

PROS@ EN CONTENEDORES

Instalación de entorno desarrollo Liberty / RSA

Fecha Abril de 2024

Versión: 2.2



Gerencia de Informática
de la Seguridad Social

VIEWNEXT
AN IBM SUBSIDIARY

CONTROL DE VERSIONES			
Título	Montaje entorno desarrollo Liberty		
Autor	Viewnext		
Fecha versión 1.0	04/9/2021		
Versión	Fecha	Responsable	Cambios introducidos
1.0	04/03/2021	Viewnext	Versión inicial
1.1	11/04/2022	Arquitectura Pros@	Pequeñas aclaraciones
1.2	24/05/2022	Arquitectura Pros@	Solución de algunos errores adicionales
1.3	09/06/2022	Arquitectura Pros@	Aclaración sobre datasources en Apéndices
1.4	24/06/2022	Arquitectura Pros@	Reestructuración del server.xml y referencias a la wiki para descargas
1.5	28/06/2022	Arquitectura Pros@	URL repositorio, instalación feature localConnector y otras mejoras
1.6	26/07/2022	Arquitectura Pros@	Separación de server.xml para variables locales
1.7	19/08/2022	Arquitectura Pros@	Corrección
1.8	18/10/2022	Arquitectura Pros@	Se añade Liberty 22.0.0.10 y cambio de ConfiguracionServidor
1.9	04/11/2022	Arquitectura Pros@	Se añaden varias explicaciones de Troubleshooting
1.10	08/11/2022	Arquitectura Pros@	Se añade opción de Renovar en Configuración de Servidor de RSA
1.11	17/03/2023	Arquitectura Pros@	Se añade configuración para publicar Web Services

ÍNDICE

1.	INSTALACIÓN PASO A PASO	4
1.1	Instalación servidor Liberty.....	4
1.2	Instalación JDK RedHat	4
1.3	Configuración archivo Host.....	4
1.4	Configuración RSA.....	6
1.4.1	Activación de prestaciones RSA	6
1.4.2	Desactivación validación al construir	6
1.4.3	Configuración JDK	7
1.4.4	Configuración Liberty	9
1.4.5	Server.xml	13
1.4.6	Keystore de Liberty	15
1.4.7	Importar EAR de infraestructura.....	16
1.4.8	Migrar/cambiar a otra versión de EAR de infraestructura.	20
1.4.9	Importar aplicación de trabajo.....	22
2.	APÉNDICES	34
2.1	Breve descripción del archivo server.xml	34
2.2	Instalación de nuevas características (features) Liberty.....	35
2.2.1	Creación archivo repositories.properties.....	35
2.2.2	Ejecución comando instalación.....	35
2.3	Instalación bibliotecas IBM MQ.....	35
2.4	Instalación de la feature local-connector	36
2.5	Encoding UTF-8 en el Workspace	37
2.6	Cómo usar un Almacén de Certificados concreto (key.p12).....	38

1. INSTALACIÓN PASO A PASO

Nos disponemos ahora a describir paso a paso el proceso de montaje del entorno de desarrollo Liberty en el escritorio virtual.

Todos los artefactos están en el siguiente repositorio de NEXUS:

[Browse - Nexus Repository Manager \(portal.ss\)](#)

1.1 Instalación servidor Liberty

En el repositorio tenemos la carpeta Software/Liberty, donde disponemos de varias versiones. Se recomienda usar la siguiente(Diciembre de 2023):



Lo descomprimimos en la carpeta de nuestra preferencia. Para que todos los desarrolladores tengan un entorno de desarrollo lo más uniforme posible, se recomienda encarecidamente la ruta **C:\Liberty**. Para la utilización de almacén de certificados, es necesario ejecutar el comando "`\RUTA_INSTALACION_LIBERTY\bin\installUtility install transportsecurity-1.0`".

1.2 Instalación JDK RedHat

Desde la siguiente url nos descargamos el zip con la openjdk del repositorio, carpeta Software/OpenJDK

Lo descomprimimos en la carpeta de nuestra preferencia. Para que todos los desarrolladores tengan un entorno de desarrollo lo más uniforme posible, se recomienda encarecidamente la ruta **C:\OpenJDK**.

