|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | Escuela Politécnica Superior  Ingeniería Informática  Prácticas de Sistemas Informáticos 2 | | | |
| **Grupo** | **2312** | **Práctica** | 1 | **Fecha** | 17/02/2015 |
| **Alumno/a** | | García, Teodoro, Roberto | | | |
| **Alumno/a** | | García, Roqué, Mario Valdemaro | | | |

# Práctica M: Título

## Cuestión número 1:

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

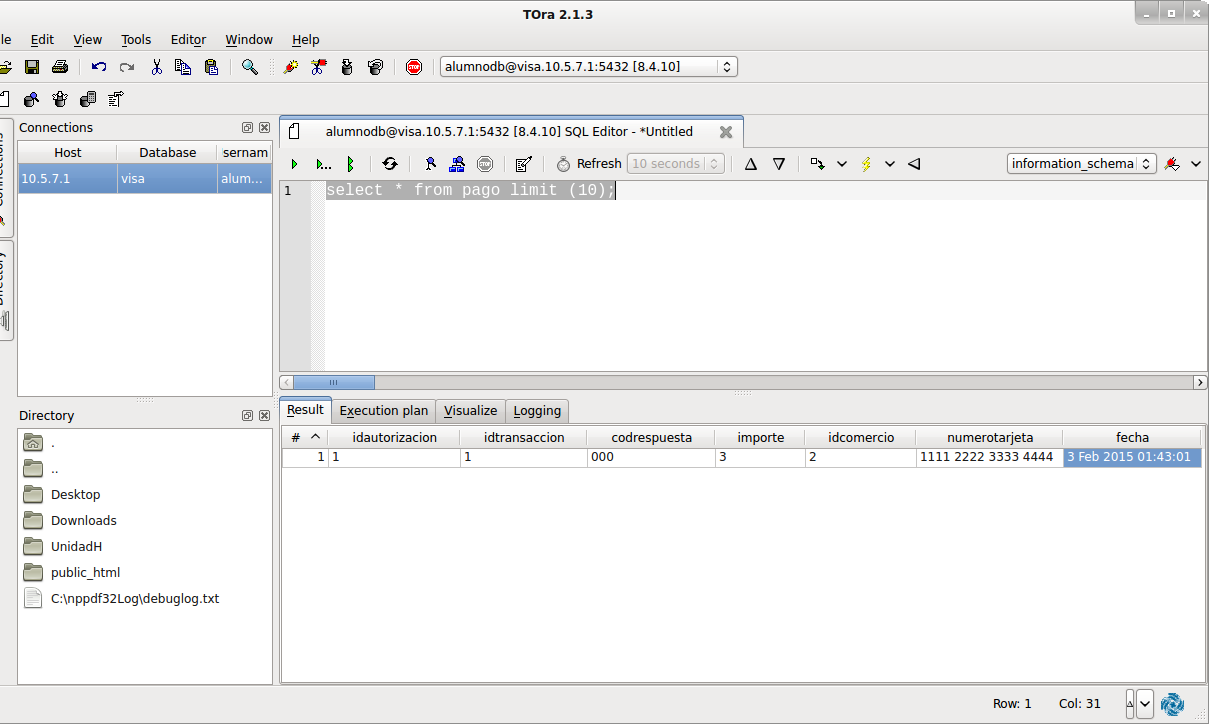
A continuación:

- 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.

Cambiamos los ficheros properties para poner correctamente las direcciones IP.

- 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.







## Cuestión número 2:

La clase VisaDAO implementa los dos tipos de conexión descritos anteriormente. Sin embargo, la configuración de la conexión utilizando la conexión directa es incorrecta. Se pide completar la información necesaria en dicha clase para llevar a cabo la conexión directa de forma correcta. Para ello habrá que fijar los atributos de dicha clase 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 9.1 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.

