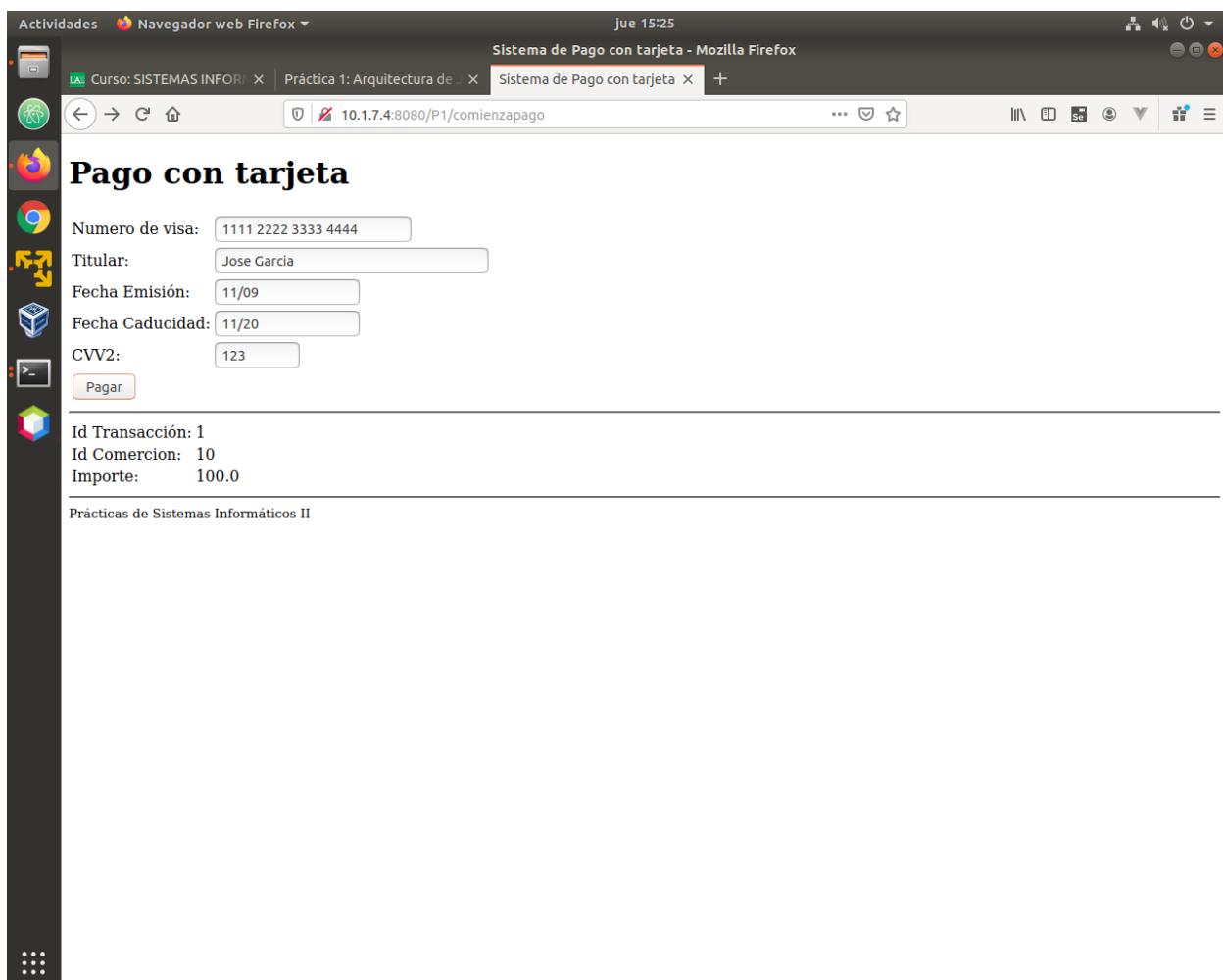
#	numerotarjeta	titular	validadesde	validahasta	codigove
1	1111 2222 3333 4444	Jose Garcia	11/09	11/20	123
2	2347 4840 5058 7931	Gabriel Avila Locke	11/09	01/20	207
3	1530 6462 9686 4119	Alberto Mas Reyes	05/09	09/20	105
4	0694 4853 5696 2094	Restituta Torres Coll	08/09	03/20	547
5	8365 5667 6696 6481	Enjuto Gonzalez Torres	03/09	11/20	421
6	7772 8952 5915 5042	John Dominguez Lopez	01/10	10/20	317
7	7581 5513 5721 0259	Jose Garau Mas	01/08	09/20	252
8	6513 0633 4651 1154	Gabriel Locke Martinez	04/08	02/20	681
9	7557 9649 3955 5976	Irene Avila Morales	01/10	03/20	185
10	7109 3591 3164 3418	Armando Avila Pomares	11/09	08/20	041
11	0847 5800 5700 6200	Emiliano Espejo Martin Diaz	10/10	08/20	402

2. Con la terminal conectada a la máquina virtual (usando ssh) podremos ejecutar el comando **psql visa -U alumnodb** para conectarse a la base de datos y finalmente con la consulta SQL **select * from tarjeta** podremos obtener la misma información que en el punto 1.

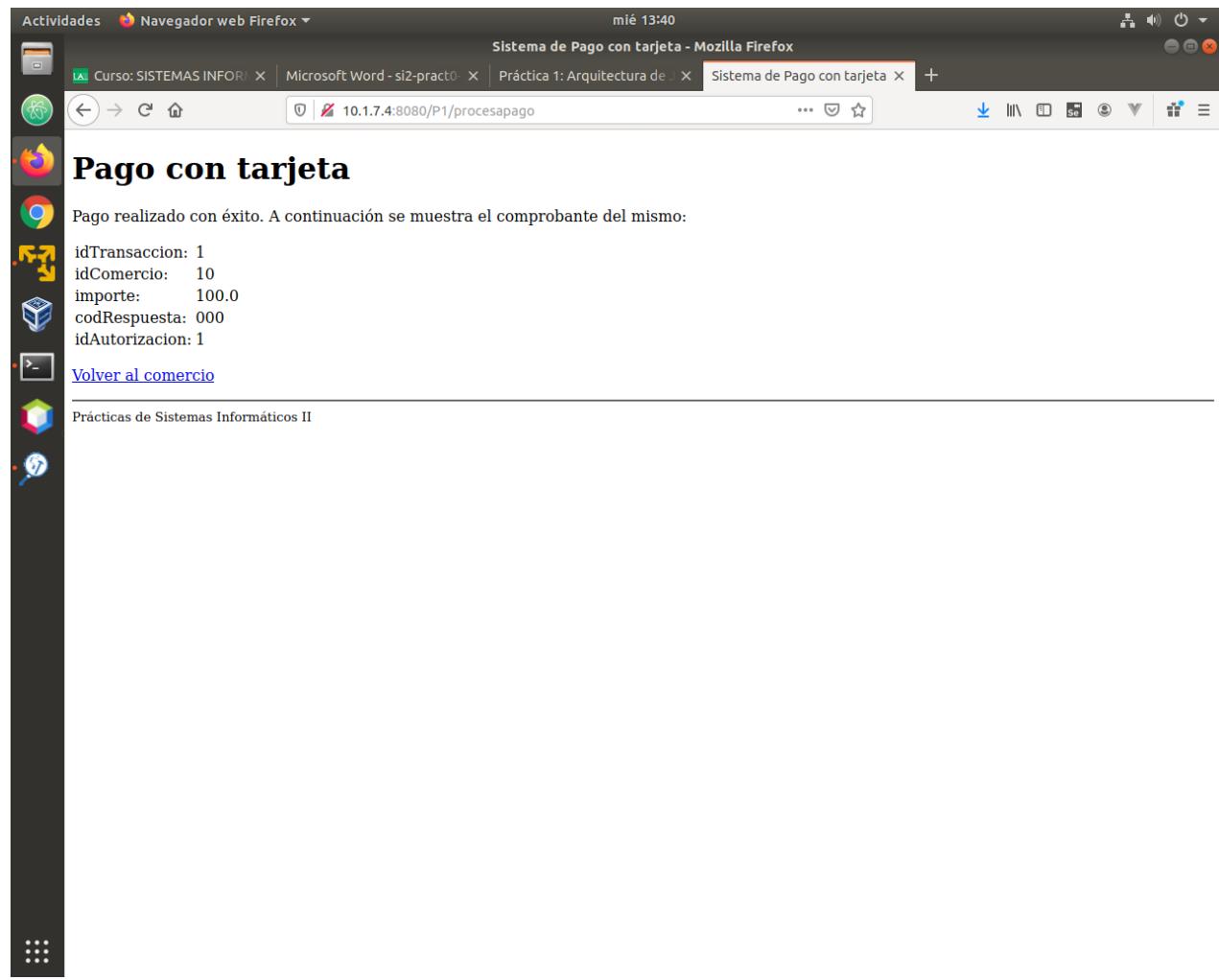
Una vez recolectados estos datos, podemos ya realizar un pago. Con un navegador web entraremos en la ruta **http://10.1.7.4:8080/P1** e introduciremos los datos de la transacción.



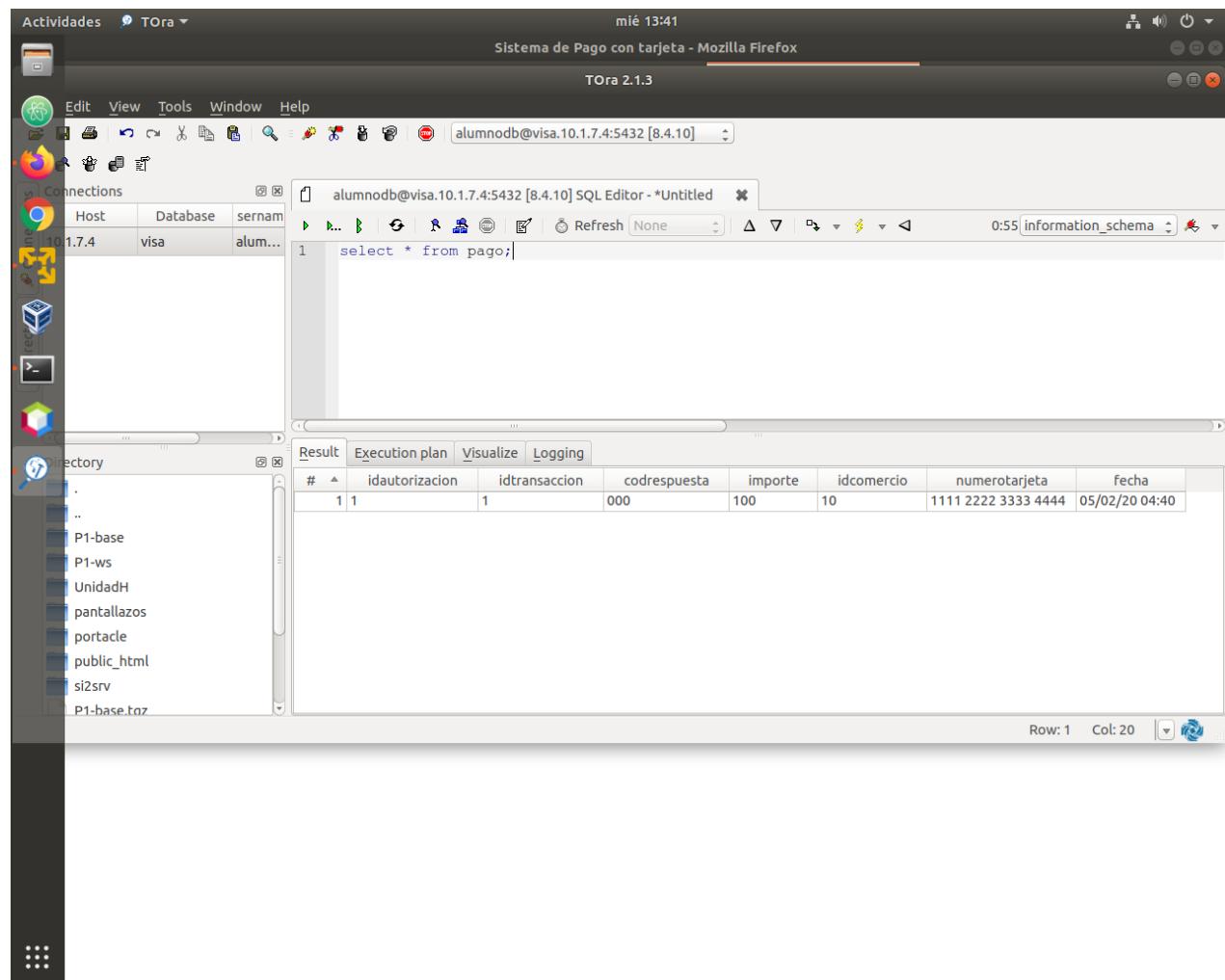
Pulsando en 'Envia Datos Pago' llegaremos a otro formulario donde introduciremos los datos de un cliente cualquiera obtenidos anteriormente.



Pulsando en 'Pagar' y si hemos realizado todo según se ha descrito obtendremos un pago exitoso.