1.3 Configuración archivo Host

Abrimos un cmd y obtenemos la ip de nuestro EV con el comando ipconfig

```
Adaptador de Ethernet Ethernet0:
  Sufijo DNS específico para la conexión... : seg-social.ss
  Vínculo: dirección IPv6 local... : fe80::110c:2dd8:5a8a:a2e2%7
  Dirección IPv4... : 10.198.163.86
  Máscara de subred... : 255.255.254.0
  Puerta de enlace predeterminada... : 10.198.162.3
C:\Users\99GU9045>
```

Ilustración 1 Ejemplo de la ejecución comando ipconfig

Abrimos en modo administrador el siguiente archivo C:\Windows\System32\drivers\etc\hosts

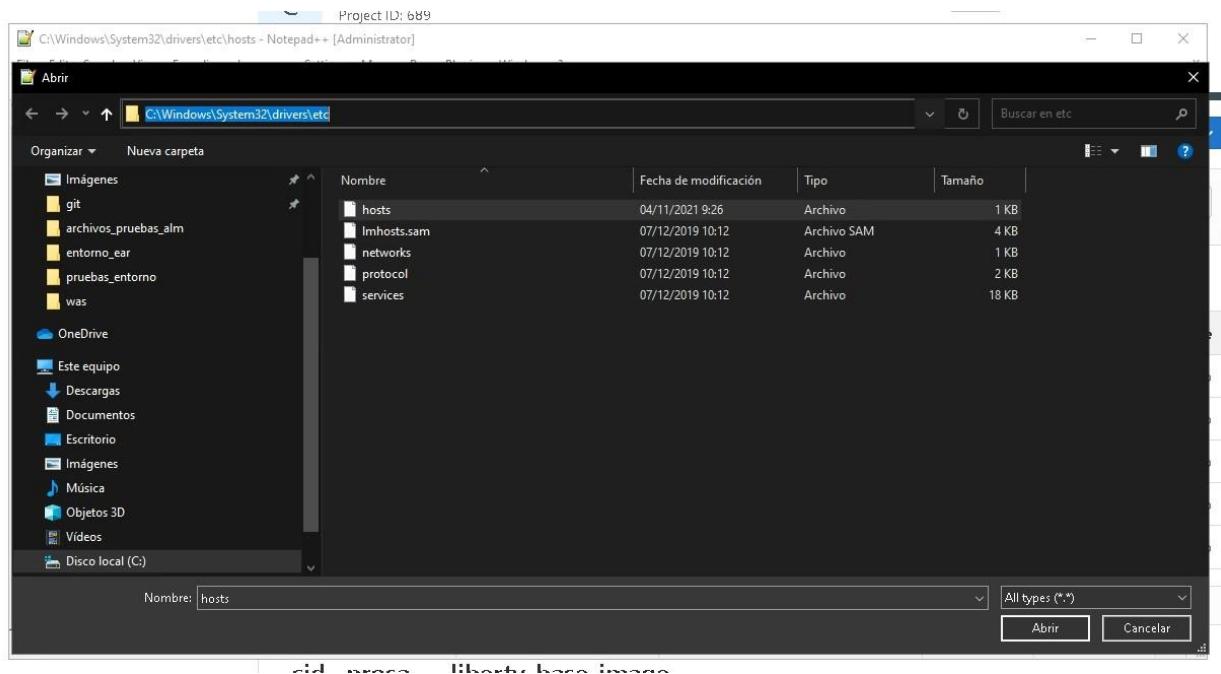
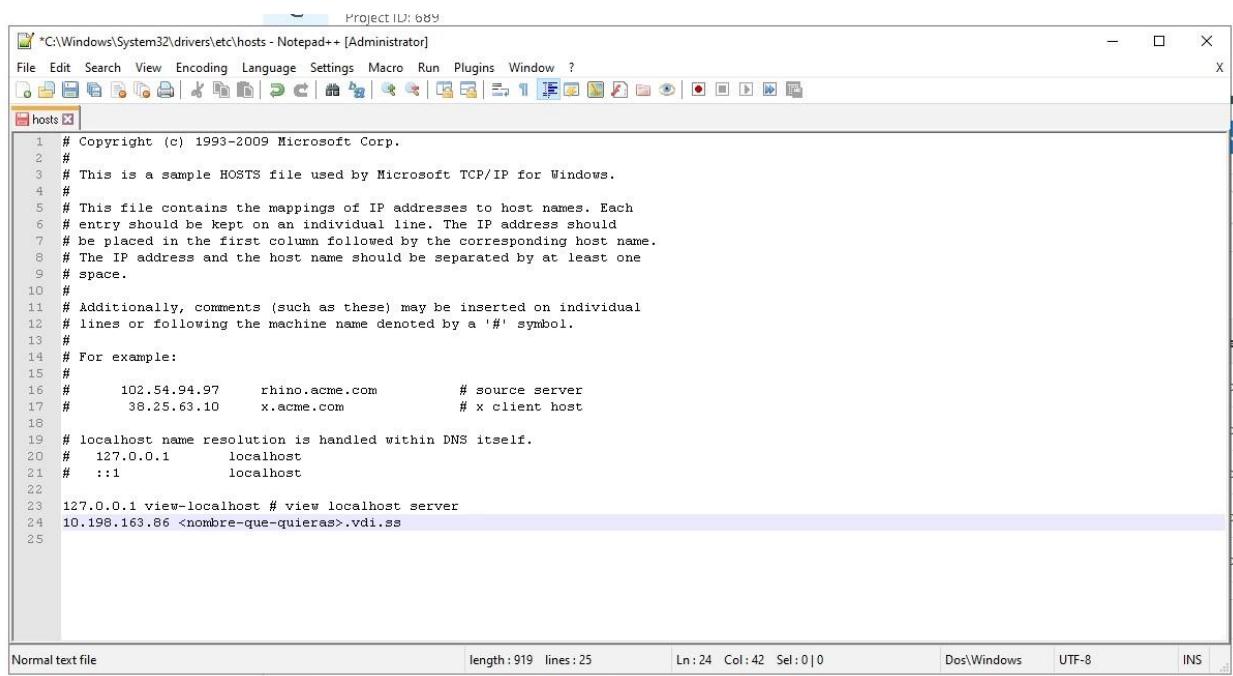