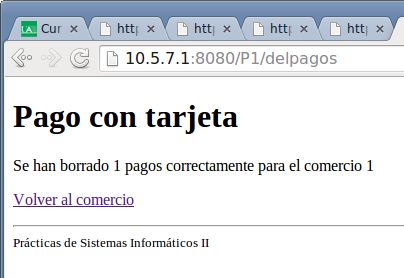
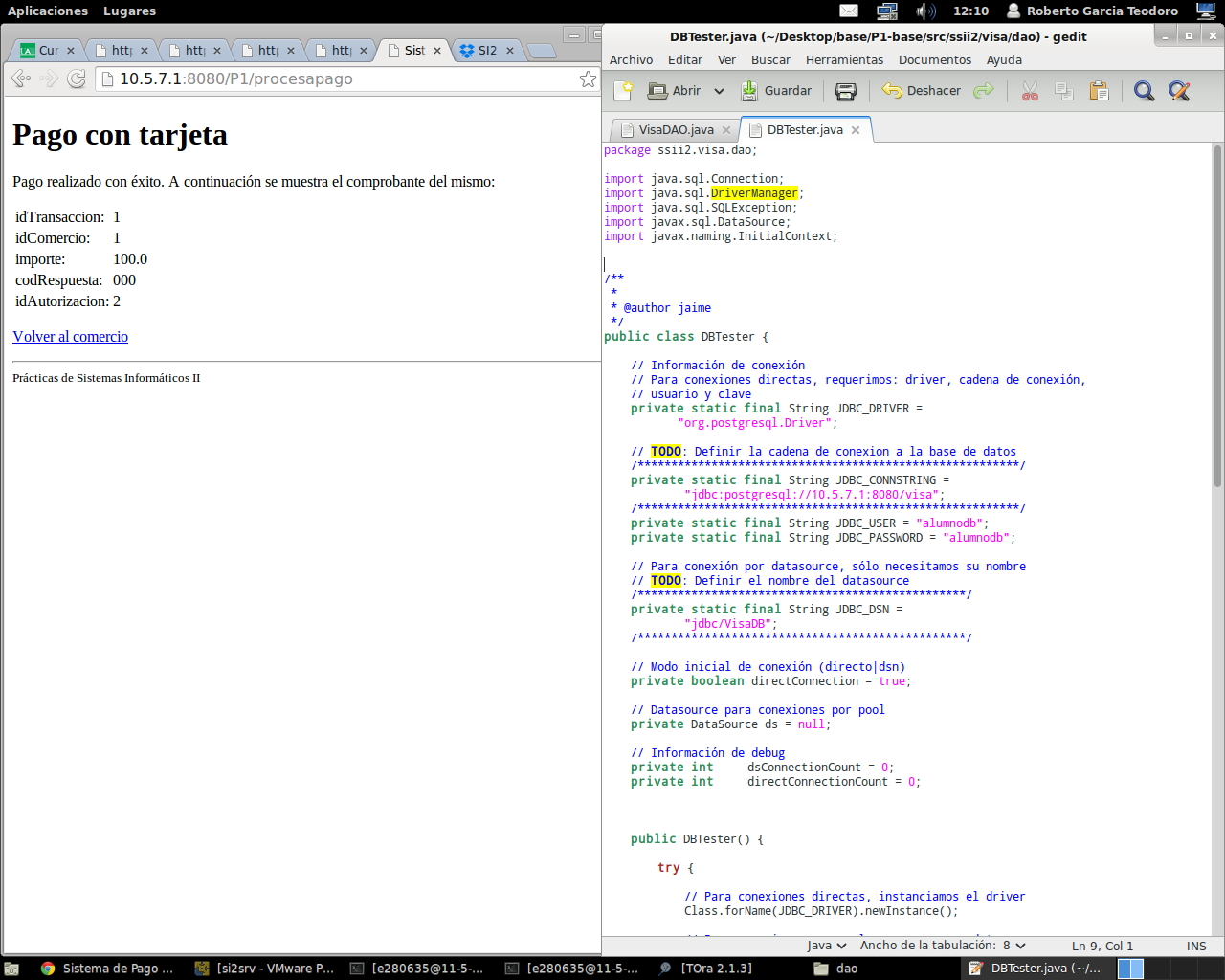
Cambiamos lo siguiente:

*private static final String JDBC\_DRIVER = "org.postgresql.Driver";*

*private static final String JDBC\_CONNSTRING ="jdbc:postgresql://10.5.7.1:5432/visa";*

