Sistemas Orientados a Servicios
Graduado en Ingeniería Informática

Práctica: Definición e implementación de un servicio web JAVA

La práctica consiste en implementar un servicio web, *UserManagement*, que simula un portal para la gestión de estudiantes. En este portal habrá un usuario administrador que dará de alta/baja a usuarios, además de usuarios que pueden cambiar su contraseña, eliminar su cuenta o consultar asignaturas y añadir-consultar notas de asignaturas. Además, se dispone de un servicio web, *UPMCourses*, que ofrece operaciones para consultar la lista de asignaturas por curso, así como consultar si existe una asignatura dada.

Las operaciones en Java del servicio web *UserManagement* que se debe generar desde el fichero WSDL, son*:*

1. **Response login (User user)**

Cada llamada a esta operación comienza una nueva sesión para un usuario (*user*). El parámetro *user* tiene dos elementos: nombre (*name*) y contraseña (*pwd*). La respuesta (*Response*) es un *booleano.* El valor *true* se devuelve si la operación de *login* tiene éxito. En caso contrario se devuelve *false*. Si esta operación tiene éxito, el usuario podrá llamar al resto de las operaciones del servicio usando esa misma sesión.

Si se llama a cualquier otra operación del servicio (salvo *logout*) sin haber comenzado una sesión con éxito, la operación llamada devolverá siempre *false*.

Si se llama repetidas veces a *login* en una sesión activa con el mismo usuario ya autenticado, el método devuelve *true* independientemente de la contraseña utilizada. La sesión del usuario en curso sigue activa.

Si se llama a *login* en una sesión activa del usuario *U1* con un usuario *U2*, distinto del usuario autenticado (U2 distinto de U1) el método devuelve *false,* independientemente de la contraseña utilizada. El usuario U2 no va a tener sesión valida, por lo tanto, no podrá acceder a ningún otro método. La sesión del usuario en curso (U1) sigue activa.

Un mismo usuario puede tener varias sesiones activas concurrentemente. Si ese usuario decide cerrar una de sus sesiones solo se cerrará la sesión elegida dejando activas todas las demás.

2. **void logout()**

Esta operación cierra la sesión de un usuario. A partir de ese momento todas las llamadas a operaciones del servicio, excepto *login*, devolverán *false.* Si esta operación es llamada sin que el usuario haya iniciado sesión (*login* correcto) la llamada es ignorada.

Si un usuario tiene varias sesiones abiertas (ha hecho varias llamadas a la operación login con éxito), una llamada a la operación logout solo cerrará una sesión. Para cerrar todas las sesiones deberá llamar a la operación logout tantas veces como sesiones hay abiertas desde el correspondiente programa/navegador.

3. **Response addUser (User user)**

Esta operación añade un usuario al sistema. Solo el usuario *superuser* puede llamar esta operación. El parámetro user tiene el nombre (name) y contraseña (pwd) del usuario a añadir. La respuesta (Response) es true si la operación tiene éxito. La operación devuelve false si el *superuser* intenta añadir un usuario con un *username* ya registrado o si un usuario distinto del *superuser* llama a esta operación.

4. **Response removeUser (Username username)**

Esta operación elimina un usuario del sistema. Cualquier usuario puede invocar esta operación para borrarse a sí mismo, pero solo el usuario *superuser* puede llamar esta operación para borrar cualquier usuario. El parámetro *username* tiene el nombre del usuario a eliminar. La respuesta (*Response*) es *true* si la operación tiene éxito. La operación devuelve *false* si el *superuser* intenta eliminar un usuario con un *username* no registrado o si un usuario distinto del *superuser* llama a esta operación.

El *superuser* puede eliminar a usuarios con sesiones activas. El *superuser* no puede auto-eliminarse.

Un usuario no puede auto-eliminarse si tiene sesiones activas.

