Práctica 3– EJB + WS + REST + Mensajes

Óscar Rubio García - UO240707
Christian Martínez Abad - UO237441

Descripción de las modificaciones añadidas a la anterior entrega: Interfaces, entradas JNDI, DataSources, etc.

- Un cambio importante que fue añadido al servidor fue un filtro de acceso a las páginas con path /rest/*. Esto significa que si alguien fuera a intentar acceder a alguna página del servicio rest debería pasar un control de seguridad. Este control de seguridad fue creado para asegurar que solo los usuario válidos del sistema puedan entrar en los servicios rest.
- Un archivo sdi3-13-ds.xml fue creado para poder acceder al datasrouce.

Para el cliente de mensajería, descripción de las modificaciones realizadas para soportar esta funcionalidad y colas o temas creados y disponibles en JNDI

- Para la creación del cliente de mensajería fue necesario crear además del cliente, un MessageListener en sdi3-13 que está encargado de escuchar los mensajes que vienen de la queue SDI3Queue. Este Listener actuará como producer además de consumer cuando reciba ciertos comandos. En el caso de que sea necesario usarlo como producer usará la queue SDI3AuditQueue para devolver cualquier mensaje de vuelta a su cliente. En caso de un intento de inicio de sesión erróneo o incorrecto el Listener enviará un mensaje detallando el intento de inicio de sesión a la queue SDI3AdminQueue.
- Los mensajes contienen el login del usuario del cliente y su password, esta informacion será utilizada para asegurarse de que el usuario existe en el sistema y esta activo, ademas se utilizará también para asegurar que cuando un mensaje de vuelta sea recibido por el cliente correcto.
- Existen 2 Listeners, uno llamado SDI3AuditMessageListener está en el cliente y es el encargado de leer los mensajes de la queue SDI3AuditQueue y

asegurarse de procesar sólo aquellos que sean relacionados con su usuario. El otro Listener llamado SDI3MessageListener está en el servidor y actúa como consumer y producer con el fin de recibir mensajes de los clientes por la queue SDI3Queue y reenviar las respuestas por la queue SDI3AuditQueue. Este último listener también enviará mensajes a la queue SDI3AdminQueue si llegara un mensaje a la queue cuyas credenciales no fueran válidas.

- Las queues fueron descritas en un archivo llamado channels-jms.xml en META-INF, estas queues son SDI3Queue, SDI3AuditQueue y SDI3AdminQueue.

Una descripción clara y detallada de las partes opcionales implementadas

Cliente SOAP C#: tiene las mismas funcionalidades que el cliente SOAP JAVA. Hace uso de los elementos recopilados en el wsdl del servidor. Para ejecutarlo, dentro de la carpeta Cli.Csharp podrás encontrar un ejecutable en *sdi3-13.Cli-CSharp/bin/debug*. El código está disponible en el archivo .cs.

Cualquier otra información necesaria para una descripción razonablemente detallada de lo entregado y su correcto despliegue y ejecución

- Comunicación mínima con la base de datos. Para reducirla al mínimo, hemos hecho algunos cambios en los modelos. Por ejemplo, ahora User tiene una array de 4 ints, relativos al número de tareas completadas, tareas completadas con retraso, tareas planificadas y no planificadas. De este modo podemos obtener toda esta información de cada usuario con una sola query.
- Es necesario añadir el archivo sdi3-13-ds.xml a el servidor wildfly, standalone/deployments. Este archivo actualmente se encuentra guardado en la carpeta META-INF.
- *SI EL CLIENTE C# SE CIERRA INESPERADAMENTE*: el wsdl contiene al final una url donde aparece el nombre del equipo en el que se está ejecutando el servidor. En el cliente java se actualiza la referencia con cada ejecución, pero en el cliente c# no. Hemos solucionado este problema "engañando" al programa, cambiando el nombre del equipo de desarrollo a "localhost". Esto debería funcionar, ya que cada ejecución busca en *localhost:8280*… Si no te funciona, añadir de nuevo la referencia wsdl al proyecto c# solucionará este problema.