*private static final String JDBC\_USER = "alumnodb";*

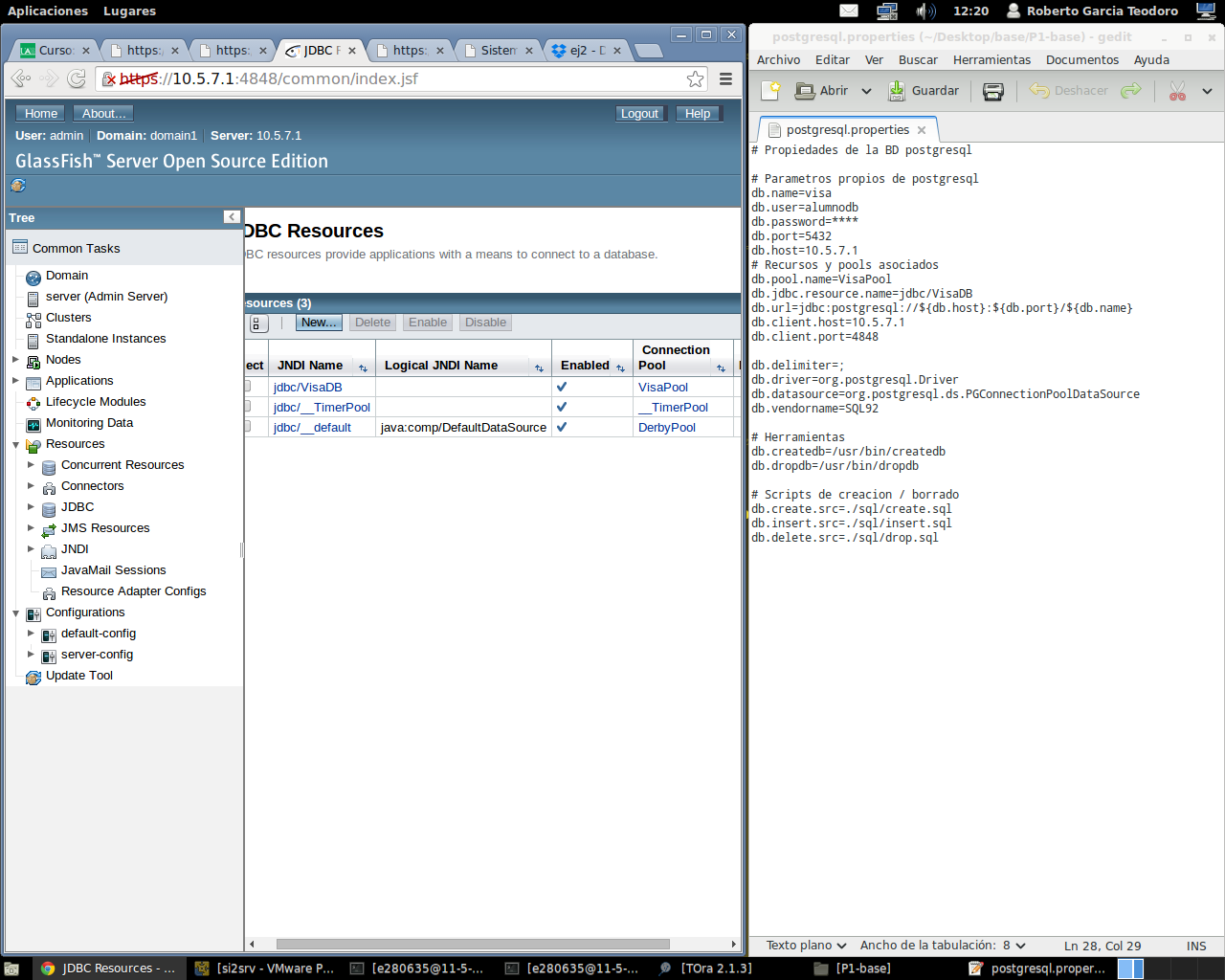
*private static final String JDBC\_PASSWORD = "";*