Ilustración 2 Ruta archivo hosts

A continuación, agregamos una nueva línea al final del documento. En ella pondremos la IP de nuestro equipo y, seguidamente, el nombre de host que le queramos dar.

Por ejemplo: 123.123.123.123 luis.vdi.ss



```
# Copyright (c) 1993-2009 Microsoft Corp.
#
# This is a sample HOSTS file used by Microsoft TCP/IP for Windows.
#
# This file contains the mappings of IP addresses to host names. Each
# entry should be kept on an individual line. The IP address should
# be placed in the first column followed by the corresponding host name.
# The IP address and the host name should be separated by at least one
# space.
#
# Additionally, comments (such as these) may be inserted on individual
# lines or following the machine name denoted by a '#' symbol.
#
# For example:
#
#      102.54.94.97    rhino.acme.com        # source server
#      38.25.63.10      x.acme.com            # x client host
#
# localhost name resolution is handled within DNS itself.
# 127.0.0.1      localhost
# ::1            localhost
#
127.0.0.1 view-localhost # view localhost server
10.198.163.66 <nombre-que-quieras>.vdi.ss
```

Ilustración 3 Ejemplo archivo hosts

1.4 Configuración RSA

Arrancamos RSA (IMPORTANTE, DEBES USAR RSA VERSIÓN 9.6.1) y creamos un nuevo Espacio de Trabajo (Workspace).

1.4.1 Activación de prestaciones RSA

Nos dirigimos a Ventana > Preferencias > General > Prestaciones y habilitamos todas las prestaciones de RSA. Aplicamos los cambios y reiniciamos RSA. Una vez hecho esto, podemos cambiar a la Perspectiva Java EE.