Podemos comprobarlo en la base de datos (con Tora o con PostgreSQL):



Y también podremos comprobarlo en la página de pruebas extendida:
<http://10.1.7.4:8080/P1/testbd.jsp>
Escribiendo en el apartado 'Consulta de pagos' el ID del comercio especificado en la transacción.

Actividades Navegador web Firefox mié 13:42 Sistema de Pago con tarjeta - Mozilla Firefox

Sistema de Pago con tarjeta - Mozilla Firefox

Curso: SISTEMAS INFOR | Microsoft Word - si2-pract0 | Práctica 1: Arquitectura de | Sistema de Pago con tarjeta | Sistema de Pago con tarjeta | +

10.1.7.4:8080/P1/testbd.jsp

Pago con tarjeta

Proceso de un pago

Id Transacción:

Id Comercio:

Importe:

Número de visa:

Titular:

Fecha Emisión:

Fecha Caducidad:

CVV2:

Modo debug: True False

Direct Connection: True False

Use Prepared: True False

Consulta de pagos

Id Comercio:

Borrado de pagos

Id Comercio:

Prácticas de Sistemas Informáticos II

Actividades Navegador web Firefox mié 13:42 Sistema de Pago con tarjeta - Mozilla Firefox

Sistema de Pago con tarjeta - Mozilla Firefox

Curso: SISTEMAS INFOR | Microsoft Word - si2-pract0 | Práctica 1: Arquitectura de | Sistema de Pago con tarjeta | Sistema de Pago con tarjeta | +

10.1.7.4:8080/P1/getpagos

Pago con tarjeta

Lista de pagos del comercio 10

idTransaccion	Importe	codRespuesta	idAutorizacion
1	100.0	000	1

[Volver al comercio](#)

Prácticas de Sistemas Informáticos II

Finalmente podremos borrar el pago realizado desde la misma página de pruebas en el apartado 'Borrado de pagos', escribiendo el ID del comercio especificado en la transacción.

Actividades Navegador web Firefox mié 13:42 Sistema de Pago con tarjeta - Mozilla Firefox

Curso: SISTEMAS INFOR | Microsoft Word - si2-pract0 | Práctica 1: Arquitectura de | Sistema de Pago con tarjeta | Sistema de Pago con tarjeta +

10.1.7.4:8080/P1/testbd.jsp

Pago con tarjeta

Proceso de un pago

Id Transacción:

Id Comercio:

Importe:

Número de visa:

Titular:

Fecha Emisión:

Fecha Caducidad:

CVV2:

Modo debug: True False

Direct Connection: True False

Use Prepared: True False

Consulta de pagos

Id Comercio:

Borrado de pagos

Id Comercio:

Prácticas de Sistemas Informáticos II

Actividades Navegador web Firefox mié 13:42 Sistema de Pago con tarjeta - Mozilla Firefox

Curso: SISTEMAS INFOR | Microsoft Word - si2-pract0 | Práctica 1: Arquitectura de | Sistema de Pago con tarjeta | Sistema de Pago con tarjeta +

10.1.7.4:8080/P1/delpagos

Pago con tarjeta

Se han borrado 1 pagos correctamente para el comercio 10

[Volver al comercio](#)

Prácticas de Sistemas Informáticos II

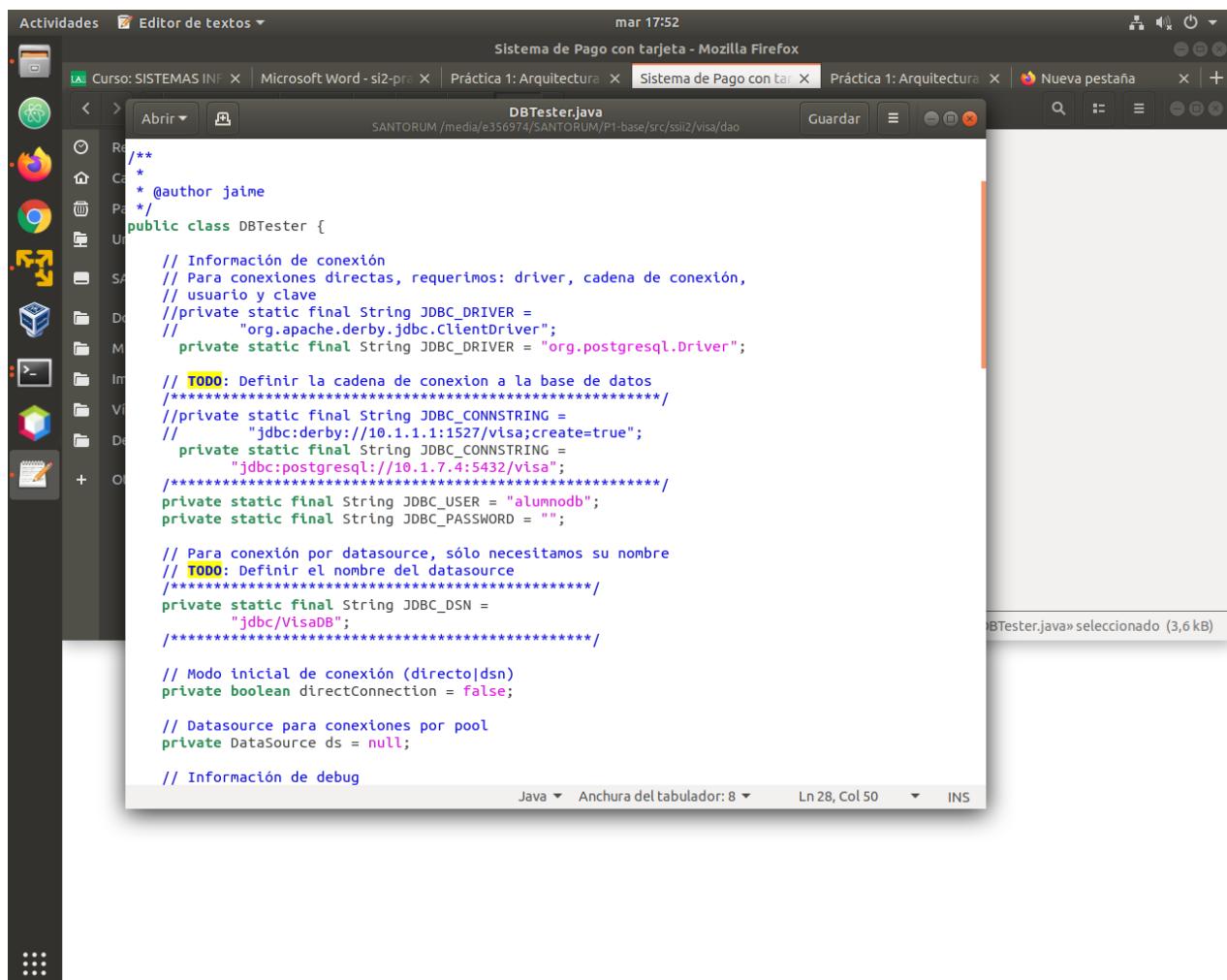
4 Ejercicio 2

Enunciado: 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.X.Y.Z:8080/P1/testbd.jsp> y pruebe a realizar un pago utilizando la conexión directa y pruebe a listarla y eliminarlo. Adjunte en la memoria evidencias de este proceso, incluyendo capturas de pantalla.

Respuesta a la cuestión:

Empezamos modificando el fichero **DBTester.java** localizado en **P1-base/src/ssii2/visa/dao**. Modificamos las siguientes variables, que estaban configuradas de forma incorrecta para una conexión directa:

- JDBC_DRIVER: cambiamos "org.apache.derby.jdbc.ClientDriver" por "org.postgresql.Driver".
- JDBC_CONNSTRING: cambiamos "jdbc:derby://10.1.1.1:1527/visa;create=true" por "jdbc:postgresql://10.1.7.4:5432/visa".
- JDBC_USER debe ser igual a "alumnodb".
- JDBC_PASSWORD es indiferente, por ejemplo "".



The screenshot shows a Linux desktop environment with a file editor window open. The window title is "DBTester.java". The code in the editor is as follows:

```
/*
 * 
 * @author jaime
 */
public class DBTester {

    // Información de conexión
    // Para conexiones directas, requerimos: driver, cadena de conexión,
    // usuario y clave
    private static final String JDBC_DRIVER =
        "org.apache.derby.jdbc.ClientDriver";
    private static final String JDBC_DRIVER = "org.postgresql.Driver";

    // TODO: Definir la cadena de conexión a la base de datos
    // ****
    private static final String JDBC_CONNSTRING =
        "jdbc:derby://10.1.1.1:1527/visa;create=true";
    private static final String JDBC_CONNSTRING =
        "jdbc:postgresql://10.1.7.4:5432/visa";
    // ****
    private static final String JDBC_USER = "alumnodb";
    private static final String JDBC_PASSWORD = "";

    // Para conexión por datasource, sólo necesitamos su nombre
    // TODO: Definir el nombre del datasource
    // ****
    private static final String JDBC_DSN =
        "jdbc/VisaDB";
    // ****

    // Modo inicial de conexión (directo/dsn)
    private boolean directConnection = false;

    // Datasource para conexiones por pool
    private DataSource ds = null;

    // Información de debug
}
```

Ahora ya podemos acceder a la página de pruebas extendida <http://10.1.7.4:8080/P1/testbd.jsp> para probar a realizar un pago con conexión directa. Para ello utilizaremos los datos usados en el ejercicio 1, y seleccionaremos la opción **Direct Connection**.

The screenshot shows a Mozilla Firefox window titled "Sistema de Pago con tarjeta - Mozilla Firefox". The address bar displays the URL `10.1.7.4:8080/P1/testbd.jsp`. The main content area contains the following sections:

- Pago con tarjeta**
- Proceso de un pago**
- Form fields:
 - Id Transacción:
 - Id Comercio:
 - Importe:
 - Número de visa:
 - Titular:
 - Fecha Emisión:
 - Fecha Caducidad:
 - CVV2:
- Mode debug: True False
- Direct Connection: True False
- Use Prepared: True False
-

- Consulta de pagos**
- Form fields:
- Id Comercio:
-

- Borrado de pagos**
- Form fields:
- Id Comercio:
-

Prácticas de Sistemas Informáticos II

Si todo ha ido bien, deberemos obtener el siguiente mensaje de confirmación. Pulsaremos en 'Volver al comercio' para seguir haciendo pruebas.

The screenshot shows a Mozilla Firefox window titled "Sistema de Pago con tarjeta - Mozilla Firefox". The address bar displays the URL `10.1.7.4:8080/P1/procesapago`. The main content area contains the following message:

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

```
idTransaccion: 1  
idComercio: 10  
importe: 100.0  
codRespuesta: 000  
idAutorizacion: 1
```

[Volver al comercio](#)

Prácticas de Sistemas Informáticos II

A continuación comprobaremos que efectivamente el pago se ha realizado con éxito, utilizando la misma página de pruebas e introduciendo el ID del comercio utilizado anteriormente.

Actividades Navegador web Firefox mar 17:53 Sistema de Pago con tarjeta - Mozilla Firefox

Sistema de Pago con tarjeta - Mozilla Firefox

Curso: SISTEMAS INF | Microsoft Word - si2-práctica 1: Arquitectura | Práctica 1: Arquitectura | Sistema de Pago con tarjeta | Práctica 1: Arquitectura | Nueva pestaña

10.1.7.4:8080/P1/testbd.jsp

Pago con tarjeta

Proceso de un pago

Id Transacción:

Id Comercio:

Importe:

Número de visa:

Titular:

Fecha Emisión:

Fecha Caducidad:

CVV2:

Modo debug: True False

Direct Connection: True False

Use Prepared: True False

Consulta de pagos

Id Comercio:

Borrado de pagos

Id Comercio:

Prácticas de Sistemas Informáticos II

Actividades Navegador web Firefox mar 17:53 Sistema de Pago con tarjeta - Mozilla Firefox

Sistema de Pago con tarjeta - Mozilla Firefox

Curso: SISTEMAS INF | Microsoft Word - si2-práctica 1: Arquitectura | Práctica 1: Arquitectura | Sistema de Pago con tarjeta | Práctica 1: Arquitectura | Nueva pestaña

10.1.7.4:8080/P1/getpagos

Pago con tarjeta

Lista de pagos del comercio 10

idTransaccion	Importe	codRespuesta	idAutorizacion
1	100.0	000	1

[Volver al comercio](#)

Prácticas de Sistemas Informáticos II

Finalmente podremos borrar el pago realizado desde la misma página de pruebas, en el apartado 'Borrado de pagos' y escribiendo el ID del comercio especificado en la transacción.

The screenshot shows a Firefox browser window with the title 'Sistema de Pago con tarjeta - Mozilla Firefox'. The address bar displays '10.1.7.4:8080/P1/testbd.jsp'. The main content area is titled 'Pago con tarjeta' and contains a form for a payment process. The fields include:

- Id Transacción: [Input field]
- Id Comercio: [Input field]
- Importe: [Input field]
- Número de visa: [Input field]
- Titular: [Input field]
- Fecha Emisión: [Input field]
- Fecha Caducidad: [Input field]
- CVV2: [Input field]
- Modo debug: True False
- Direct Connection: True False
- Use Prepared: True False

A 'Pagar' button is located at the bottom of the form.