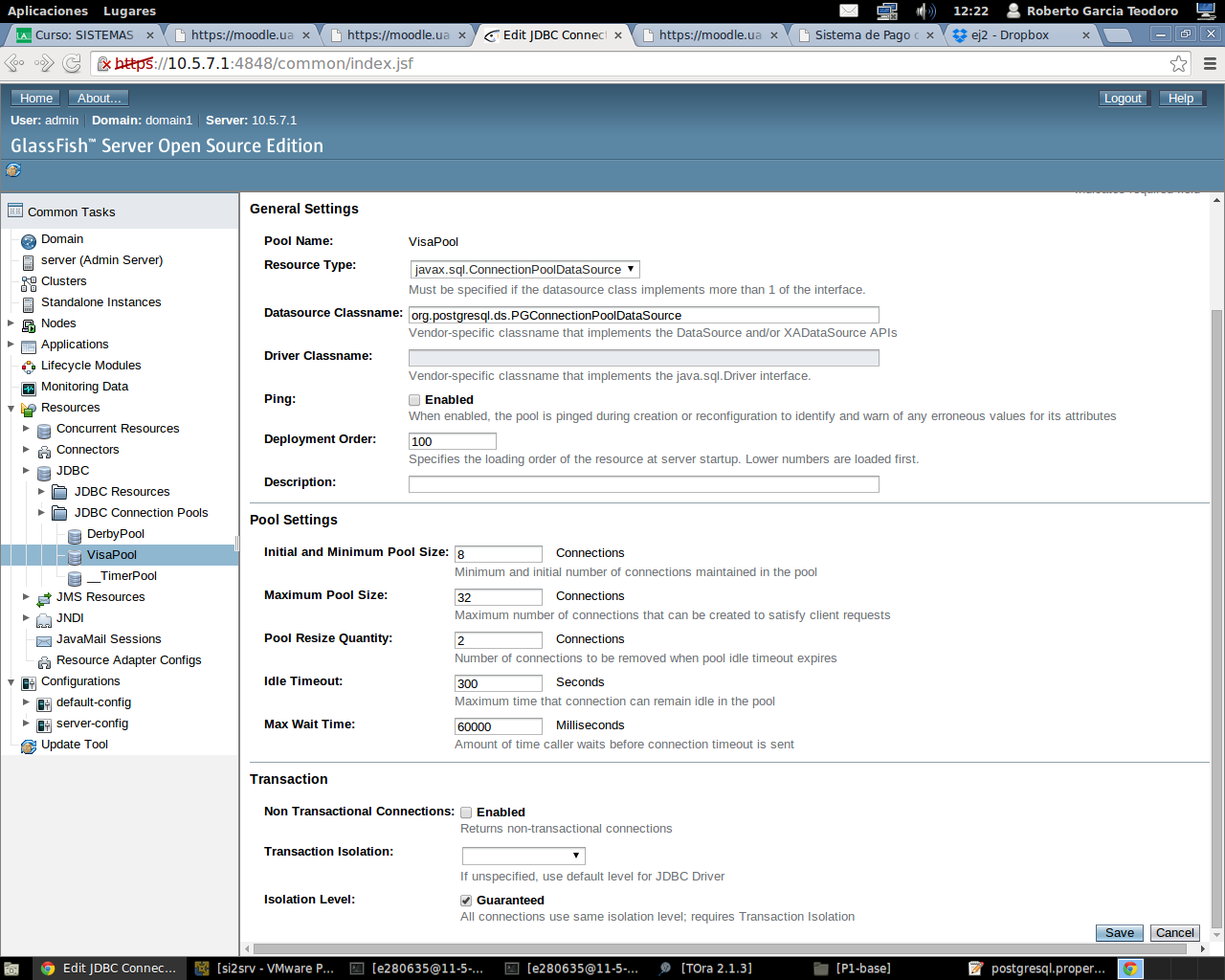
## Cuestión número 3:

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.