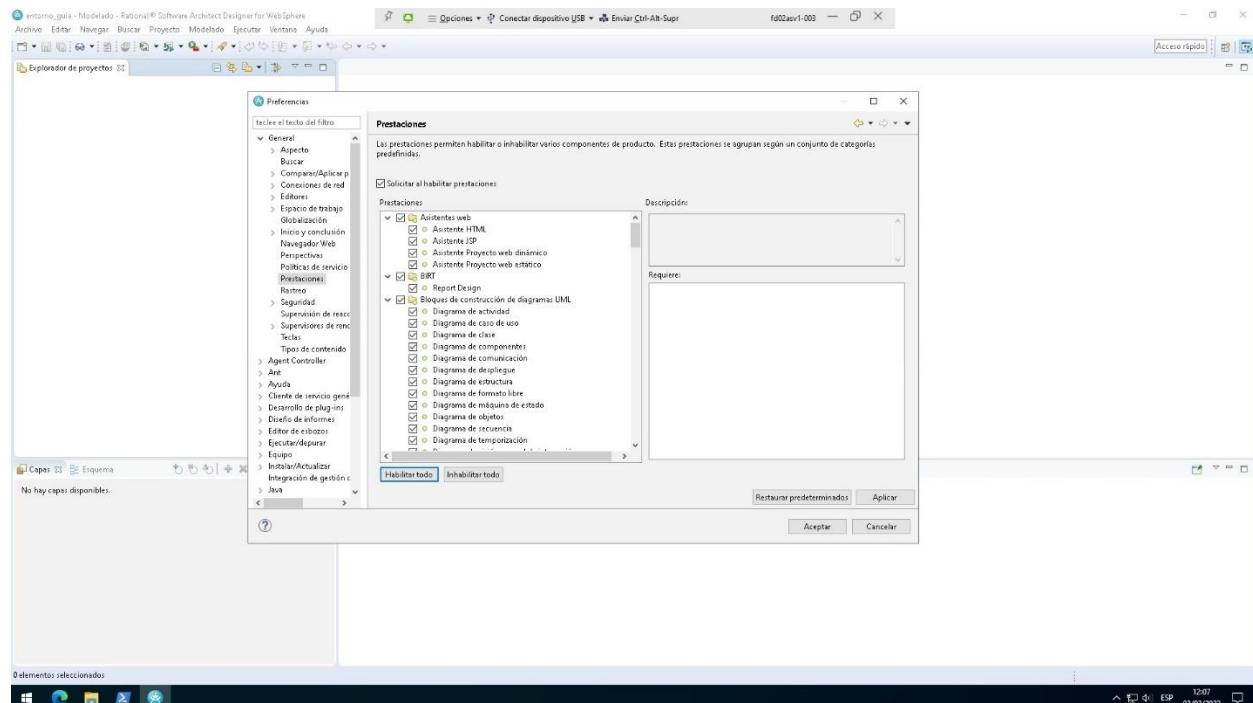


Ilustración 4 Activación de todas las prestaciones

1.4.2 Desactivación validación al construir

Una vez RSA vuelva a arrancar nos dirigimos a Ventana > Preferencias > Validación y desactivamos todas las validaciones de la columna Construir. Aplicamos los cambios y reiniciamos RSA.

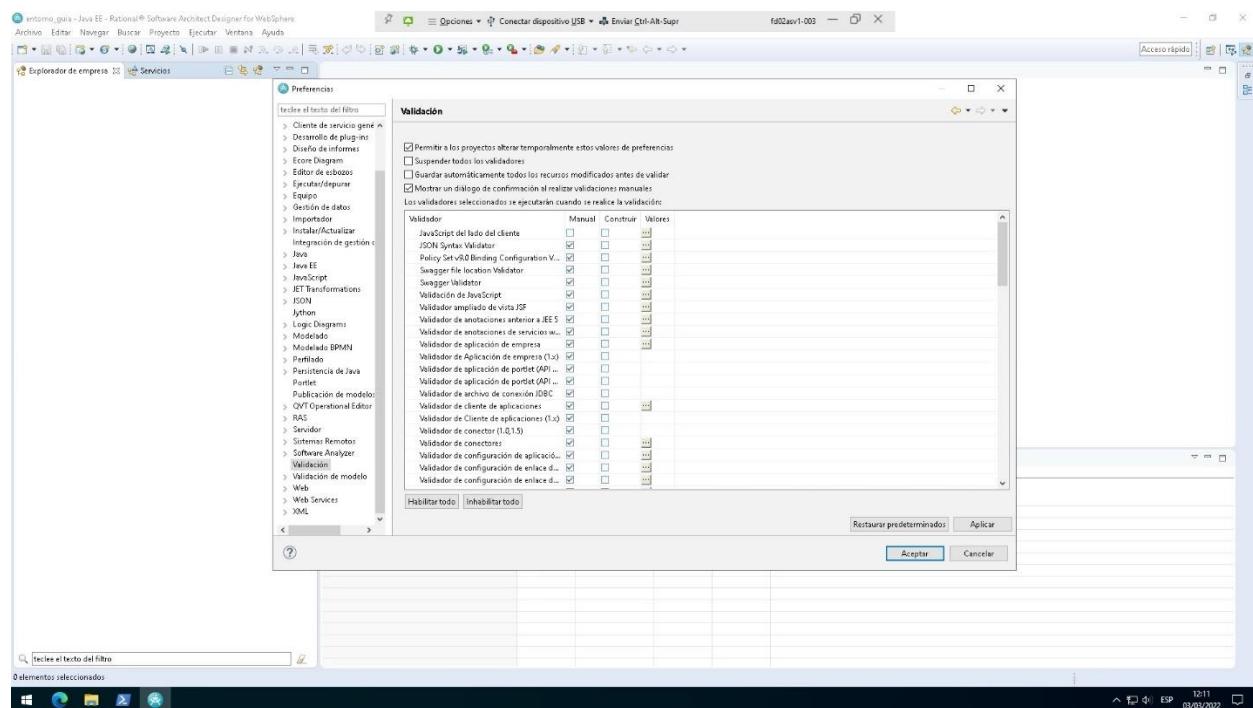


Ilustración 5 Desactivación validaciones automáticas

1.4.3 Configuración JDK

Ahora vamos a decirle a RSA qué JDK debe usar, para ello vamos a Ventana > Preferencias > Java > JRE Instalados, pinchamos en “Añadir” en la ventana que nos aparece seleccionamos “VM estándar” y damos “Siguiente”

