Esta práctica se puede realizar por parejas o de manera individual y su entrega es obligatoria.

Funcionalidad

Partiendo de la entrega de la práctica anterior se debe añadir esta funcionalidad:

- **Implementar capa de servicios con EJB**
  - Debe permitir acceso local y remoto.
  - Modificar adecuadamente capa de persistencia para obtener control transaccional correcto.
  - Debe ser seleccionable por configuración (no por programación):
    * La ubicación local o remota de la capa web con respecto a la de lógica.
    * Las consultas de la capa de persistencia.
    * La obtención de la conexión a la BDD.
- Realizar un **cliente EJB remoto** para tareas administrativas[1] que permita:
  - Listar usuarios del sistema: sus datos personales, el número de tareas completadas, el número de tareas completadas retrasadas, el número de tareas planificadas y el número de tareas sin planificar.
  - Deshabilitar un usuario del sistema (UserStatus.DISABLED).
  - Eliminar todo rastro en el sistema de un usuario. Implica eliminar sus tareas, categorías y finalmente el usuario.
- Añadir **acceso WS SOAP** al servidor y
  - Hacer otro cliente con la misma funcionalidad que el anterior.
- Añadir **acceso WS REST** al servidor y hacer un cliente que tras pedir la identidad del usuario le permita:
  - ver la lista categorías,
  - seleccionar una categoría de la lista anterior para ver sus tareas pendientes y atrasadas, ordenadas por fecha de más antiguo a más reciente
  - marcar una tarea como completada, y
  - registrar una nueva tarea.

  Las peticiones REST deben ir con autenticación, presentando las credenciales del usuario registrado propietario de las tareas.
- **Mensajería**. Desarrollar un programa cliente que, tras pedir la identidad del usuario, le permita:
  - Visualizar las tareas de la pseudolista *Hoy* y atrasadas
  - Marcar como terminada una de esas tareas
  - Añadir una nueva tarea

  La comunicación entre el cliente y el servidor debe hacerse exclusivamente por paso de mensajes. Cada mensaje enviado será un comando que se procesará en el servidor. La respuesta vendrá en un nuevo mensaje que sólo el emisor del mensaje de petición procesará. El servidor verificará que el usuario que manda el mensaje es válido en el sistema (credenciales correctas y no deshabilitado). En caso de ser un envío de un usuario no válido o un mensaje con formato incorrecto el mensaje será reenviado a una cola que, a modo de log, almacenará todos esos mensajes ilegales para su posterior análisis por un administrador del sistema.

[1] Puede ser necesario añadir algún método más a la interfaz de la capa de servicios.

Opcionales

- Desarrollar un cliente WS con otra tecnología (C#, C++, Ruby, Android, iOS, etc.)[2].
- Desarrollar un cliente REST con otra tecnología (JavaScript, C#, Ruby, Android, iOS, etc.).
- Hacer la capa de persistencia con JPA e integrarla con los EJB.
- Alguna otra propuesta consensuada con el profesor.

Diseño

Todas las consideraciones de funcionalidad y diseño referentes a la entrega anterior son de aplicación aquí (esto es una ampliación de la entrega anterior). Son de destacar especialmente:

- Las indicaciones referentes a la implementación de las capas de presentación, negocio y persistencia.
- El contenido mínimo de la base de datos y el método de resetado de ésta.

Los clientes desarrollados que necesiten autenticación (acceso al JNDI o REST) presentarán al servidor credenciales del usuario "sdi/password" que ya está registrado en los archivos *application-users.properties* y *application-roles.properties* de los servidores del entorno. En el caso del cliente REST si está activada la autenticación con filtro podrá ser cualquier otro usuario registrado en la aplicación.

El fichero de configuración de *Destinations JMS* debe ir en la carpeta META-INF del proyecto EJB y con un nombre que siga el patrón **sdi3-<n>-jms.xml**. De la misma forma, si se usa una configuración de *Datasource* distinta de la que hay por defecto en el entorno, el fichero debe ir en la misma carpeta META-INF y seguir el patrón de nombre **sdi3-<n>-ds.xml**.

Entrega

- Todos los proyectos Eclipse de la solución (servidor y clientes). El nombre de dichos proyectos empezará por "sdi3-n" (en minúsculas) siendo n el número asignado en el listado publicado en el campus virtual. Si ha habido cambios en alguna pareja deben ser gestionados con el profesor a fin de obtener un nuevo valor n.

EAR	sdi3-n
WEB	sdi3-n.Web
EJB	sdi3-n.EJB
Cliente EJB	sdi3-n.Cli-EJB
Cliente SOAP	sdi3-n.Cli-SOAP
Cliente REST	sdi3-n.Cli-REST
Cliente mensajería	sdi3-n.Cli-MSG
Cliente otra tecnología	sdi3-n.Cli-<…>

- Se deberá subir a la tarea correspondiente del campus virtual un único archivo ZIP (usando el formato ZIP) con el nombre sdi3-n.zip (en minúsculas) y que deberá contener en su raíz:

[2] El profesor puede pedirte que hagas una demostración de funcionamiento si no tiene el entorno adecuado para probarlo.

- Un documento PDF (con el mismo nombre que el proyecto sdi3-n.pdf) que contenga:
  * Descripción de las modificaciones añadidas a la anterior entrega: Interfaces, entradas JNDI, DataSources, etc.
  * Para el cliente de mensajería descripción de las modificaciones realizadas para soportar esta funcionalidad y colas o temas creados y disponibles en JNDI.
  * Una descripción clara y detallada de las partes opcionales implementadas.
  * Cualquier otra información necesaria para una descripción razonablemente detallada de lo entregado y su correcto despliegue y ejecución.
- Las carpetas de los proyectos Eclipse, sin comprimir, cada una denominada conforme al patrón de nombres presentado en la tabla anterior.

– Los proyectos estarán configurados para funcionar en el servidor interno.

Fecha máxima de entrega

En la semana del 15 de mayo cada alumno entregará el día de la semana coincidente con el grupo de laboratorio al que pertenece. En el caso de parejas basta con que entregue uno de los miembros.

Evaluación

* = obligatorio	**Puntos**
* Capa de EJB	2
* Cliente remoto EJB	1,5
* Cliente SOAP	1,5
* Cliente REST	2
* Cliente Mensajería	2
Cliente WS otra tecnología	1
Cliente REST otra tecnología	1
Persistencia con JPA integrado	1

* Estas partes son obligatorias. Si falta alguna de ellas la entrega no será corregida y figurará con nota = 0

Se penalizará:

– La escasa claridad del código.
– Que el código Java NO se ajuste a las *Java Code Conventions* (nombres poco significativos, mala capitalización, mal sangrado, líneas demasiado largas, etc.)
– Que la compilación del código genere "*warnings*".