Consulta de pagos

Id Comercio: [Input field]

Borrado de pagos

Id Comercio: 10

Prácticas de Sistemas Informáticos II

The screenshot shows a Firefox browser window with the title 'Sistema de Pago con tarjeta - Mozilla Firefox'. The address bar displays '10.1.7.4:8080/P1/delpagos'. The main content area is titled 'Pago con tarjeta' and displays the message: 'Se han borrado 1 pagos correctamente para el comercio 10'. Below this message is a link 'Volver al comercio'. The sidebar on the left shows various application icons.

5 Ejercicio 3

Enunciado: 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.

Respuesta a la cuestión:

Examinando el fichero **postgresql.properties** es fácil obtener el nombre del recurso JDBC del DataSource y el nombre del *pool*.

- **Nombre JDBC DataSource:** jdbc/visaDB
- **Nombre pool:** VisaPool

The screenshot shows a Linux desktop environment. In the foreground, a terminal window is open with a yellow gradient background. The terminal window title is "Actividades" and the application title is "Editor de textos". The file being edited is "postgresql.properties" located at "SANTORUM /media/e356974/SANTORUM/P1-base". The content of the file includes configuration for a PostgreSQL database named "visaDB" with host "10.1.7.4", port "5432", and driver "org.postgresql.Driver". It also defines a connection pool named "VisaPool" with resource name "jdbc/visaDB". Other sections include "Propiedades de la BD postgresql", "Parametros propios de postgresql", "Recursos y pools asociados", "Herramientas", and "Scripts de creacion / borrado". The terminal window shows some Java code at the bottom. In the background, a file manager window is visible with icons for "ld.xml", "passwordfile", and "postgresql.properties".

Con esta información podremos acceder a la Consola de Administración y comprobar el estado de los recursos JDBC y *pool* de conexiones.

Para ser más exactos, deberemos ir a la ruta <http://10.1.7.4:4848/> y allí podremos ver los recursos solicitados en **Common Tasks → Resources → JDBC → JDBC Connection Pools → VisaPool**.

En esta ruta, aparte de ver toda la configuración de los recursos solicitados, también podemos realizar un **Ping JDBC** a la base de datos.

Todo esto lo podemos ver en la siguiente imagen aportada.

En esta página es fácil ver los valores de las variables solicitadas.

- **Initial and Minimum Pool Size:** 8 conexiones
- **Maximum Pool Size:** 32 conexiones
- **Pool Resize Quantity:** 2 conexiones
- **Idle Timeout:** 300 segundos
- **Max Wait Time:** 60000 milisegundos

La constante **Initial and Minimum Pool Size** indica el número mínimo e inicial de hilos de la *pool*. Dependiendo del número de solicitudes de conexión este número debería variar si buscamos tener un buen rendimiento. Si nuestra aplicación tiene un tráfico constante y de un tamaño alto, este número debería ser alto, evitando levantar continuamente nuevas conexiones. Por el contrario, si nuestra aplicación tiene un tráfico relativamente bajo, deberíamos configurar este valor con un número menor para no sobrecargar el sistema de hilos en paralelo levantados.

La misma filosofía se puede utilizar con la constante **Maximum Pool Size**, que es la cantidad máxima de conexiones creables para atender a clientes. Si tenemos mucho tráfico de clientes, este número debería ser alto; y bajo si nuestro número de clientes habituales es menor.

También, con un pensamiento parecido, podemos analizar la constante **Pool Resize Quantity**, que es la cantidad de hilos que se crean (caso a) o se destruyen (caso b) de la *pool* de hilos cuando se requieren más conexiones (caso a) o cuando hay varios hilos inactivos durante mucho tiempo (caso b).

Finalmente, las constantes **Idle Timeout** y **Max Wait Time** se utilizan para establecer el tiempo máximo de espera por un hilo a ser solicitado y el tiempo máximo que el servidor espera a que el cliente actúe respectivamente. Estos dos valores tienen que ser medianamente analizados ya que valores bajos provocarán que los hilos se destruyan muy rápido si **Idle Timeout** es bajo o que el servidor no le proporcione tiempo suficiente al cliente para contestar si el valor **Max Wait Time** es bajo. Análogamente, si los valores son altos, provocarán largas esperas innecesarias, reduciendo el rendimiento y aumentando la sobrecarga del servidor.

6 Ejercicio 4

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

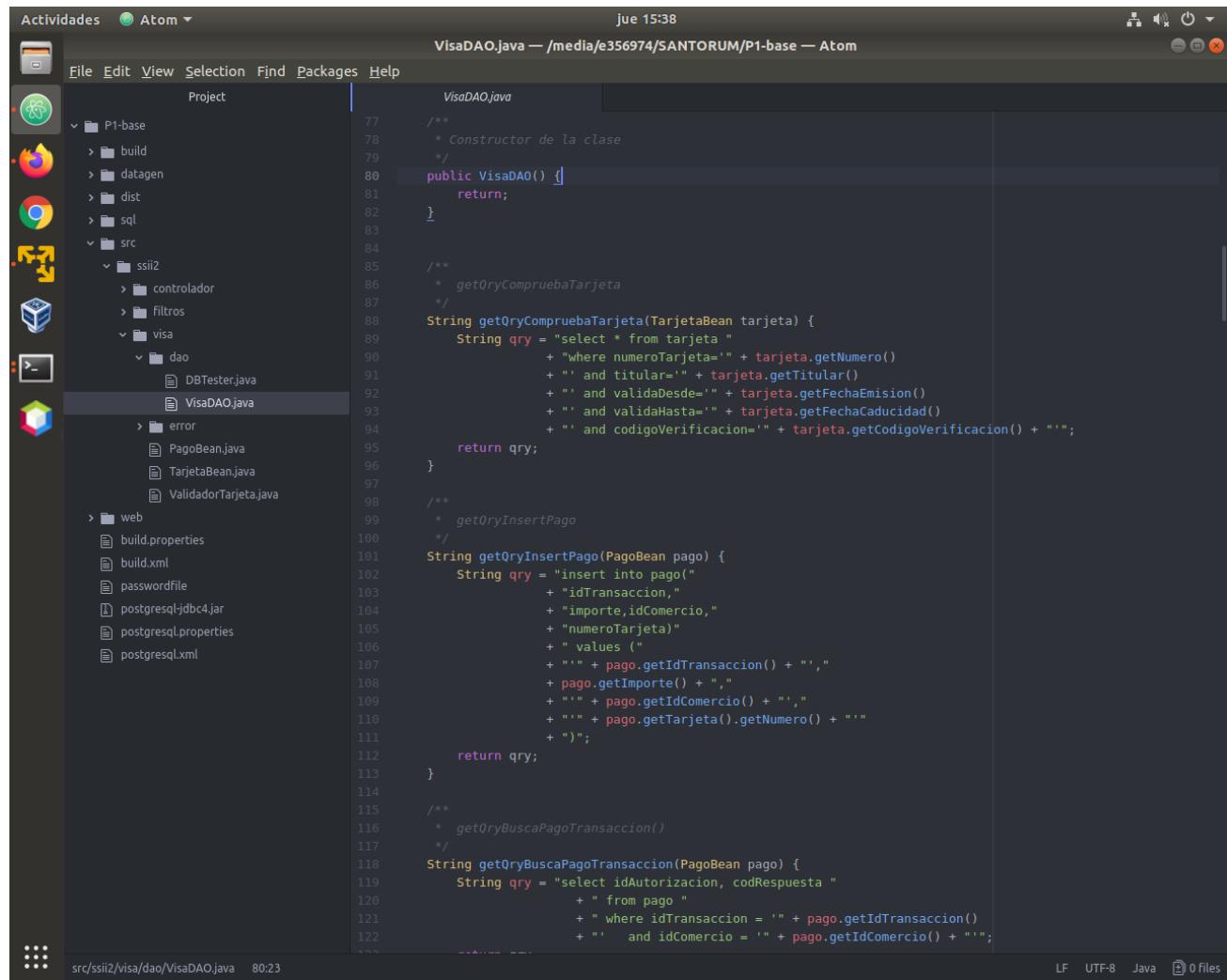
- Consulta de si una tarjeta es válida.
- Ejecución del pago.

Incluya en la memoria de prácticas dichas consultas.

Respuesta a la cuestión:

Se pueden encontrar en el fichero **VisaDAO.java**, situado en el directorio: **P1-base/src/ssii2/visa/dao**, en las siguientes líneas:

- Consulta de si una tarjeta es válida: método **getQryCompruebaTarjeta** entre las líneas 88 y 96.
- Consulta de ejecución del pago: método **getQryInsertPago** entre las líneas 101 y 113.



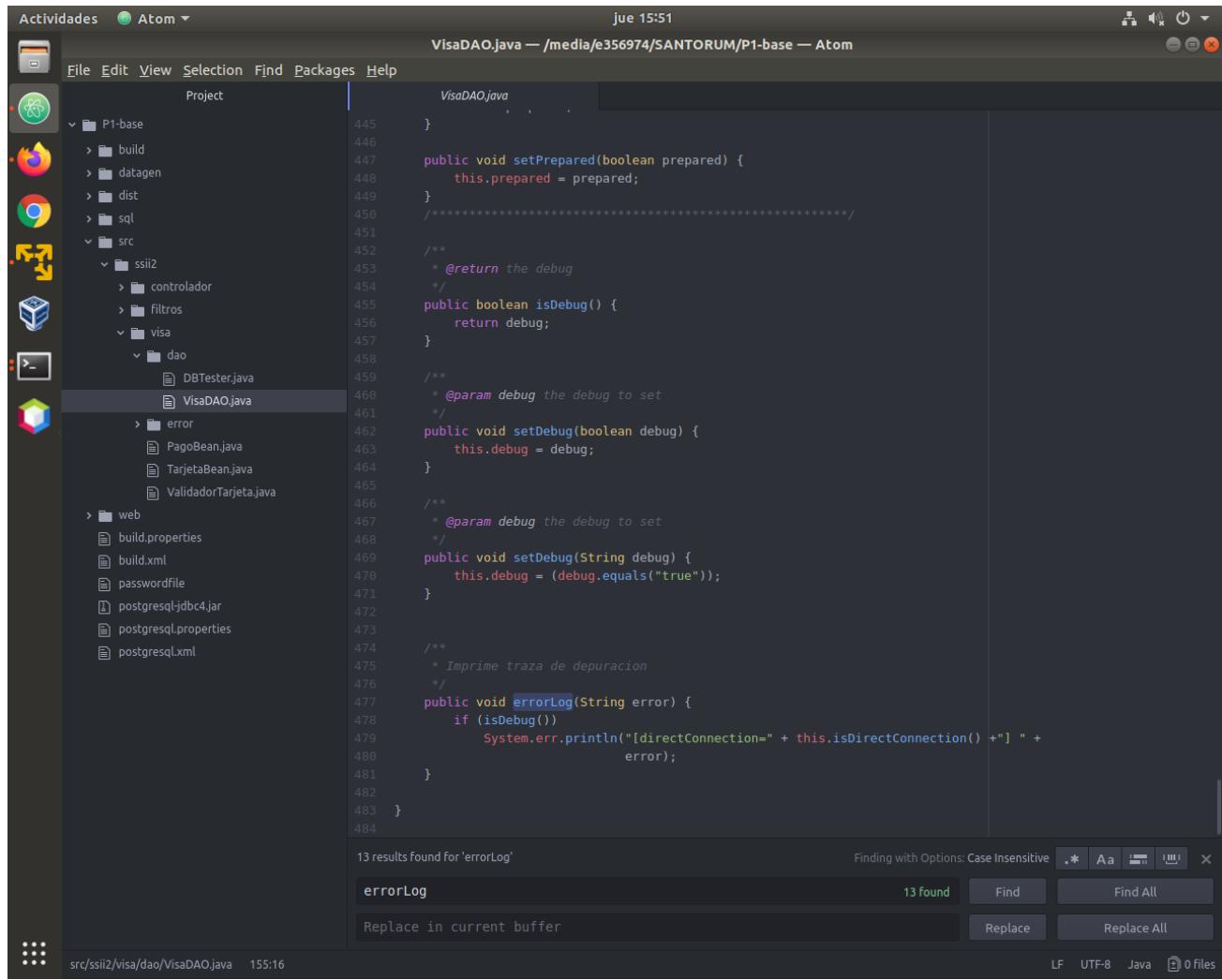
```
Actividades Atom ▾
File Edit View Selection Find Packages Help
Project VisaDAO.java — /media/e356974/SANTORUM/P1-base — Atom
jue 15:38
77 /**
78 * Constructor de la clase
79 */
80 public VisaDAO() {
81     return;
82 }
83
84
85 /**
86 * getQryCompruebaTarjeta
87 */
88 String getQryCompruebaTarjeta(TarjetaBean tarjeta) {
89     String qry = "select * from tarjeta "
90         + "where numeroTarjeta=" + tarjeta.getNumero()
91         + " and titular=" + tarjeta.getTitular()
92         + " and validadesde=" + tarjeta.getFechaEmision()
93         + " and validaHasta=" + tarjeta.getFechaCaducidad()
94         + " and codigoVerificacion=" + tarjeta.getCodigoVerificacion() + "";
95     return qry;
96 }
97
98 /**
99 * getQryInsertPago
100 */
101 String getQryInsertPago(PagoBean pago) {
102     String qry = "insert into pago"
103         + "idTransaccion,"
104         + "importe,idComercio,"
105         + "numeroTarjeta"
106         + " values ("
107         + "'" + pago.getIdTransaccion() + "',"
108         + pago.getImporte() + ","
109         + "'" + pago.getIdComercio() + "'"
110         + "'" + pago.getTarjeta().getNumero() + "'"
111         + ")";
112     return qry;
113 }
114
115 /**
116 * getQryBuscaPagoTransaccion()
117 */
118 String getQryBuscaPagoTransaccion(PagoBean pago) {
119     String qry = "select idAutorizacion, codRespuesta "
120         + " from pago "
121         + " where idTransaccion = '" + pago.getIdTransaccion()
122         + "' and idComercio = '" + pago.getIdComercio() + "'";
123 }
```

7 Ejercicio 5

Enunciado: 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.