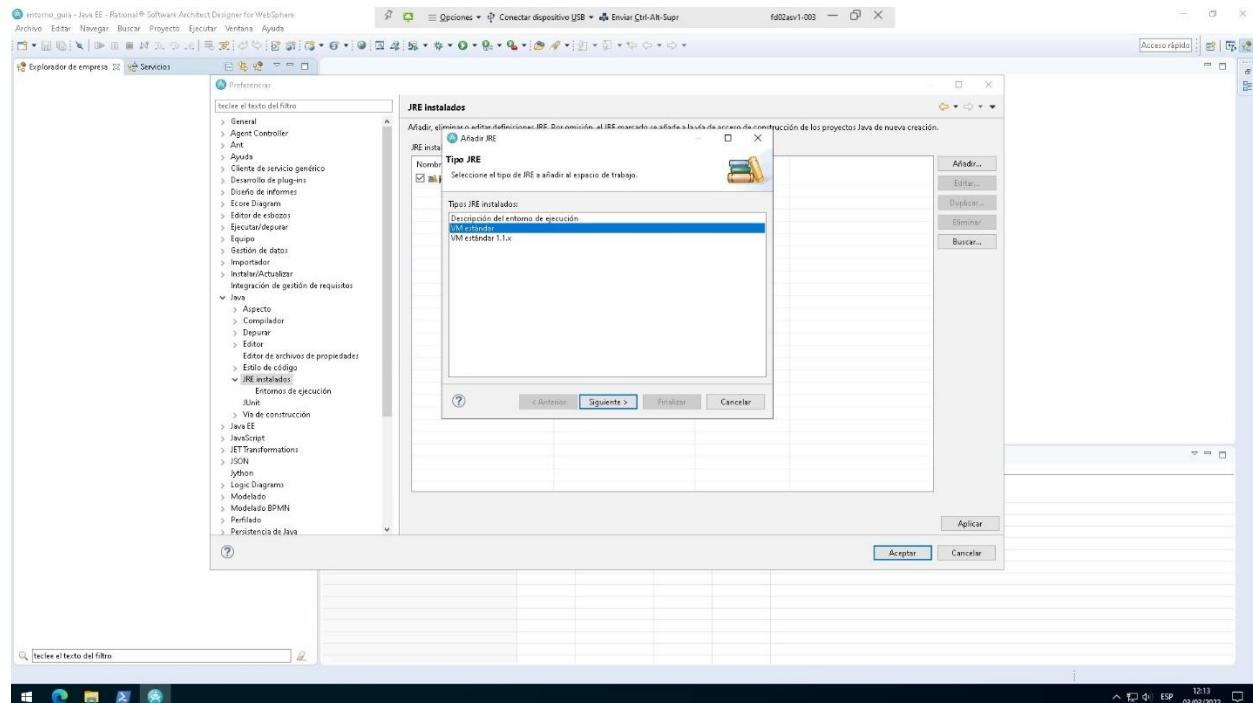
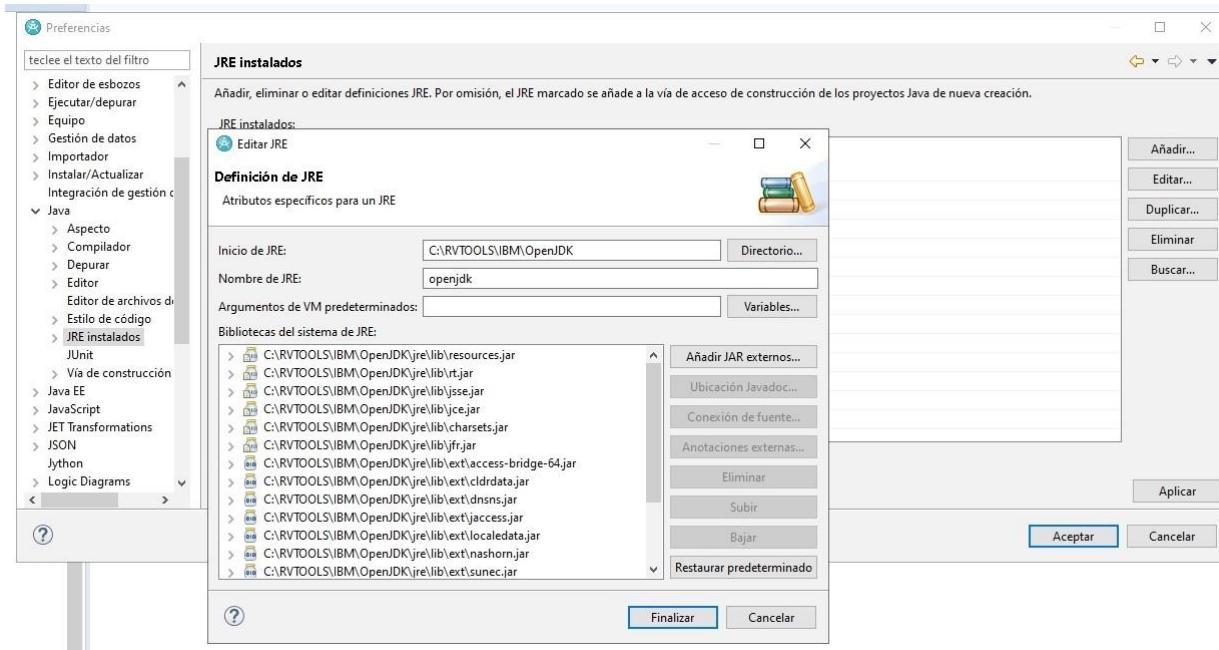