5. **Response changePassword (PasswordPair passwordPair)**

Esta operación permite que un usuario (*superuser* incluido) ya registrado y que ha iniciado sesión pueda cambiar su contraseña. El parámetro *passwordPair* incluye la contraseña antigua (*oldpwd*) y la nueva (*newpwd*). La respuesta (*Response*) es *true* si la operación tiene éxito, es decir, la contraseña antigua coincide con la contraseña actual (en el caso de varias sesiones abiertas, cada llamada a changePassword modificaría la contraseña actual) y se ha realizado el cambio de contraseña. La operación devuelve *false* en caso contrario.

6. **CourseResponse showCourses (Course course)**

Esta operación permite que un usuario que haya iniciado sesión pueda consultar la lista de asignaturas de un curso determinado. Esta operación debe acceder al método *showCourses* del servicio *UPMCourses.* La operación devuelve un objeto respuesta (*Response*) con un *boolean* (*result*) que es *true* si ha tenido éxito y un array con las asignaturas correspondientes al curso enviado por parámetro.

7. **addCourseGradeResponse addCourseGrade (CourseGrade courseGrade)**

Esta operación permite que un usuario que haya iniciado sesión correctamente pueda agregar una nota a una asignatura existente. El parámetro *courseGrade* incluye un campo *String* (*course*) con el nombre de la asignatura, que debe existir en el sistema (debe hacer uso del método *checkCourse* del servicio *UPMCourses*) y un campo *double* con la nota que se quiere añadir para la asignatura (desde 0.0 a 10.0).

La respuesta (*Response*) es *true* si la operación tiene éxito y *false* en caso contrario.

Un usuario puede introducir tantas veces como desee la nota de una asignatura, pero solamente se quedará registrado el último cambio.

8. **showAllGradesResponse showAllGrades ()**

Esta operación devuelve la lista de asignaturas y notas (*CourseGrade*) del usuario. La operación devuelve un objeto respuesta (*Response*) que contiene un *boolean* (*result*) que es *true* si el usuario que llama a la operación ha hecho *login* previo con éxito y un array de objetos CourseGrade ordenados por nota (mayor a menor) con las asignaturas y notas del usuario.

Para realizar la práctica se emplearán las herramientas de Axis2 para Java y se deberá desplegar el servicio en Tomcat. El WSDL que define este servicio se encuentra disponible en la siguiente dirección:

http://ochoa.dia.fi.upm.es:8080/practica1819/UserManagement.wsdl

El WSDL del servicio ***UPMCourses*** se encuentra disponible y funcionando en:

http://ochoa.dia.fi.upm.es:8080/axis2/services/UPMCourses.wsdl

y tiene las siguientes operaciones:

1. **showCoursesResponse showCourses (ShowCourses course)**

Esta operación devuelve la lista de asignaturas para un curso dado. El parámetro de entrada contiene el curso del que se desea obtener el listado de asignaturas (1 a 4) y devuelve un listado con las asignaturas existentes en el sistema para ese curso. En caso contrario devuelve una lista vacía.

2. **checkCourseResponse checkCourse (CheckCourse course)**

Esta operación comprueba si la asignatura pasada por parámetro existe en el sistema. El parámetro de entrada contiene un campo String, *name, que* debe coincidir (sin tener en cuenta mayúsculas y minúsculas) con una asignatura del sistema, en caso de haber coincidencia devuelve true en la respuesta, en caso contrario devuelve false.

Requisitos del servicio web *UserManagement*

1. En el momento del despliegue el servicio, éste tendrá al usuario superuser con username admin y contraseña admin. Solo puede haber un superuser en el sistema y éste puede cambiar su contraseña utilizando la operación changePassword.
2. La información de los usuarios (username, password) debe ser almacenada en el servicio UserManagement (en memoria).
3. La información de los calificaciones de asignaturas (*name*, *grade*) por usuario debe ser almacenada en el servicio *UserManagement* (en memoria).
4. Se tendrán que hacer pruebas con distintos clientes conectados al mismo tiempo.
5. El estado del servicio tiene que persistir a la ejecución de los clientes. Por ejemplo, ejecutando varias veces el mismo cliente el estado inicial del servicio que se encontrará el cliente será distinto en cada ejecución.