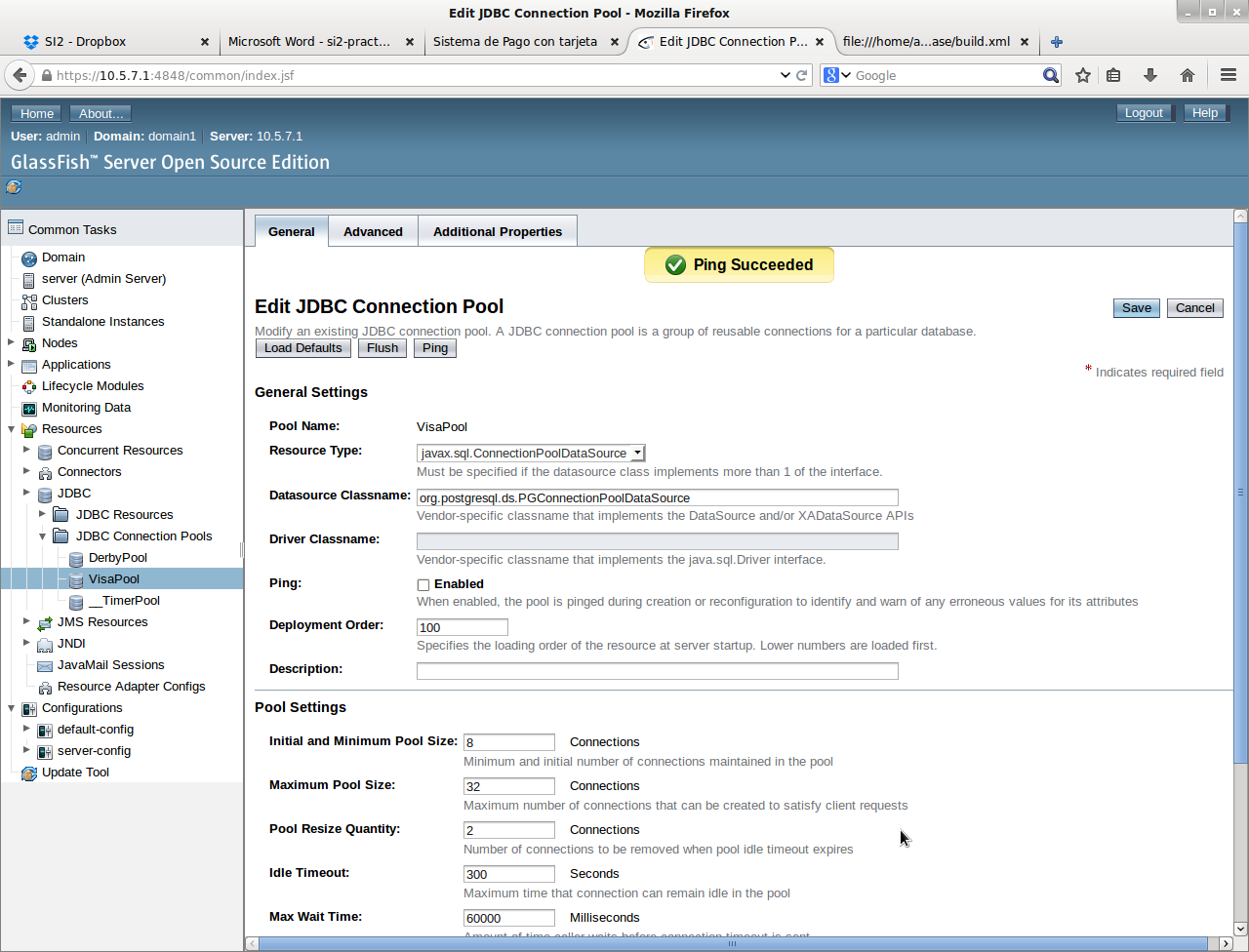
Examinamos el archivo y accedemos a la consola de administración:



Vemos los valores:



Hacemos un ping:



Pues por ejemplo si cambiamos el número de conexión que permitimos ya nos darían rendimientos distintos, y además nos cambiaría el tipo de comportamiento de la aplicación, es decir puede haber casos en los que nos interese más permitir muchas conexiones y otros en los que no, ya sea un servicio de internet como una tienda, etc.

También otros datos como el tiempo máximo de espera son bastante útiles ya que si permitimos muchas conexiones necesitaremos un tiempo de espera mayor ya que puede que el servidor este ocupado realizando otras operaciones