Respuesta a la cuestión:

Entrando en el fichero **VisaDAO.java**, situado en el directorio ya mencionado anteriormente, podemos buscar el método **errorLog**, que se usa 12 veces en el código de ese fichero para realizar acciones de *debugging* y mostrar trazas de ejecución más significativas que las que habitualmente salen.



```
Actividades Atom ▾
File Edit View Selection Find Packages Help
Project VisaDAO.java
445     }
446
447     public void setPrepared(boolean prepared) {
448         this.prepared = prepared;
449     }
450     /**
451      * @return the debug
452      */
453     public boolean isDebug() {
454         return debug;
455     }
456
457     /**
458      * @param debug the debug to set
459      */
460     public void setDebug(boolean debug) {
461         this.debug = debug;
462     }
463
464     /**
465      * @param debug the debug to set
466      */
467     public void setDebug(String debug) {
468         this.debug = (debug.equals("true"));
469     }
470
471     /**
472      * Imprime traza de depuracion
473      */
474     public void errorLog(String error) {
475         if (isDebug())
476             System.err.println("[directConnection=" + this.isDirectConnection() +"] " +
477                             error);
478     }
479
480
481
482
483 }
```

Las partes en donde se utiliza este método son en otros métodos que acceden a la base de datos con el objetivo de hacer una consulta y/o modificación. Al tratarse de procedimientos delicados, es necesario preservar en el *log* trazas exhaustivas por si se da el caso en que algo falla y poder actuar y arreglarlo.

Los métodos que se ayudan de errorLog son, principalmente, el método de **comprobación de tarjeta válida**, el método de **insercción de pago**, el procedimiento de **consulta de pagos** y el método de **eliminación de pagos**.

A continuación realizaremos un pago y observaremos qué trazas obtenemos en el log del servidor de aplicaciones.

Utilizaremos para ello los datos obtenidos en el ejercicio 1 y con la opción de **debug activada**.

The screenshot shows a Mozilla Firefox window titled "Sistema de Pago con tarjeta - Mozilla Firefox". The address bar displays "10.1.7.4:8080/P1/testbd.jsp". The main content area contains a form titled "Pago con tarjeta" with the following fields:

- Id Transacción:
- Id Comercio:
- Importe:
- Número de visa:
- Titular:
- Fecha Emisión:
- Fecha Caducidad:
- CVV2:

Below the form are three radio buttons:

- Modo debug: True False
- Direct Connection: True False
- Use Prepared: True False

A "Pagar" button is located at the bottom of the form.

Consulta de pagos

Id Comercio:

Borrado de pagos

Id Comercio:

Prácticas de Sistemas Informáticos II

Pulsando en 'Pagar' deberíamos obtener la siguiente salida, un pago exitoso.

The screenshot shows a Mozilla Firefox window titled "Sistema de Pago con tarjeta - Mozilla Firefox". The address bar displays "10.1.7.4:8080/P1/procesapago". The main content area contains a message and some payment details:

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

```
idTransaccion: 1
idComercio: 10
importe: 100.0
codRespuesta: 000
idAutorizacion: 3
```

[Volver al comercio](#)

Prácticas de Sistemas Informáticos II

Si comprobamos en el servidor de aplicaciones el *log* podemos ver qué ha ocurrido por detrás.

The screenshot shows the Mozilla Firefox Log Viewer interface. The URL is <https://10.1.7.4:4848/common/logViewer/logViewer.jsf?instanceName=server&logLevel=INFO&viewResults=true>. The log level is set to INFO. The log file is server.log. The table below shows Log Viewer Results (40) from Record Number 328 to 367. A red box highlights several entries related to a card being checked:

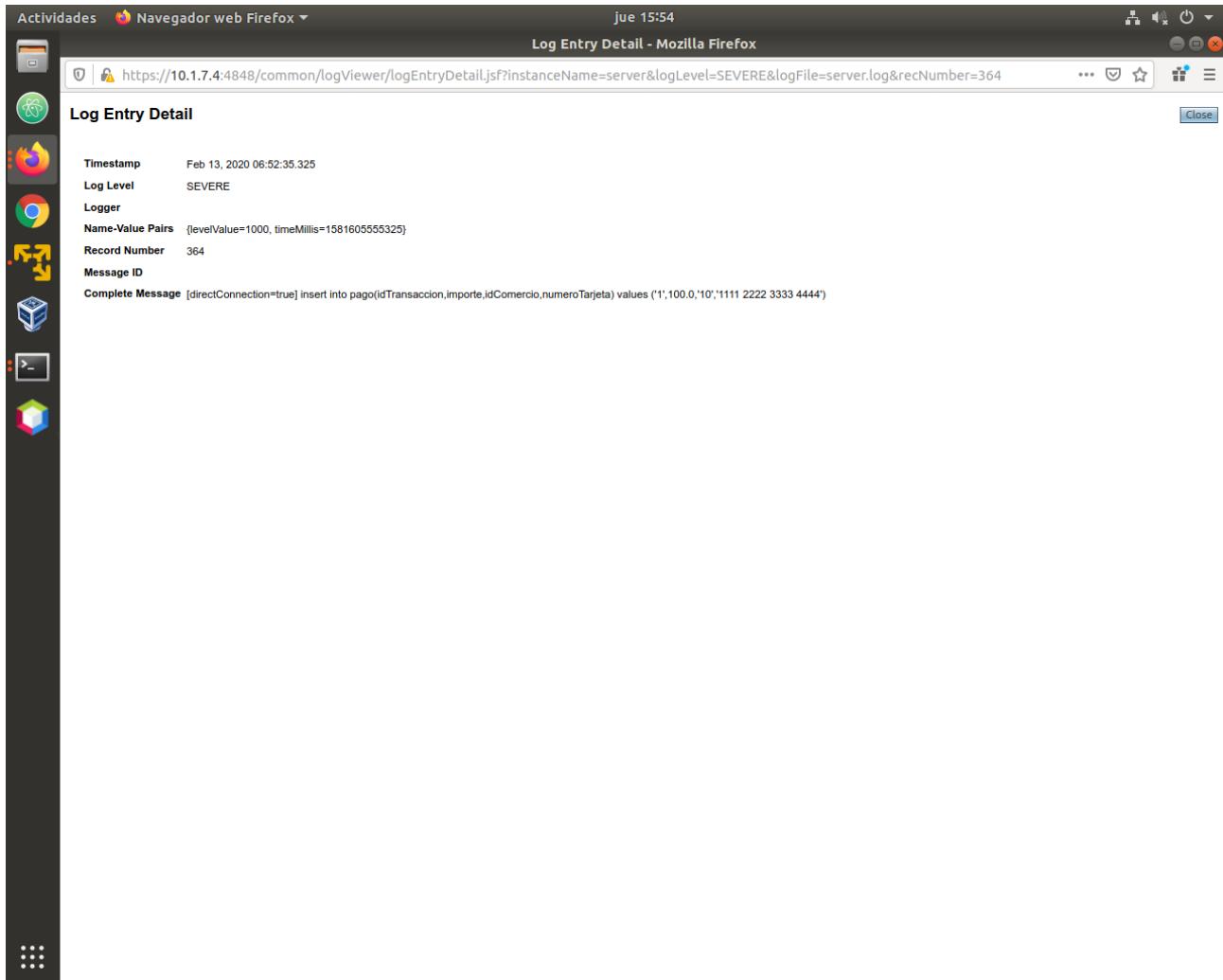
Record Number	Log Level	Message	Logger	Timestamp	Name-Value Pairs
367	INFO	WebModule[null] ServletContext.log():[INFO] Acceso correcto:/testbd.jsp(details)	javax.enterprise.web	Feb 13, 2020 06:52:46.352	(levelValue=800, timeMillis=1581605566352)
366	INFO	WebModule[null] ServletContext.log():[INFO] Acceso correcto:/testbd.jsp(details)	javax.enterprise.web	Feb 13, 2020 06:52:45.458	(levelValue=800, timeMillis=1581605565458)
365	SEVERE	[directConnection=true] select idAutorizacion, codRespuesta from pago where idTransaccion = '1' ... (details)		Feb 13, 2020 06:52:35.327	(levelValue=1000, timeMillis=1581605555327)
364	SEVERE	[directConnection=true] insert into pago(idTransaccion,importe,idComercio,numeroTarjeta) values ('1'... (details))		Feb 13, 2020 06:52:35.325	(levelValue=1000, timeMillis=1581605555325)
363	SEVERE	[directConnection=true] select * from tarjeta where numeroTarjeta='1111 2222 3333 4444' and titular='...' (details)		Feb 13, 2020 06:52:35.323	(levelValue=1000, timeMillis=1581605555323)
362	INFO	WebModule[null] ServletContext.log():[INFO] Acceso correcto:/procesapago(details)	javax.enterprise.web	Feb 13, 2020 06:52:35.317	(levelValue=800, timeMillis=1581605555317)
361	INFO	WebModule[null] ServletContext.log():[INFO] Acceso correcto:/testbd.jsp(details)	javax.enterprise.web	Feb 13, 2020 06:52:10.514	(levelValue=800, timeMillis=1581605530514)
360	INFO	Admin Console: Initializing Session Attributes...(details)	org.glassfish.admingui	Feb 13, 2020 06:28:31.461	(levelValue=800, timeMillis=1581604111461)
359	INFO	Redirecting to /index.jsf(details)	org.glassfish.admingui	Feb 13, 2020 06:28:31.362	(levelValue=800, timeMillis=1581604111362)
358	INFO	Listening to REST requests at context: /management/domain.(details)	javax.enterprise.admin.rest	Feb 13, 2020 06:28:31.349	(levelValue=800, timeMillis=1581604111349)
357	SEVERE	The SSL certificate has expired: [[Version: V3 Subject: CN=Entrust.net Secure Server Certifica... (details)]	javax.enterprise.system.security.ssl	Feb 13, 2020 06:28:29.987	(levelValue=1000, timeMillis=1581604109987)
356	SEVERE	The SSL certificate has expired: [[Version: V3 Subject: CN=Entrust.net Certification Authority... (details)]	javax.enterprise.system.security.ssl	Feb 13, 2020 06:28:29.986	(levelValue=1000, timeMillis=1581604109986)
355	SEVERE	The SSL certificate has expired: [[Version: V1 Subject: EMAILADDRESS=info@valicert.com, CN=ht... (details)]	javax.enterprise.system.security.ssl	Feb 13, 2020 06:28:29.986	(levelValue=1000, timeMillis=1581604109985)
354	SEVERE	The SSL certificate has expired: [[Version: V3 Subject: CN=Class 2 Primary CA, O=Certplus, C=... (details)]	javax.enterprise.system.security.ssl	Feb 13, 2020 06:28:29.985	(levelValue=1000, timeMillis=1581604109985)
353	SEVERE	The SSL certificate has expired: [[Version: V1 Subject: CN=GTE CyberTrust Global Root, OU=GTE... (details)]	javax.enterprise.system.security.ssl	Feb 13, 2020 06:28:29.985	(levelValue=1000, timeMillis=1581604109985)
352	SEVERE	The SSL certificate has expired: [[Version: V3 Subject: CN=UTN - DATAcorp SGC, OU=http://www.u... (details)]	javax.enterprise.system.security.ssl	Feb 13, 2020 06:28:29.985	(levelValue=1000, timeMillis=1581604109985)
351	SEVERE	The SSL certificate has expired: [[Version: V3 Subject: CN=Class 3P Primary CA, O=Certplus, C=... (details)]	javax.enterprise.system.security.ssl	Feb 13, 2020 06:28:29.984	(levelValue=1000, timeMillis=1581604109984)
350	SEVERE	The SSL certificate has expired: [[Version: V3 Subject: CN=UTN-USERFirst-Object, OU=http://www... (details)]	javax.enterprise.system.security.ssl	Feb 13, 2020 06:28:29.984	(levelValue=1000, timeMillis=1581604109984)
349	SEVERE	The SSL certificate has expired: [[Version: V3 Subject: CN=Deutsche Telekom Root CA 2, OU=T... (details)]	javax.enterprise.system.security.ssl	Feb 13, 2020 06:28:29.983	(levelValue=1000, timeMillis=1581604109983)
348	SEVERE	The SSL certificate has expired: [[Version: V1 Subject: (details)]	javax.enterprise.system.security.ssl	Feb 13, 2020	(levelValue=1000, timeMillis=1581604109980)

Primero se consulta que la tarjeta sea la correcta:

The screenshot shows the Mozilla Firefox Log Entry Detail interface. The URL is <https://10.1.7.4:4848/common/logViewer/logEntryDetail.jsf?instanceName=server&logLevel=SEVERE&logFile=server.log&recNumber=363>. The log entry details are as follows:

- Timestamp: Feb 13, 2020 06:52:35.323
- Log Level: SEVERE
- Logger:
- Name-Value Pairs: (levelValue=1000, timeMillis=1581605555323)
- Record Number: 363
- Message ID:
- Complete Message: [directConnection=true] select * from tarjeta where numeroTarjeta='1111 2222 3333 4444' and titular='Jose Garcia' and validaDesde='11/09' and validaHasta='11/20' and codigoVerificacion='123'

Y después, si la tarjeta es correcta, se inserta el pago con la información aportada:



Ahora vamos a comprobar el pago en la página extendida con el ID del comercio anteriormente suministrado.