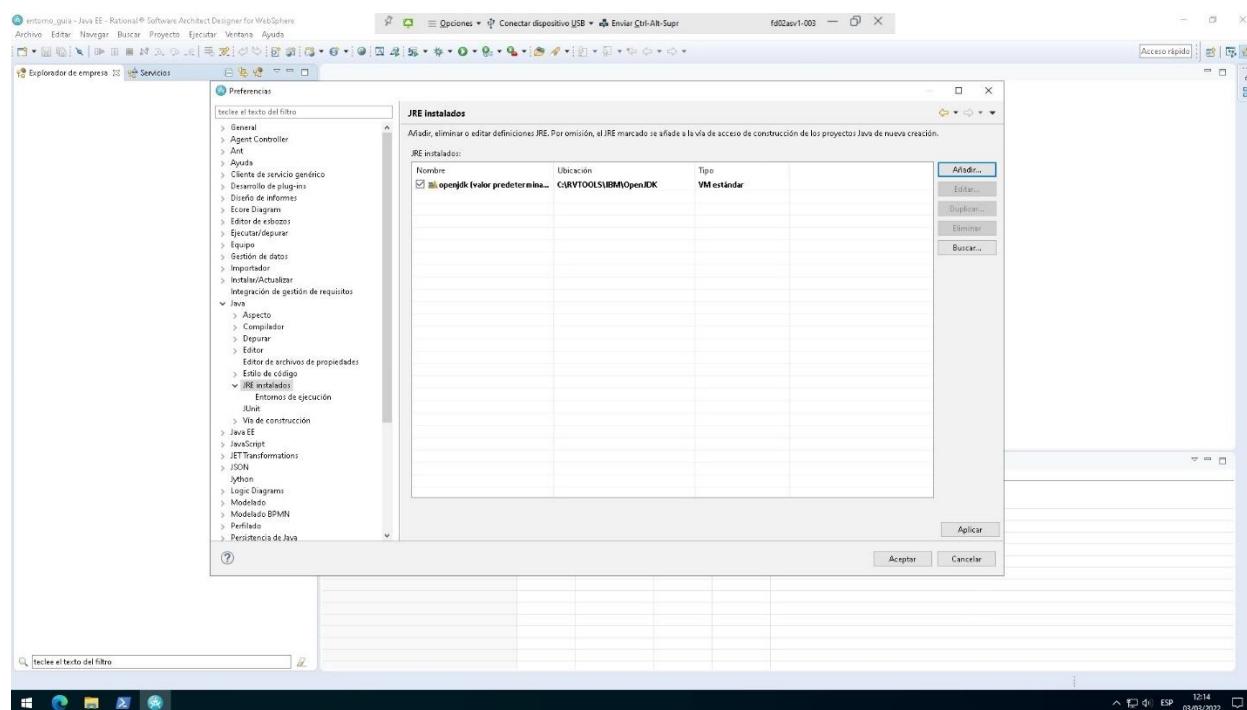