## Cuestión número 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.

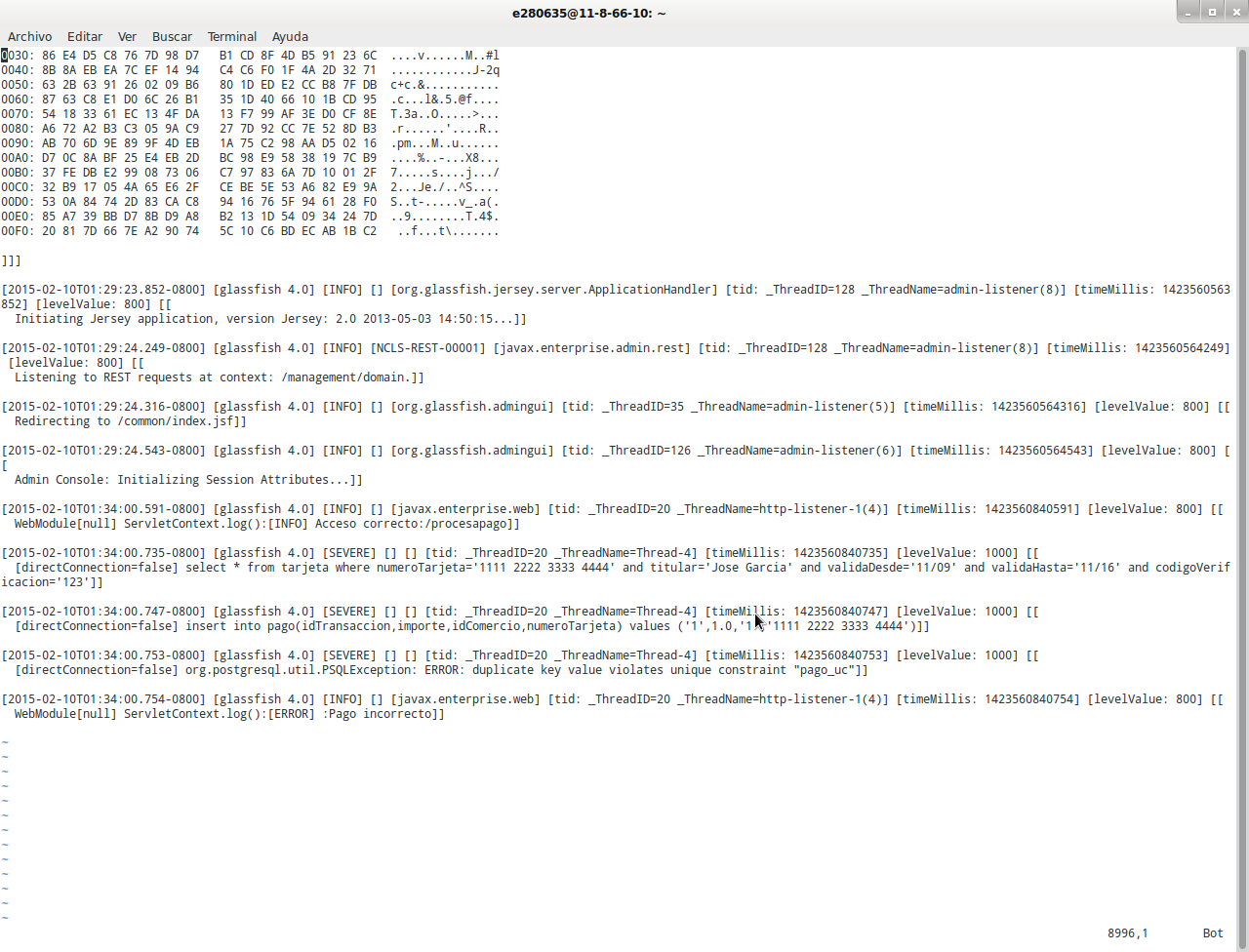


## Cuestión número 5:

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

errorLog se llama en: compruebaTarjeta, realizaPago, getPagos y delPagos

log del servidor:



## Cuestión número 6:

Realícese 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()

• 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:

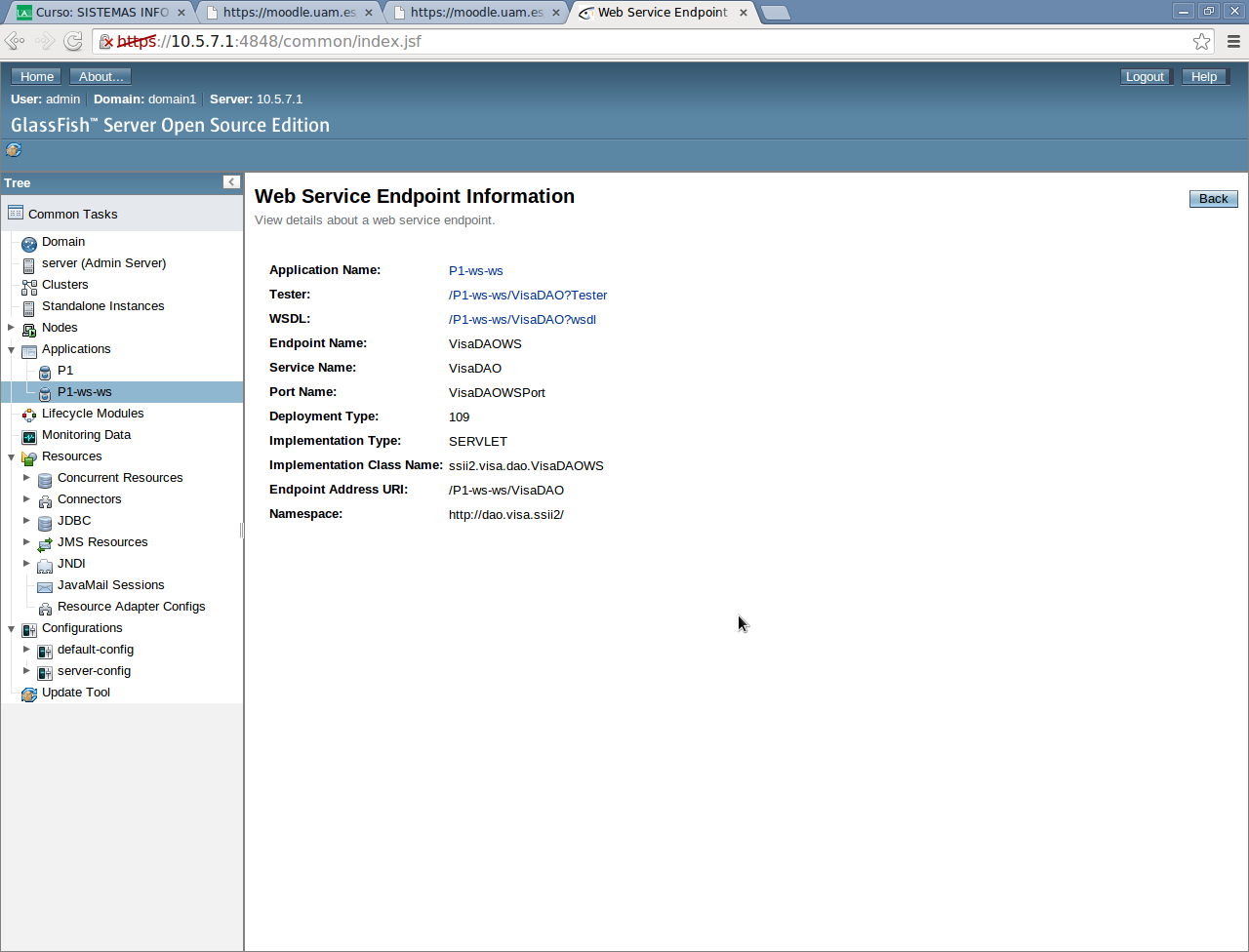
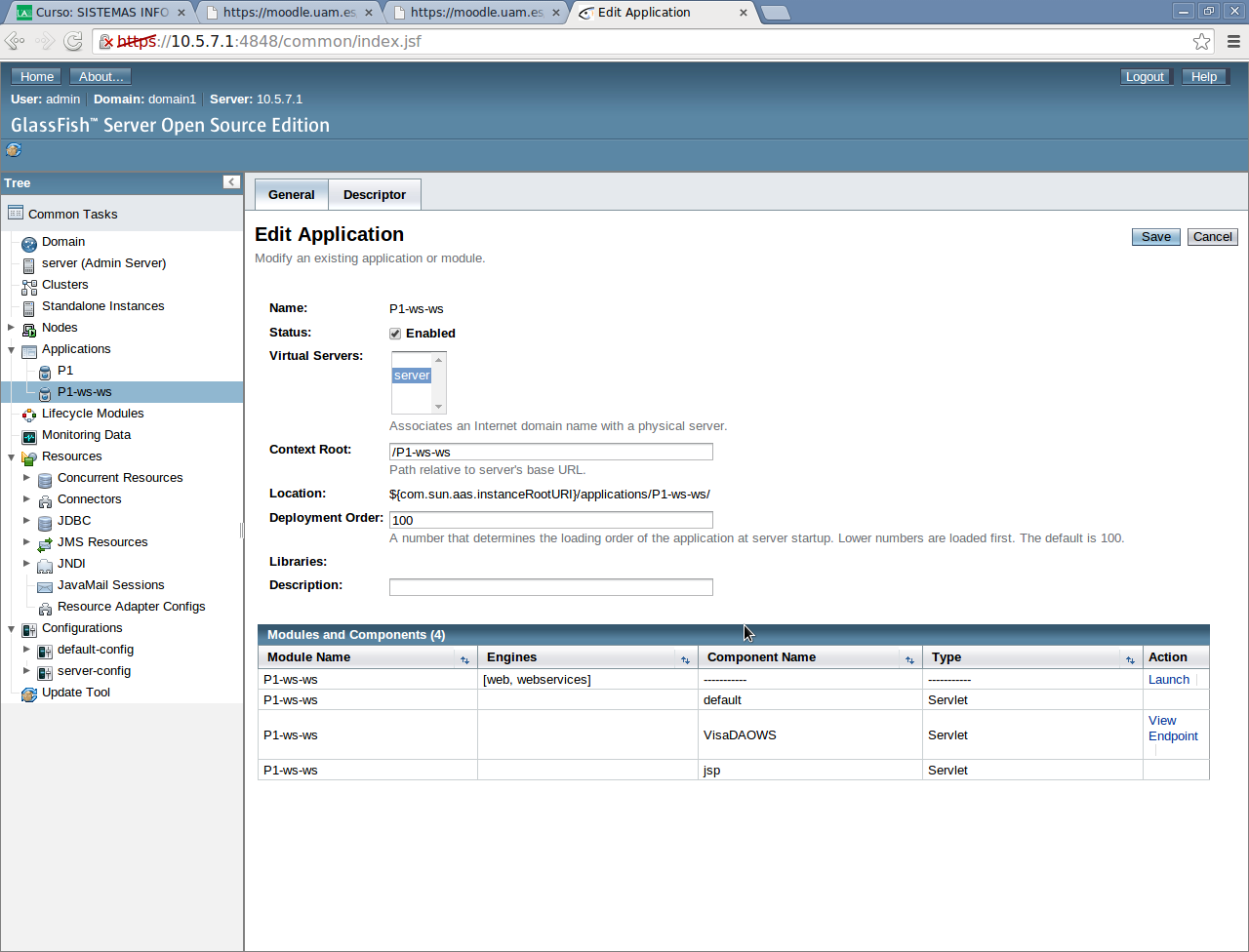
• Con identificador de autorización y código de respuesta correcto en caso de haberse realizado.