The screenshot shows a Mozilla Firefox window titled "Sistema de Pago con tarjeta - Mozilla Firefox". The address bar indicates the URL is 10.1.7.4:8080/P1/testbd.jsp. The page contains the following sections:

- Pago con tarjeta**
- Proceso de un pago**
 - Id Transacción:
 - Id Comercio:
 - Importe:
 - Numero de visa:
 - Titular:
 - Fecha Emisión:
 - Fecha Caducidad:
 - CVV2:
 - Modo debug: True False
 - Direct Connection: True False
 - Use Prepared: True False
- Consulta de pagos**
 - Id Comercio:
 -
- Borrado de pagos**
 - Id Comercio:
 -

Pago con tarjeta

Lista de pagos del comercio 10

idTransaccion	Importe	codRespuesta	idAutorizacion
1	100.0	000	3

[Volver al comercio](#)

Y vemos en el *log* qué se ha ejecutado:

Log Viewer Results (40)

Record Number	Log Level	Message	Logger	Timestamp	Name-Value Pairs
370	INFO	WebModule[null] ServletContext.log():[INFO] Acceso correcto./getpagos{details}	javax.enterprise.web	Feb 13, 2020 06:54:31.040	{levelValue=800, timeMillis=1581605671040}
369	INFO	WebModule[null] ServletContext.log():[INFO] Acceso correcto./testbd.jsp{details}	javax.enterprise.web	Feb 13, 2020 06:54:25.136	{levelValue=800, timeMillis=1581605665136}
368	INFO	WebModule[null] ServletContext.log():[INFO] Acceso correcto./testbd.jsp{details}	javax.enterprise.web	Feb 13, 2020 06:54:24.875	{levelValue=800, timeMillis=1581605664875}
367	INFO	WebModule[null] ServletContext.log():[INFO] Acceso correcto./testbd.jsp{details}	javax.enterprise.web	Feb 13, 2020 06:52:46.352	{levelValue=800, timeMillis=158160566352}
366	INFO	WebModule[null] ServletContext.log():[INFO] Acceso correcto./testbd.jsp{details}	javax.enterprise.web	Feb 13, 2020 06:52:45.458	{levelValue=800, timeMillis=1581605565458}
365	SEVERE	[directConnection=true] select idAutorizacion, codRespuesta from pago where idTransaccion = '1' ... {details}		Feb 13, 2020 06:52:35.327	{levelValue=1000, timeMillis=158160555327}
364	SEVERE	[directConnection=true] insert into pago(idTransaccion,importe,idComercio,numeroTarjeta) values ('1'... {details})		Feb 13, 2020 06:52:35.325	{levelValue=1000, timeMillis=158160555325}
363	SEVERE	[directConnection=true] select * from tarjeta where numeroTarjeta='1111 2222 3333 4444' and titular... {details}		Feb 13, 2020 06:52:35.323	{levelValue=1000, timeMillis=158160555323}
362	INFO	WebModule[null] ServletContext.log():[INFO] Acceso correcto./procesapago{details}	javax.enterprise.web	Feb 13, 2020 06:52:35.317	{levelValue=800, timeMillis=158160555317}
361	INFO	WebModule[null] ServletContext.log():[INFO] Acceso correcto./testbd.jsp{details}	javax.enterprise.web	Feb 13, 2020 06:52:10.514	{levelValue=800, timeMillis=158160553014}
360	INFO	Admin Console: Initializing Session Attributes...{details}	org.glassfish.admingui	Feb 13, 2020 06:28:31.461	{levelValue=800, timeMillis=1581604111461}
359	INFO	Redirecting to /index.jsf{details}	org.glassfish.admingui	Feb 13, 2020 06:28:31.362	{levelValue=800, timeMillis=1581604111362}
358	INFO	Listening to REST requests at context: /management/domain.{details}	javax.enterprise.admin.rest	Feb 13, 2020 06:28:31.349	{levelValue=800, timeMillis=1581604111349}
357	SEVERE	The SSL certificate has expired: [[Version: V3 Subject: CN=Entrust.net Secure Server Certificate... {details}]	javax.enterprise.system.security.ssl	Feb 13, 2020 06:28:29.987	{levelValue=1000, timeMillis=1581604109987}
356	SEVERE	The SSL certificate has expired: [[Version: V3 Subject: CN=Entrust.net Certification Authority... {details}]	javax.enterprise.system.security.ssl	Feb 13, 2020 06:28:29.986	{levelValue=1000, timeMillis=1581604109986}
355	SEVERE	The SSL certificate has expired: [[Version: V1 Subject: EMAILADDRESS=info@valcert.com, CN=ht... {details}]	javax.enterprise.system.security.ssl	Feb 13, 2020 06:28:29.986	{levelValue=1000, timeMillis=1581604109986}

Como podemos ver se ha realizado la consulta **getPagos** exitosamente, devolviendo los pagos realizados.

Actividades Navegador web Firefox jue 15:54 Log Entry Detail - Mozilla Firefox

Timestamp: Feb 13, 2020 06:28:31.461
Log Level: INFO
Logger: org.glassfish.admingui
Name-Value Pairs: {levelValue=800, timeMillis=1581604111461}
Record Number: 370
Message ID: Complete Message Admin Console: Initializing Session Attributes...

Finalmente, repetiremos este procedimiento para eliminar los pagos del comercio con el ID utilizado.

Sistema de Pago con tarjeta - Mozilla Firefox

10.1.7.4:8080/P1/testbd.jsp

Pago con tarjeta

Proceso de un pago

Id Transacción:
Id Comercio:
Importe:
Número de visa:
Titular:
Fecha Emisión:
Fecha Caducidad:
CVV2:
Modo debug: True False
Direct Connection: True False
Use Prepared: True False

Consulta de pagos

Id Comercio:

Borrado de pagos

Id Comercio: 10

Prácticas de Sistemas Informáticos II

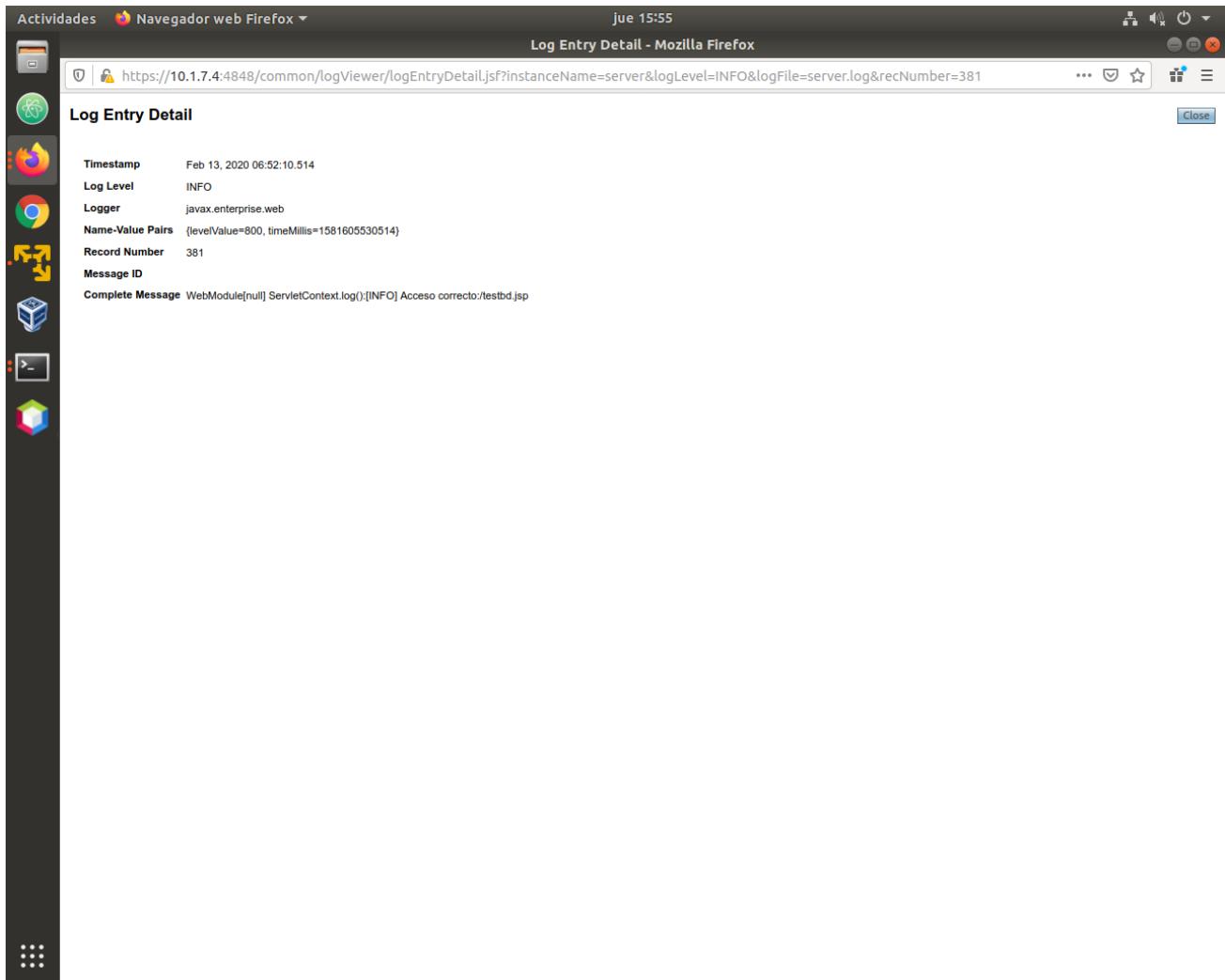
En la página extendida obtenemos un mensaje de eliminación exitosa.

The screenshot shows a Firefox browser window titled "Sistema de Pago con tarjeta - Mozilla Firefox". The address bar displays "10.1.7.4:8080/P1/delpagos". The main content area has a heading "Pago con tarjeta" and a message stating "Se han borrado 1 pagos correctamente para el comercio 10". Below this, there is a link "Volver al comercio". The browser's sidebar on the left shows various application icons.