Ilustración 6

y nos saltará otra ventana donde tendremos que decir a RSA donde está la JDK, para ello pinchamos en “Directorio” y navegamos hasta la raíz de la carpeta donde instalamos la OpenJDK. También debemos cambiar el nombre de la JRE de OpenJDK a openjdk (dejar el nombre en minúsculas).



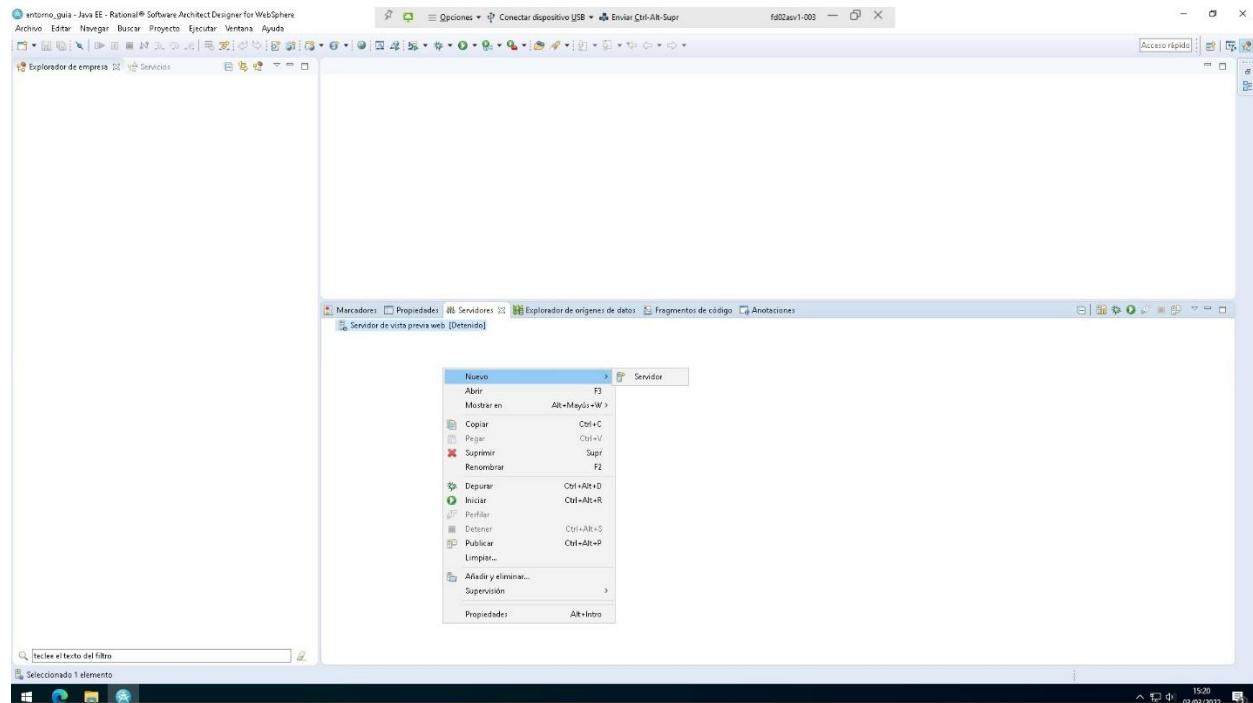
Una vez hecho esto pinchamos en “Finalizar” y ya tenemos la JDK configurada. (La JDK que viene por defecto en RSA la podemos borrar para evitar confusiones).