• Con null en caso de no haberse podido realizar.

Se adjunta el fichero visaDAOWS con los métodos cambiados.

## Cuestión número 7:

Despliegue el servicio con la regla correspondiente en el build.xml. Acceda al WSDL remotamente con el navegador.



Se adjunta un fichero xml que contiene el WSDL

## Cuestión número 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. C:\Users\eps\Downloads\ej8_nuevascosas.png

Hemos tenido que cambiar las referencias que había a VisaDAOWSService por VisaDAO ya que no nos funcionaba, porque en algún punto antes del ejercicio 7 el nombre se puso o lo pusimos mal, se puede ver en la captura del ejercicio 7.

## Cuestión número 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 12.5.1 para más información y edite el fichero web.xml y analice los comentarios que allí se incluyen.



## Cuestión número 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.

Hemos cambiado los lugares donde se llamaba a getPagos, por ejemplo:

*PagoBean[] pagos = (PagoBean [])dao.getPagos(idComercio).toArray();*

## Cuestión número 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.

wsimport -d build/client/WEB-INF/classes -p ssii2.visa http://10.5.7.1:8080/P1-ws-ws/VisaDAO?wsdl

## Cuestión número 12:

Complete el target generar-stubs definido en build.xml para que invoque a wsimport.

Hemos incluido lo siguiente en el build.xml:

*<exec executable= "wsimport">*

*<arg line="-d" />*

*<arg line="${build.client}/WEB-INF/classes" />*

*<arg line="-p" />*

*<arg line="${paquete}.visa" />*

*<arg line="${wsdl.url}" />*

*</exec>*

## Cuestión número 13:

Realice un despliegue de la aplicación completo en dos nodos.

• Probar a realizar pagos correctos a través de la página testbd.jsp. Ejecutar las consultas SQL necesarias para comprobar que se realiza el pago.

