1. Definición de comunicación e integración entre proyectos propios:
   1. La primera opción sería que el proyecto LogisticaREST conozca al EJB Project (LogisticaApp y LogisticaAppClient), mediante la opción de *Deployment Assembly,* para conocer las interfaces de los EJB. De este modo automáticamente se van a compilar esos dos proyectos como .*jar* y se van a poder inyectar los EJB y usar directamente en el proyecto de LogisticaREST.
   2. La segunda opción, que es en definitiva la que hicimos, sería que a través del lookup de JNDI, encontramos las interfaces de los EJB, y luego llamamos a los métodos (estilo similar a RMI). No tiene nada de malo esta opción según Leandro, y así lo resolvió el grupo de Lucas Aguirre.

Problemas de esta opción: ¿dónde metemos la interfaz y dónde metemos los DTOs? Podríamos tener un repositorio en común, donde colocamos los DTOs, interfaz y enums, y este repositorio se puede compilar en un *.jar* y se puede pasar a otras aplicaciones; o bien, se puede copiar los DTOs y las interfaces en el proyecto de LogisticaREST, pero obviamente tenemos la desventaja que si cambia algo en cualquier DTO de LogisticaApp hay que copiarlo y pegarlo en LogisticaREST, es decir, mantenerlo siempre actualizados. Ese es el problema de tener copias exactas.

1. Test WebService: Para poder probar los servicios REST que ofrece el proyecto de LogisticaREST tenemos dos opciones:
   1. Hay que instalar *JBoss Tools* completo (se puede hacer desde el Menú *Help* del Eclipse, *Eclipse Marketplace…*), es pesado, conviene mejor seleccionar directamente la opción *JBoss JAX-RS Tools* (en las opciones de instalación de *JBoss Tools*). También debemos marcar la opción de *JBoss WebServices Tools* que es la interfaz para probar los Web Services con el IDE del Eclipse.
   2. Instalar un plugin en el Chrome, que se llama *Advanced REST client*. Es la mejor opción!
2. Respecto a las respuestas de las peticiones de los servicios REST, lo mejor sería enviar una respuesta con el código 200 con el formato del protocolo HTTP (en caso OK) o mandar un 505, 405, etc. (depende el error o exception) y mandarlo como un JSON.
3. Algo a tener en cuenta, según sugerencia de Leandro, podríamos crear dos packages distintos (*dtos* y *jsons*) para separar su uso en las comunicaciones. Lo mejor sería usar los DTOs para las peticiones y respuestas de la página web y utilizar los JSONs (que son iguales o tienen menos atributos que los DTOs) para realizar exclusivamente la comunicación con los servicios web expuestos. Podríamos usar siempre los DTOs para convertir a JSON y viceversa, pero no es la mejor opción.