Y en el *log* del servidor de aplicaciones también podemos ver que se ha ejecutado satisfactoriamente la consulta **delPagos**:

The screenshot shows a Firefox browser window titled "Log Viewer - Mozilla Firefox" with the URL "https://10.1.7.4:4848/common/logViewer/logViewer.jsp?instanceName=server&loglevel=INFO&viewResults=true". The main area displays log viewer settings and results. The "Log Viewer Results (40)" table shows several log entries, with two specific ones highlighted by a red box:

Record Number	Log Level	Message	Logger	Timestamp	Name-Value Pairs
382	INFO	WebModule[null] ServletContext.log():[INFO] Acceso correcto:/testbd.jsp(details)	javax.enterprise.web	Feb 13, 2020 06:55:07.781	{levelValue=800, timeMillis=1581605707781}
381	INFO	WebModule[null] ServletContext.log():[INFO] Acceso correcto:/delpagos(details)	javax.enterprise.web	Feb 13, 2020 06:55:05.035	{levelValue=800, timeMillis=1581605705035}
380	INFO	WebModule[null] ServletContext.log():[INFO] Acceso correcto:/getpagos(details)	javax.enterprise.web	Feb 13, 2020 06:54:31.040	{levelValue=800, timeMillis=1581605671040}
379	INFO	WebModule[null] ServletContext.log():[INFO] Acceso correcto:/testbd.jsp(details)	javax.enterprise.web	Feb 13, 2020 06:54:25.136	{levelValue=800, timeMillis=1581605665136}
378	INFO	WebModule[null] ServletContext.log():[INFO] Acceso correcto:/testbd.jsp(details)	javax.enterprise.web	Feb 13, 2020 06:54:24.875	{levelValue=800, timeMillis=1581605664875}
377	INFO	WebModule[null] ServletContext.log():[INFO] Acceso correcto:/testbd.jsp(details)	javax.enterprise.web	Feb 13, 2020 06:52:46.352	{levelValue=800, timeMillis=158160566352}
376	INFO	WebModule[null] ServletContext.log():[INFO] Acceso correcto:/testbd.jsp(details)	javax.enterprise.web	Feb 13, 2020 06:52:45.458	{levelValue=800, timeMillis=1581605654548}
375	SEVERE	[directConnection=true] select idAutorizacion, codRespuesta from pago where idTransaccion = '1' ... (details)		Feb 13, 2020 06:52:35.327	{levelValue=1000, timeMillis=1581605555327}
374	SEVERE	[directConnection=true] insert into pago(idTransaccion,importe,idComercio,numeroTarjeta) values ('1'... (details))		Feb 13, 2020 06:52:35.325	{levelValue=1000, timeMillis=1581605555325}
373	SEVERE	[directConnection=true] select * from tarjeta where numeroTarjeta='1111 2222 3333 4444' and titular... (details)		Feb 13, 2020 06:52:35.323	{levelValue=1000, timeMillis=1581605555323}
372	INFO	WebModule[null] ServletContext.log():[INFO] Acceso correcto:/procesapago(details)	javax.enterprise.web	Feb 13, 2020 06:52:35.317	{levelValue=800, timeMillis=1581605555317}
371	INFO	WebModule[null] ServletContext.log():[INFO] Acceso correcto:/testbd.jsp(details)	javax.enterprise.web	Feb 13, 2020 06:52:10.514	{levelValue=800, timeMillis=1581605530514}
370	INFO	Admin Console: Initializizing Session Attributes (details)	com.claudiocarrasco.adminui	Feb 13, 2020	{levelValue=800, timeMillis=1581605530514}



8 Ejercicio 6

Enunciado:

Realíicense 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()
 - isDebugEnabled() / setDebugEnabled() (Nota: VisaDAO.java contiene dos métodos setDebugEnabled que reciben distintos argumentos. Solo uno de ellos podrá ser exportado como servicio web).
 - isPrepared() / setPrepared()

De 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). En ningún caso se debe añadir ni modificar nada de la clase DBTester.

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.

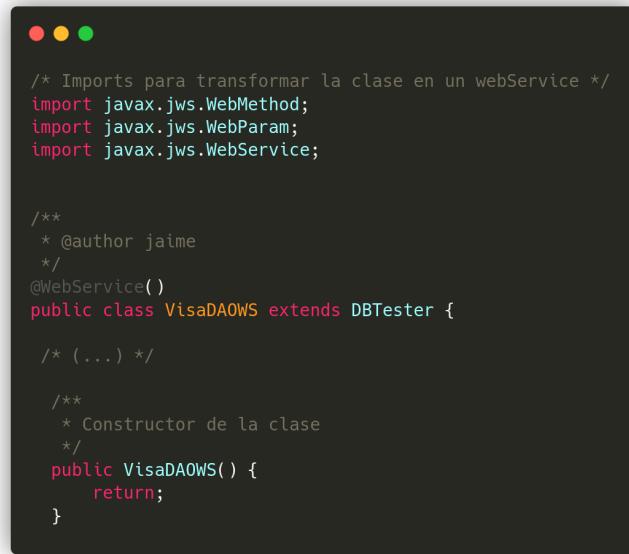
Incluye en la memoria cada fragmento de código donde se han ido añadiendo las modificaciones requeridas.

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 en lugar de un boolean?

Respuesta a la cuestión:

Para comenzar, añadimos los *imports* especificados en el enunciado en el fichero **VisaDAOWS.java**. A continuación cambiamos el nombre de la clase por el nuevo: **VisaDAOWS**



```
/* Imports para transformar la clase en un webService */
import javax.jws.WebMethod;
import javax.jws.WebParam;
import javax.jws.WebService;

/**
 * @author jaime
 */
@WebService()
public class VisaDAOWS extends DBTester {

    /* (...)

    /**
     * Constructor de la clase
     */
    public VisaDAOWS() {
        return;
    }
}
```

Después, tenemos que añadir las anotaciones que indican que implementamos un nuevo servicio web y sus métodos, así como cambiar los retornos de la función **realizaPago**, ya que ahora devuelve un **PagoBean** en lugar de un boolean/null.



```
@WebMethod(operationName = "compruebaTarjeta")
public boolean compruebaTarjeta( @WebParam(name = "tarjeta") TarjetaBean tarjeta) {
    // ...
}

@WebMethod(operationName = "realizaPago")
public synchronized PagoBean realizaPago(@WebParam(name = "pago") PagoBean pago) {
    // ...
    ret = null;
    if (!pst.execute()
        && pstmt.executeUpdate() == 1) {
        ret = pago;
    }
    // Cambiamos todos los ret = false por ret = null y ret = true por ret = pago.
    // ...
}

@WebMethod(operationName = "isPrepared")
public boolean isPrepared() {
    //...
}

@WebMethod(operationName = "setPrepared")
public void setPrepared(boolean prepared) {
    //...
}

@WebMethod(operationName = "isDebugEnabled")
public boolean isDebugEnabled() {
    //...
}

@WebMethod(operationName = "setDebugEnabled")
public void setDebugEnabled(boolean debug) {
    //...
}

@WebMethod(exclude = true)
public void setDebugEnabled(String debug) {
    //...
}

@WebMethod(operationName = "isDirectConnection")
@Override
public boolean isDirectConnection() {
    return super.isDirectConnection();
}

@WebMethod(operationName = "setDirectConnection")
@Override
public void setDirectConnection(@WebParam(name = "directConnection") boolean directConnection) {
    super.setDirectConnection(directConnection);
}
```

9 Ejercicio 7

Enunciado:

Despliegue el servicio con la regla correspondiente en el build.xml. Acceda al WSDL remotalemente 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:

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

Respuesta a la cuestión:

Desplegamos el servicio utilizando los comandos aportados en el fichero build.xml. Ahora accedemos a la página de administración en <http://10.1.7.1:4848> y en 'Applications' podemos encontrar P1-ws-ws:

Select	Name	Deployment Order	Enabled	Engines	Action
	P1	100	✓	web	Launch Redeploy Reload
	P1-ws-ws	100	✓	webservices, web	Launch Redeploy Reload

Pinchando en P1-ws-ws y en 'View Endpoint' podemos ver la información del web service

The screenshot shows the GlassFish Server Open Source Edition application management interface. The left sidebar lists various application components like Domain, Clusters, Instances, Nodes, Applications, and Resources. Under Applications, 'P1-ws-ws' is selected. The main content area displays 'Web Service Endpoint Information' for the 'VisaDAOService'. Key details include:

- Application Name:** P1-ws-ws
- Tester:** /P1-ws-ws/VisaDAOService?Tester
- WSDL:** /P1-ws-ws/VisaDAOService?wsdl
- Endpoint Name:** VisaDAOWS
- Service Name:** VisaDAOService
- Port Name:** VisaDAOSService
- Deployment Type:** 109
- Implementation Type:** SERVLET
- Implementation Class Name:** ssil2.visa.dao.VisaDAOWS
- Endpoint Address URI:** /P1-ws-ws/VisaDAOService
- Namespace:** http://dao.visa.ssil2/

Desde aquí podemos acceder al WSDL de la aplicación pinchando en el tercer enlace que nos aparece (`/P1-ws-ws/VisaDAOSService?wsdl`, recuerde que la URL contiene la dirección IP de la máquina virtual donde se encuentra el servidor de aplicaciones).

Nos debería aparecer el siguiente fichero:

The screenshot shows a browser window displaying the XML content of the WSDL file. The URL is `10.1.7.1:8080/P1-ws-ws/VisaDAOSService?wsdl`. The XML code is as follows:

```

<!-- Published by JAX-WS RI (http://jax-ws.java.net). RI's version is Metro/2.3.2-b608 (trunk-7970; 2015-01-21T12:50:19+0000) JAXWS-RI/2.2.11-b150120.1832 JAXWS-API/2.2.12 JAXB-RI/2.2.12-b141219.163 -->
<!-- Generated by JAX-WS RI (http://jax-ws.java.net). RI's version is Metro/2.3.2-b608 (trunk-7970; 2015-01-21T12:50:19+0000) JAXWS-RI/2.2.11-b150120.1832 JAXWS-API/2.2.12 JAXB-RI/2.2.12-b141219.163 -->
<definitions targetNamespace="http://dao.visa.ssil2/" name="VisaDAOSService">
  <types>
    <xsd:schema>
      <xsd:import namespace="http://dao.visa.ssil2/" schemaLocation="http://10.1.7.1:8080/P1-ws-ws/VisaDAOSService?xsd=1"/>
    </xsd:schema>
  </types>
  <message name="delPagos">
    <part name="parameters" element="tns:delPagos"/>
  </message>
  <message name="delPagosResponse">
    <part name="parameters" element="tns:delPagosResponse"/>
  </message>
  <message name="setPrepared">
    <part name="parameters" element="tns:setPrepared"/>
  </message>
  <message name="setPreparedResponse">
    <part name="parameters" element="tns:setPreparedResponse"/>
  </message>
  <message name="getPagos">
    <part name="parameters" element="tns:getPagos"/>
  </message>
  <message name="getPagosResponse">
    <part name="parameters" element="tns:getPagosResponse"/>
  </message>
  <message name="errorLog">
    <part name="parameters" element="tns:errorLog"/>
  </message>
  <message name="errorLogResponse">
    <part name="parameters" element="tns:errorLogResponse"/>
  </message>
  <message name="compruebaTarjeta">
  </message>

```

Con la información recogida en este fichero podremos contestar a las preguntas de este ejercicio.

En primer lugar se nos pregunta en qué fichero están definidos los tipos de datos intercambiados con el webservice. Este fichero es el aportado en la etiqueta `xxsd:import namespace=...` `schemaLocation= ...`, para ser más exactos, es el enlace situado en 'schemaLocation'. Los tipos de datos predefinidos que se usan se pueden observar en este último fichero, bajo la etiqueta `xs:element ... type="xs:..."`

```

<xs:element name="arg0" type="xs:string" minOccurs="0"/>
</xs:sequence>
</xs:complexType>
<-<xs:complexType name="errorLogResponse">
<-<xs:sequence/>
</xs:complexType>
<-<xs:complexType name="compruebaTarjeta">
<-<xs:sequence>
<-<xs:element name="tarjeta" type="tns:tarjetaBean" minOccurs="0"/>
<-<xs:sequence/>
</xs:complexType>
<-<xs:complexType name="tarjetaBean">
<-<xs:sequence>
<-<xs:element name="codigoVerificacion" type="xs:string" minOccurs="0"/>
<-<xs:element name="fechaCaducidad" type="xs:string" minOccurs="0"/>
<-<xs:element name="fechaEmision" type="xs:string" minOccurs="0"/>
<-<xs:element name="numero" type="xs:string" minOccurs="0"/>
<-<xs:element name="titular" type="xs:string" minOccurs="0"/>
<-<xs:sequence>
</xs:complexType>
<-<xs:complexType name="compruebaTarjetaResponse">
<-<xs:sequence>
<-<xs:element name="return" type="xs:boolean"/>
<-<xs:sequence>
</xs:complexType>
<-<xs:complexType name="setDebug">
<-<xs:sequence>
<-<xs:element name="arg0" type="xs:boolean"/>
<-<xs:sequence>
</xs:complexType>
<-<xs:complexType name="setDebugResponse">
<-<xs:sequence>
</xs:complexType>
<-<xs:complexType name="realizaPago">
<-<xs:sequence>
<-<xs:element name=" pago" type="tns:pagoBean" minOccurs="0"/>
<-<xs:sequence>
</xs:complexType>
</xs:complexType>

```

También podemos nosotros definir nuestros propios tipos de datos, bajo la etiqueta **xs:element** ... **type="tns:..."**. Esto también se puede ver en el fichero del cual estábamos hablando.

```

<!-- Published by JAX-WS RI (http://jax-ws.java.net). RI's version is Metro/2.3.2-b609 (trunk-7979; 2015-01-21T12:58:19+0000) JAXWS-RI/2.2.11-b150120.1832 JAXWS-API/2.2.12 JAXB-RI/2.2.12-b141219.1637 -->
<-<xss:schema version="1.0" targetNamespace="http://dao.visa.ssif2/">
<-<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="getPagos" type="tns:getPagos"/>
<-<xs:element name="getPagosResponse" type="tns:getPagosResponse"/>
<-<xs:element name="isDebugEnabled" type="tns:isDebugEnabled"/>
<-<xs:element name="isDebugEnabledResponse" type="tns:isDebugEnabledResponse"/>
<-<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="setDebugEnabled" type="tns:setDebugEnabled"/>
<-<xs:element name="setDebugEnabledResponse" type="tns:setDebugEnabledResponse"/>
<-<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="errorLog">
<-<xs:sequence>
<-<xs:element name="arg0" type="xs:string" minOccurs="0"/>
<-<xs:sequence>
</xs:complexType>
<-<xs:complexType name="errorLogResponse">
<-<xs:sequence/>
</xs:complexType>
<-<xs:complexType name="compruebaTarjeta">
<-<xs:sequence>