Se pide:

- Implementar el servicio web en java empleando Axis2.

- Programar una aplicación que realice las pruebas para el completo funcionamiento del servicio web anteriormente descrito. Se debe tener en cuenta la creación de distintos clientes para comprobar el correcto funcionamiento del servicio.

Instrucciones para la entrega de la práctica:

FECHA DE ENTREGA: 26-05-2019 hasta las 23:55.

La práctica debe realizarse por parejas. Solo se entregará una práctica por pareja a través de Moodle.

Todas las parejas deberán subir a Moodle el fichero comprimido (.tar.gz, .rar, .zip, .7z) con una carpeta llamada appellido1appellido2 con el siguiente contenido:

– Una carpeta llamada "servicio" con todo el código fuente del servicio, la carpeta resources y el build.xml. El formato de la carpeta tiene que estar listo para que ejecutando el comando "ant" dentro de la carpeta "servicio" se cree el fichero.aar

– Una carpeta llamada "cliente" con todo el código fuente del cliente, la carpeta resources y el build.xml.

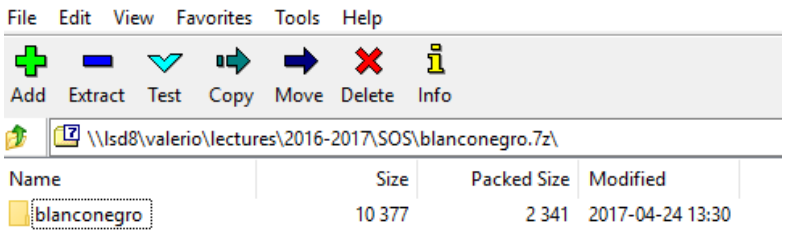
– Una copia de la clase "skeleton" con la implementación del servicio.

– El fichero de despliegue .aar para desplegar el servicio en Tomcat.

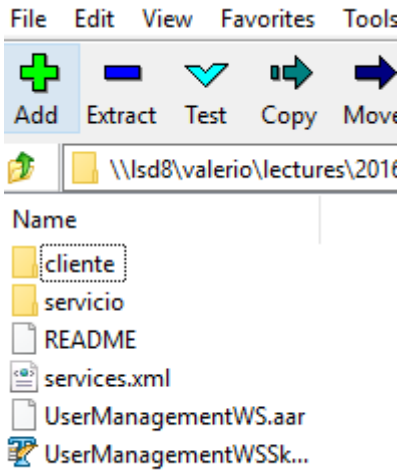
– Un README con los datos de la pareja.

Ejemplo de archivo de entrega

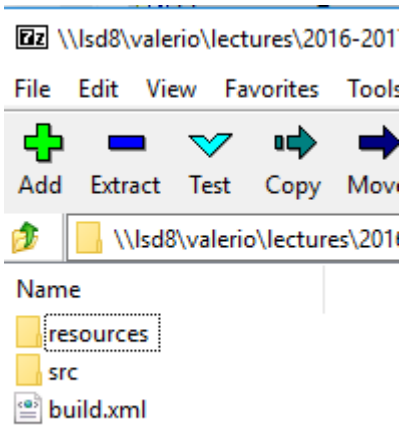
- Pareja: Juan Blanco y Paco Negro Raíz del fichero rar:



- Explorando la carpeta "blanconegro" hay:



- Explorando la carpeta “servicio” hay:



El nombre completo de la clase skeleton debe ser:

es.upm.fi.sos.t3.usermanagement.UserManagementWSSkeleton.java

Los alumnos pueden descargarse una máquina virtual con el mismo entorno que se utilizará para la corrección de la práctica. Ésta se encuentra disponible en Moodle

Para aprobar la práctica, ésta deberá funcionar bien con el software incluido en la máquina virtual y descrito en la guía de instalación de herramientas (con JDK versión 1.7 y axis2 versión 1.6.2).