```

Ahora se nos pregunta por la etiqueta que está asociada a los métodos invocados en el webservice. Esta es la etiqueta **operation**:

```

<message name="setDebug">
  <part name="parameters" element="tns:setDebug"/>
</message>
<message name="setDebugResponse">
  <part name="parameters" element="tns:setDebugResponse"/>
</message>
<portType name="VisaDAOWS">
  <operation name="delPagos">
    <input wsam:Action="http://dao.visa.ssl2/VisaDAOWS/delPagosRequest" message="tns:delPagos"/>
    <output wsam:Action="http://dao.visa.ssl2/VisaDAOWS/delPagosResponse" message="tns:delPagosResponse"/>
  </operation>
  <operation name="setPrepared">
    <input wsam:Action="http://dao.visa.ssl2/VisaDAOWS/setPreparedRequest" message="tns:setPrepared"/>
    <output wsam:Action="http://dao.visa.ssl2/VisaDAOWS/setPreparedResponse" message="tns:setPreparedResponse"/>
  </operation>
  <operation name="getPagos">
    <input wsam:Action="http://dao.visa.ssl2/VisaDAOWS/getPagosRequest" message="tns:getPagos"/>
    <output wsam:Action="http://dao.visa.ssl2/VisaDAOWS/getPagosResponse" message="tns:getPagosResponse"/>
  </operation>
  <operation name="errorLog">
    <input wsam:Action="http://dao.visa.ssl2/VisaDAOWS/errorLogRequest" message="tns:errorLog"/>
    <output wsam:Action="http://dao.visa.ssl2/VisaDAOWS/errorLogResponse" message="tns:errorLogResponse"/>
  </operation>
  <operation name="compruebaTarjeta">
    <input wsam:Action="http://dao.visa.ssl2/VisaDAOWS/compruebaTarjetaRequest" message="tns:compruebaTarjeta"/>
    <output wsam:Action="http://dao.visa.ssl2/VisaDAOWS/compruebaTarjetaResponse" message="tns:compruebaTarjetaResponse"/>
  </operation>
  <operation name="setDirectConnection">
    <input wsam:Action="http://dao.visa.ssl2/VisaDAOWS/setDirectConnectionRequest" message="tns:setDirectConnection"/>
    <output wsam:Action="http://dao.visa.ssl2/VisaDAOWS/setDirectConnectionResponse" message="tns:setDirectConnectionResponse"/>
  </operation>
  <operation name="isDirectConnection">
    <input wsam:Action="http://dao.visa.ssl2/VisaDAOWS/isDirectConnectionRequest" message="tns:isDirectConnection"/>
    <output wsam:Action="http://dao.visa.ssl2/VisaDAOWS/isDirectConnectionResponse" message="tns:isDirectConnectionResponse"/>
  </operation>
  <operation name="realizaPago">
    <input wsam:Action="http://dao.visa.ssl2/VisaDAOWS/realizaPagoRequest" message="tns:realizaPago"/>
    <input wsam:Action="http://dao.visa.ssl2/VisaDAOWS/realizaPagoResponse" message="tns:realizaPagoResponse"/>
  </operation>
</portType>

```

Y también se pide la etiqueta que describe los mensajes intercambiados en la invocación de los métodos del webservice. Esta es **message**:

```

<message name="isPrepared">
  <part name="parameters" element="tns:isPrepared"/>
</message>
<message name="isPreparedResponse">
  <part name="parameters" element="tns:isPreparedResponse"/>
</message>
<message name="isDebugEnabled">
  <part name="parameters" element="tns:isDebugEnabled"/>
</message>
<message name="isDebugEnabledResponse">
  <part name="parameters" element="tns:isDebugEnabledResponse"/>
</message>
<message name="setDebugEnabled">
  <part name="parameters" element="tns:setDebugEnabled"/>
</message>
<message name="setDebugEnabledResponse">
  <part name="parameters" element="tns:setDebugEnabledResponse"/>
</message>
<portType name="VisaDAOWS">
  <operation name="delPagos">
    <input wsam:Action="http://dao.visa.ssl2/VisaDAOWS/delPagosRequest" message="tns:delPagos"/>
    <output wsam:Action="http://dao.visa.ssl2/VisaDAOWS/delPagosResponse" message="tns:delPagosResponse"/>
  </operation>
  <operation name="setPrepared">
    <input wsam:Action="http://dao.visa.ssl2/VisaDAOWS/setPreparedRequest" message="tns:setPrepared"/>
    <output wsam:Action="http://dao.visa.ssl2/VisaDAOWS/setPreparedResponse" message="tns:setPreparedResponse"/>
  </operation>
  <operation name="getPagos">
    <input wsam:Action="http://dao.visa.ssl2/VisaDAOWS/getPagosRequest" message="tns:getPagos"/>
    <output wsam:Action="http://dao.visa.ssl2/VisaDAOWS/getPagosResponse" message="tns:getPagosResponse"/>
  </operation>
  <operation name="errorLog">
    <input wsam:Action="http://dao.visa.ssl2/VisaDAOWS/errorLogRequest" message="tns:errorLog"/>
    <output wsam:Action="http://dao.visa.ssl2/VisaDAOWS/errorLogResponse" message="tns:errorLogResponse"/>
  </operation>
  <operation name="compruebaTarjeta">
    <input wsam:Action="http://dao.visa.ssl2/VisaDAOWS/compruebaTarjetaRequest" message="tns:compruebaTarjeta"/>
    <output wsam:Action="http://dao.visa.ssl2/VisaDAOWS/compruebaTarjetaResponse" message="tns:compruebaTarjetaResponse"/>
  </operation>
</portType>

```

Como podemos observar, la etiqueta **message** también sirve para especificar los tipos de parámetros de entrada y salida de las operaciones entre el cliente y el servidor de aplicaciones.

Finalmente, se nos pregunta por la etiqueta que especifica el protocolo de comunicación con el webservice, y la etiqueta que especifica la URL a la que se deberá conectar un cliente para acceder al webservice. La primera es la etiqueta **soap:binding** y la segunda es la etiqueta **soap:address**:

Aplicaciones

dom 23 de feb 15:07

Web Service Endpoint Information - Firefox Nightly

10.1.7.1:8080/P1-ws-ws/VisaDAO

User: admin Domain: domain

GlassFish Server Open

Common Tasks

- Domain
- server (Admin Server)
- Clusters
- Standalone Instances
- Nodes
- Applications
- P1
- P1-ws-ws
- Lifecycle Modules
- Monitoring Data
- Resources
- Concurrent Resources
- Connectors
- JDBC
- JMS Resources
- JNDI
- JavaMail Sessions
- Resource Adapter Con
- Configurations
- default-config
- server-config
- Update Tool

```

<input wsam:Action="http://dao.visa.ssl2/VisaDAOWS/isPreparedRequest" message="tns:isPrepared"/>
<output wsam:Action="http://dao.visa.ssl2/VisaDAOWS/isPreparedResponse" message="tns:isPreparedResponse"/>
</operation>
<operation name="isDebug">
<input wsam:Action="http://dao.visa.ssl2/VisaDAOWS/isDebugRequest" message="tns:isDebug"/>
<output wsam:Action="http://dao.visa.ssl2/VisaDAOWS/isDebugResponse" message="tns:isDebugResponse"/>
</operation>
<operation name="setDebug">
<input wsam:Action="http://dao.visa.ssl2/VisaDAOWS/setDebugRequest" message="tns:setDebug"/>
<output wsam:Action="http://dao.visa.ssl2/VisaDAOWS/setDebugResponse" message="tns:setDebugResponse"/>
</operation>
</portType>
<binding name="VisaDAOSSPortBinding" type="tns:VisaDAOWS">
<soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
<operation name="delPagos">
<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" />
</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>

```

soap:binding

Resaltar todo Coincidencia de mayúsculas/minúsculas Coincijir diacríticos Palabras completas 1 de 1 abierto

Aplicaciones

dom 23 de feb 15:07

Web Service Endpoint Information - Firefox Nightly

10.1.7.1:8080/P1-ws-ws/VisaDAO

User: admin Domain: domain

GlassFish Server Open

Common Tasks

- Domain
- server (Admin Server)
- Clusters
- Standalone Instances
- Nodes
- Applications
- P1
- P1-ws-ws
- Lifecycle Modules
- Monitoring Data
- Resources
- Concurrent Resources
- Connectors
- JDBC
- JMS Resources
- JNDI
- JavaMail Sessions
- Resource Adapter Con
- Configurations
- default-config
- server-config
- Update Tool

```

<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="isDebug">
<soap:operation soapAction="" />
<input>
<soap:body use="literal" />
</input>
<output>
<soap:body use="literal" />
</output>
</operation>
<operation name="setDebug">
<soap:operation soapAction="" />
<input>
<soap:body use="literal" />
</input>
<output>
<soap:body use="literal" />
</output>
</operation>
</binding>
</service>
</definitions>
<service name="VisaDAOSService">
<port name="VisaDAOSSPort" binding="tns:VisaDAOSSPortBinding">
<soap:address location="http://10.1.7.1:8080/P1-ws-ws/VisaDAOSService"/>
</port>
</service>
</definitions>

```

soap:address

Resaltar todo Coincidencia de mayúsculas/minúsculas Coincijir diacríticos Palabras completas 1 de 1 abierto

10 Ejercicio 8

Enunciado:

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

Respuesta a la cuestión:

Como se nos ha explicado, realizamos los cambios necesarios en el fichero **ProcesaPago.java**: añadimos los *imports* especificados, realizamos la instanciación de la clase remota en dos pasos y tenemos en cuenta las posibles nuevas excepciones y que realizaPago ahora devuelve un PagoBean en lugar de un boolean/null.

```
// ProcesaPago.java
// (...)
// import ssii2.visa.dao.VisaDAO;

/* Imports para instanciar un stub */
import ssii2.visa.VisaDAOWSService;
import ssii2.visa.VisaDAOWS;
import javax.xml.ws.WebServiceRef;

// (...)
VisaDAOWSService service = null;
VisaDAOWS dao = null;

try {
    service = new VisaDAOWSService();
    dao = service.getVisaDAOWebService();
}

} catch (Exception e) {
    enviaError(e, request, response);
    return;
}

// (...)
if ((pago = dao.realizaPago(pago)) == null) {
    enviaError(new Exception("Pago incorrecto"), request, response);
    return;
}
```

11 Ejercicio 9

Enunciado:

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.

Respuesta a la cuestión:

Modificamos primero el fichero **web.xml** para añadir un parámetro de inicialización tal y como se nos indica en el apéndice 15.1:

```
<!-- web.xml -->

<context-param>
    <param-name>pathVisaWS</param-name>
    <param-value>http://10.1.7.1:8080/P1-ws-ws/VisaDAOSService</param-value>
</context-param>
```

Finalmente, modificamos el fichero **ProcesaPago.java** para actualizar la llamada al servicio para que la ruta al servicio remoto se obtenga del fichero de configuración **web.xml**

```
// ProcesaPago.java

// (...)

/* Imports para instanciar un stub */
import ssii2.visa.VisaDAOSService;
import ssii2.visa.VisaDAOWS;
import javax.xml.ws.WebServiceRef;
import javax.xml.ws.BindingProvider;

// (...)

VisaDAOSService service = null;
VisaDAOWS dao = null;
BindingProvider bp = null;

try {
    service = new VisaDAOSService();
    dao = service.getVisaDAOSSPort();

    bp = (BindingProvider) dao;
    bp.getRequestContext().put(BindingProvider.ENDPOINT_ADDRESS_PROPERTY,
getServletContext().getInitParameter("pathVisaWS"));
} catch (Exception e) {
    enviaError(e, request, response);
    return;
}
```

12 Ejercicio 10

Enunciado:

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.

Respuesta a la cuestión:

Realizamos cambios muy parecidos a los hechos en el fichero ProcesaPago.java en los ficheros VisaDAO.java, DelPagos.java y GetPagos.java.

Empezamos cambiando el retorno de la función getPagos(...) del fichero VisaDAO.java. Además añadimos los decoradores @WebMethod y @WebParam en los métodos getPagos y delPagos del mismo fichero.

```
// VisaDAO.java
@WebMethod(operationName = "getPagos")
public ArrayList<PagoBean> getPagos( @WebParam(name = "idComercio") String idComercio) {
    // ...
}

// PagoBean[] ret = null;
// ...

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

return pagos;
}

@WebMethod(operationName = "delPagos")
public int delPagos( @WebParam(name = "idComercio") String idComercio) {
    // ...
}
```

Por otro lado, actualizamos los *imports* en los ficheros **DelPagos.java** y **GetPagos.java**. También modificamos la ruta del servidor remoto sin cambiar la definición del servicio como en el ejercicio 9:

```
// DelPagos.java

// ...
// imports para acceder al WS remoto
import ssii2.visa.VisaDAOSService;
import ssii2.visa.VisaDAO;
import javax.xml.ws.WebServiceRef;
import javax.xml.ws.BindingProvider;

// ...
VisaDAOSService service = null;
VisaDAO dao = null;
BindingProvider bp = null;

try {
    service = new VisaDAOSService();
    dao = service.getVisaDAOSPort();

    bp = (BindingProvider) dao;
    bp.getRequestContext().put(BindingProvider.ENDPOINT_ADDRESS_PROPERTY,
        getServletContext().getInitParameter("pathVisaWS"));
} catch (Exception e) {
    enviaError(e, request, response);
    return;
}
```

```

// GetPagos.java

// (...)
// imports para usar Arrays
import java.util.ArrayList;
import java.util.List;
// imports para acceder al WS remoto
import ssii2.visa.VisaDAOService;
import ssii2.visa.VisaDAO;
import javax.xml.ws.WebServiceRef;
import javax.xml.ws.BindingProvider;

// (...)
VisaDAOSService service = null;
VisaDAO dao = null;
BindingProvider bp = null;

try {
    service = new VisaDAOSService();
    dao = service.getVisaDAOSPort();

    bp = (BindingProvider) dao;
    bp.getRequestContext().put(BindingProvider.ENDPOINT_ADDRESS_PROPERTY,
getServletContext().getInitParameter("pathVisaWS"));
} catch (Exception e) {
    enviaError(e, request, response);
    return;
}

// (...)
// Obtención de pagos desde el webService y su casting a ArrayList desde List
List<PagoBean> pagosRet = dao.getPagos(idComercio);
ArrayList<PagoBean> pagos = new ArrayList<PagoBean>(pagosRet);

/* Casting a PagoBean[] para la vista .jsp */
PagoBean[] pagosArr = null;
pagosArr = new PagoBean[pagos.size()];
pagosArr = pagos.toArray(pagosArr);

```

13 Ejercicio 11

Enunciado:

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.

Respuesta a la cuestión:

Se han requerido ejecutar los siguientes comandos:

Primero **ant generar-stubs**, para crear los directorios y generar el .jar con el *manifest*.

Segundo **wsimport -d build/client/WEB-INF/classes -p ssii2.visa http://10.1.7.1:8080/P1-ws-ws/VisaDAOSService?wsdl**, que crea las clases del servicio necesarias en cliente, a partir del fichero WSDL definido por el servidor. Las clases generadas son las siguientes:

- CompruebaTarjeta.class
- ErrorLogResponse.class
- IsDirectConnection.class
- package-info.class x
- SetDebugResponse.class
- TarjetaBean.class x
- CompruebaTarjetaResponse.class
- GetPagos.class

- IsDirectConnectionResponse.class
- PagoBean.class x
- SetDirectConnection.class
- VisaDAOWS.class x
- DelPagos.class
- GetPagosResponse.class
- IsPrepared.class
- RealizaPago.class
- SetDirectConnectionResponse.class
- VisaDAOWSService.class x
- DelPagosResponse.class
- IsDebug.class
- IsPreparedResponse.class
- RealizaPagoResponse.class
- SetPrepared.class
- ErrorLog.class
- IsDebugEnabled.class
- ObjectFactory.class x
- SetDebugEnabled.class
- SetPreparedResponse.class

14 Ejercicio 12

Enunciado:

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

Respuesta a la cuestión:

Simplemente utilizamos la funcionalidad `exec` en el fichero **build.xml** en el *target generar-stubs*:

```

<!-- web.xml -->

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

```

15 Ejercicio 13

Enunciado:

- 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

Respuesta a la cuestión:

Cambiamos las variables de entorno del fichero **build.properties**, de tal forma que **as.host.client** pasa a valer **10.1.7.2** y **as.host.server** sigue siendo **10.1.7.1**, igual que la base de datos. A continuación compilamos, empaquetamos y desplegamos el cliente (**ant cliente**). Ahora podemos acceder a la funcionalidad del servidor de aplicaciones desde la página <http://10.1.7.2:8080/P1-ws-cliente> e intentar realizar un pago:



Sistema de Pago con tarjeta - Firefox Nightly

dom 23 de feb 18:29

[Sistema de Pago con tarjeta - Firefox Nightly](#)

10.1.7.2:8080/P1-ws-cliente/comenzapago

Pago con tarjeta

Número de visa: 1111 2222 3333 4444
 Titular: José García
 Fecha Emisión: 11/09
 Fecha Caducidad: 11/20
 CVV2: 123

Id Transacción: 1
 Id Comercio: 10
 Importe: 100.0

Prácticas de Sistemas Informáticos II

Como veamos a continuación, el pago se ha realizado éxitosamente.

Ahora vamos a hacer lo mismo, pero desde la página de *debug*:

<http://10.1.7.2/P1-ws-cliente/testbd.jsp>.

Empezamos probando la funcionalidad de consulta de pagos realizados:

Sistema de Pago con tarjeta - Firefox Nightly

dom 23 de feb 18:30

[Sistema de Pago con tarjeta - Firefox Nightly](#)

10.1.7.2:8080/P1-ws-cliente/testbd.jsp

Pago con tarjeta

Proceso de un pago

Id Transacción:
 Id Comercio:
 Importe:
 Número de visa:
 Titular:
 Fecha Emisión:
 Fecha Caducidad:
 CVV2:
 Modo debug: True False
 Direct Connection: True False
 Use Prepared: True False

Consulta de pagos

Id Comercio: 10

Borrado de pagos

Id Comercio:

Prácticas de Sistemas Informáticos II

Sistema de Pago con tarjeta - Firefox Nightly

dom 23 de feb 18:30

[Sistema de Pago con tarjeta - Firefox Nightly](#)

10.1.7.2:8080/P1-ws-cliente/getpagos

Pago con tarjeta

Lista de pagos del comercio 10

ID Transacción	Importe	Cod Respuesta	ID Autorización
1	100.0	000	2

[Volver al comercio](#)

Prácticas de Sistemas Informáticos II

También aprovechamos y probamos la opción de borrado de pagos:

The screenshot shows a Firefox browser window with the title "Sistema de Pago con tarjeta - Firefox Nightly". The URL is 10.1.2.8080/P1-ws-cliente/testbd.jsp. The page contains two main sections: "Pago con tarjeta" and "Borrado de pagos".

Pago con tarjeta

Proceso de un pago

Form fields for payment details:

- Id Transacción: [input field]
- Id Comercio: [input field]
- Importe: [input field]
- Número de visa: [input field]
- Titular: [input field]
- Fecha Emisión: [input field]
- Fecha Caducidad: [input field]
- CVV2: [input field]

Boolean checkboxes:

- Modo debug: True False
- Direct Connection: True False
- Use Prepared: True False

Consulta de pagos

Form field:

Borrado de pagos

Form fields:

Prácticas de Sistemas Informáticos II

Ahora volvemos a probar que nuestro servicio funciona correctamente, pero esta vez desde esta página:

The screenshot shows a Firefox browser window with the title "Sistema de Pago con tarjeta - Firefox Nightly". The URL is 10.1.2.8080/P1-ws-cliente/delpagos. The page displays a success message after a payment has been deleted.

Pago con tarjeta

Se han borrado 1 pagos correctamente para el comercio 10

[Volver al comercio](#)

Prácticas de Sistemas Informáticos II

Sistema de Pago con tarjeta - Firefox Nightly

dom 23 de feb 18:31

10.1.7.2:8080/P1-ws-cliente/testbd.jsp

Pago con tarjeta

Proceso de un pago

Id Transacción:	1
Id Comercio:	10
Importe:	100
Número de visa:	1111 2222 3333 4444
Titular:	Jose Garcia
Fecha Emisión:	11/09
Fecha Caducidad:	11/20
CVV2:	123
Modo debug:	<input type="radio"/> True <input type="radio"/> False
Direct Connection:	<input type="radio"/> True <input type="radio"/> False
Use Prepared:	<input type="radio"/> True <input type="radio"/> False
<input type="button" value="Pagar"/>	

Consulta de pagos

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

Borrado de pagos

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

Prácticas de Sistemas Informáticos II

Sistema de Pago con tarjeta - Firefox Nightly

dom 23 de feb 18:32

10.1.7.2:8080/P1-ws-cliente/procesapago

Pago con tarjeta

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

```
idTransaccion: 1
idcomercio: 10
importe: 100.0
codRespuesta: 000
idAutorizacion: 3
```

[Volver al comercio](#)

Prácticas de Sistemas Informáticos II

Todo parece haber ido según lo esperado. Comprobemos esto último directamente desde la base de datos utilizando *ssh*:

```
ssh si2@10.1.7.1
+-----+
| P1-ws:alum | ssh si2@10.1.7.1 | ssh si2@10.1.7.2 |
+-----+
si2@si2serv01:~$ psql -U alumnodeb -d visa
psql (8.4.10)
Type "help" for help.

visa=# SELECT * FROM pago;
   _idautorizacion | _idtransaccion | codrespuesta | importe | _idcomercio | numerotarjeta | fecha
   +-----+-----+-----+-----+-----+-----+-----+
   3 | 1 | 000 | 100 | 10 | 1111 2222 3333 4444 | 2020-02-23 09:32:86.710827
(1 row)

visa=#
```

16 Cuestiones

16.1 Cuestión 1

Enunciado:

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.

Respuesta:

Se introduce el pago en `pago.html`, se envía y se pasa por el servlet `ComienzaPago.java`. Aquí se comprueba que los datos del formulario de `pago.html`, sean correctos y se sirve al cliente web el fichero jsp `formdatosvisa.jsp`. Al llenar los datos y enviarlos, se lanza el servlet `ProcesaPago.java` donde se comprueba que la fecha ha expirado (antes de comprobar en la base de datos que la tarjeta sea o no correcta), entonces se redirige a `formdatosvisa.jsp` indicando que la fecha es incorrecta.

16.2 Cuestión 2

Enunciado:

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

Respuesta:

El servlet `ComienzaPago.java` sirve `formdatosvisa.jsp`, que es necesario para solicitar la información sobre el pago con tarjeta. Al enviar este formulario se invoca al servlet `ProcesaPago.java` que es el encargado de procesarla con diversas llamadas a `VisaDAO`.

16.3 Cuestión 3

Enunciado:

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?

Respuesta:

`pago.html` contiene la información para solicitar los datos del pago (id, idComercio e importe). Estos son procesados por el servlet `ComienzaPago.java`. Esta información se almacena en la sesión (en una instancia de `PagoBean`), posteriormente, se sirve el jsp `formdatosvisa.jsp` que solicita la información de la tarjeta. Una vez se envía este formulario, el servlet `ProcesaPago.java` se encarga de procesar los datos de la tarjeta (almacenandolos en un `TarjetaBean`) y, si los datos son correctos, comparte estos datos y el objeto `PagoBean` con `VisaDAO`. `VisaDAO` tiene acceso a los datos del pago y de la tarjeta. Tras realizar la comprobación de que la tarjeta esté autorizada, este módulo se pone en contacto con la base de datos PSQL y almacena la información del pago.

16.4 Cuestión 4

Enunciado:

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?

Respuesta:

Cuando se utiliza la página de pruebas extendida `testbd.jsp`, accedemos directamente al servlet `ProcesaPago.java`, de esta forma evitamos invocar `ComienzaPago.java` y no se nos sirve `formdatostarjeta.jsp`.

Sigue funcionando el pago ya que la única diferencia es que `ComienzaPago.java` instancia un `PagoBean` con la información del pago. Como ahora no pasamos por `ComienzaPago`,

en ProcesaPago existe un control para comprobar que la variable de sesión con la información del pago sea distinta de `null`. Si no está inicializado (en caso de que lleguemos por `testbd.jsp`), lo inicializa con la información solicitada.

17 Conclusiones

El objetivo fundamental de esta práctica era adentrarse en la arquitectura de JAVA EE desde el punto de vista del arquitecto de software. Adicionalmente, hemos alcanzado los siguientes subobjetivos:

- Experimentar con un sistema multicapa (multitier) de varios niveles: interfaz cliente, aplicación servidora y base de datos. La aplicación servidora la subdividimos en varios niveles según el proyecto adquiera más complejidad.
- Introducir la aplicación de ejemplo que emplearemos a lo largo de todas las prácticas: Aplicación VISA para venta electrónica. Esta aplicación hace uso de JSP, Servlets y JavaBeans.
- Conocer JDBC como API de acceso a base de datos.
- Conocer y experimentar con la tecnología de publicación de Servicios Web o Web Services
- Automatización de tareas de construcción y despliegue con la herramienta ant.

Ahora avanzamos a la segunda parte de esta práctica, donde podremos continuar profundizando en elementos de la arquitectura JAVA EE.

18 Bibliografía

- Java EE 7 Tutorial
- API Java EE 7
- Documentación P1A Moodle