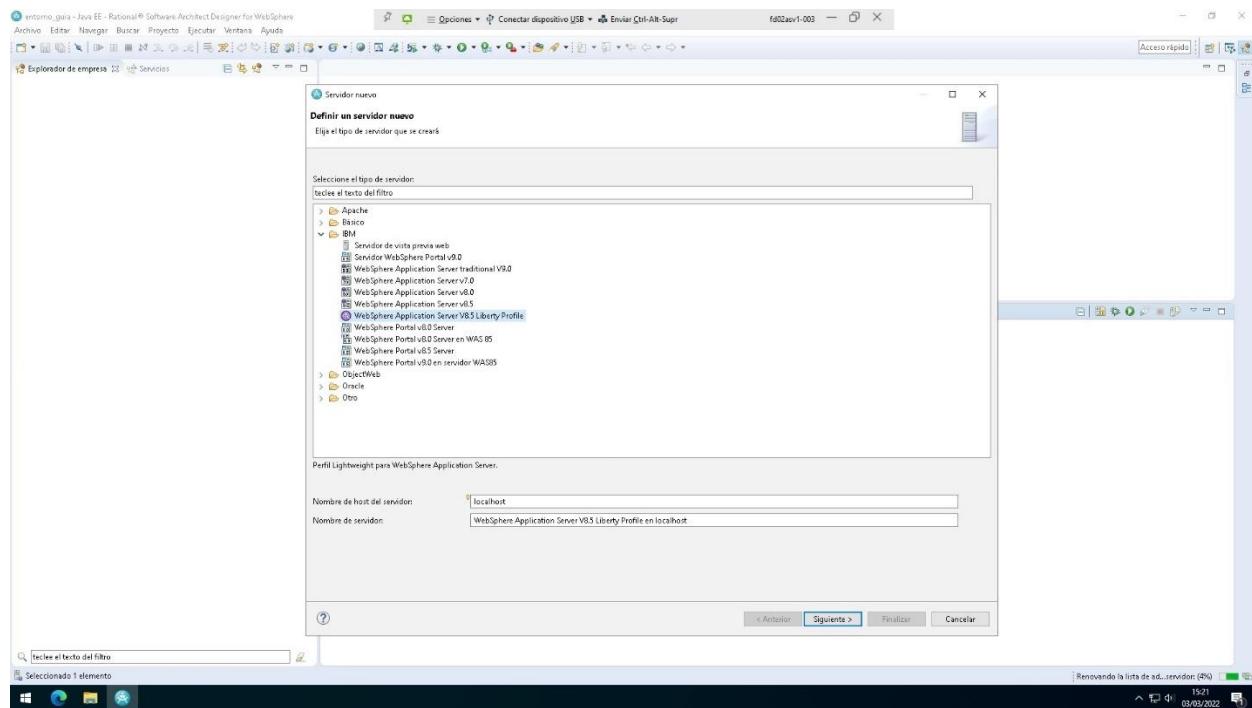
1.4.4 Configuración Liberty

Una vez configurada la JDK, estamos en disposición de configurar el servidor Liberty.

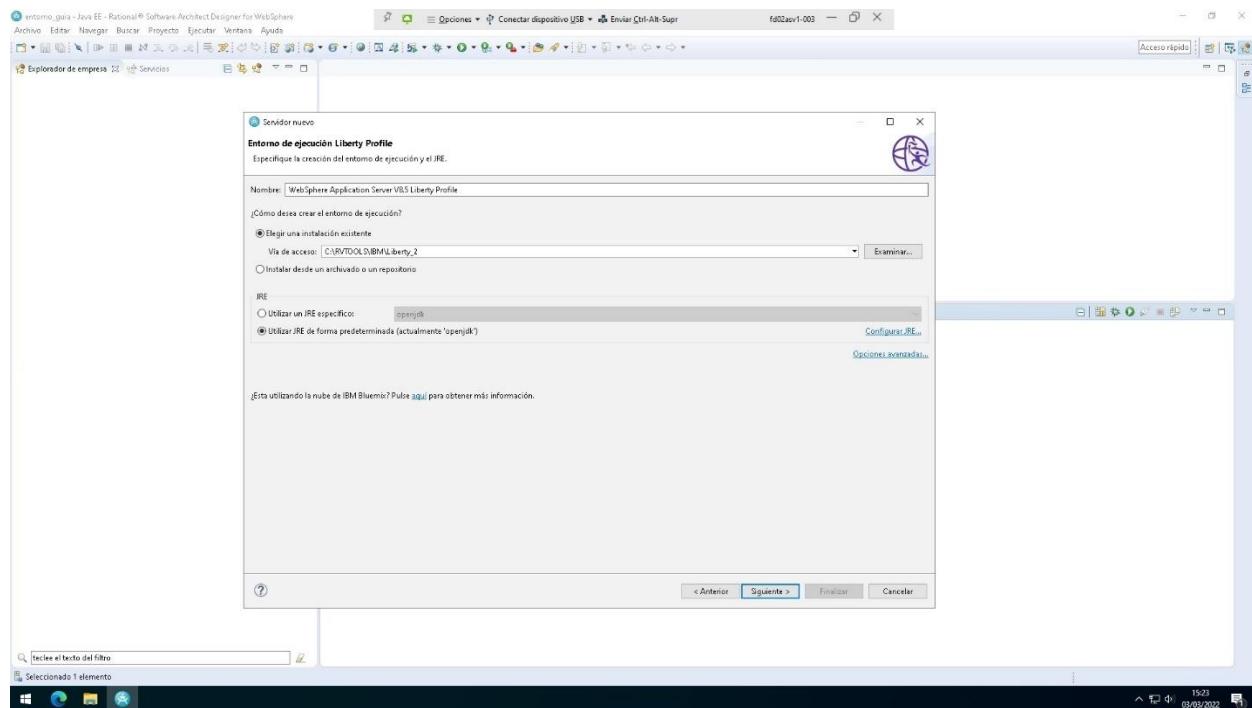
Como estamos en la perspectiva Java EE vamos a la pestaña “Servidores”, pinchamos botón secundario > Nuevo > Servidor.



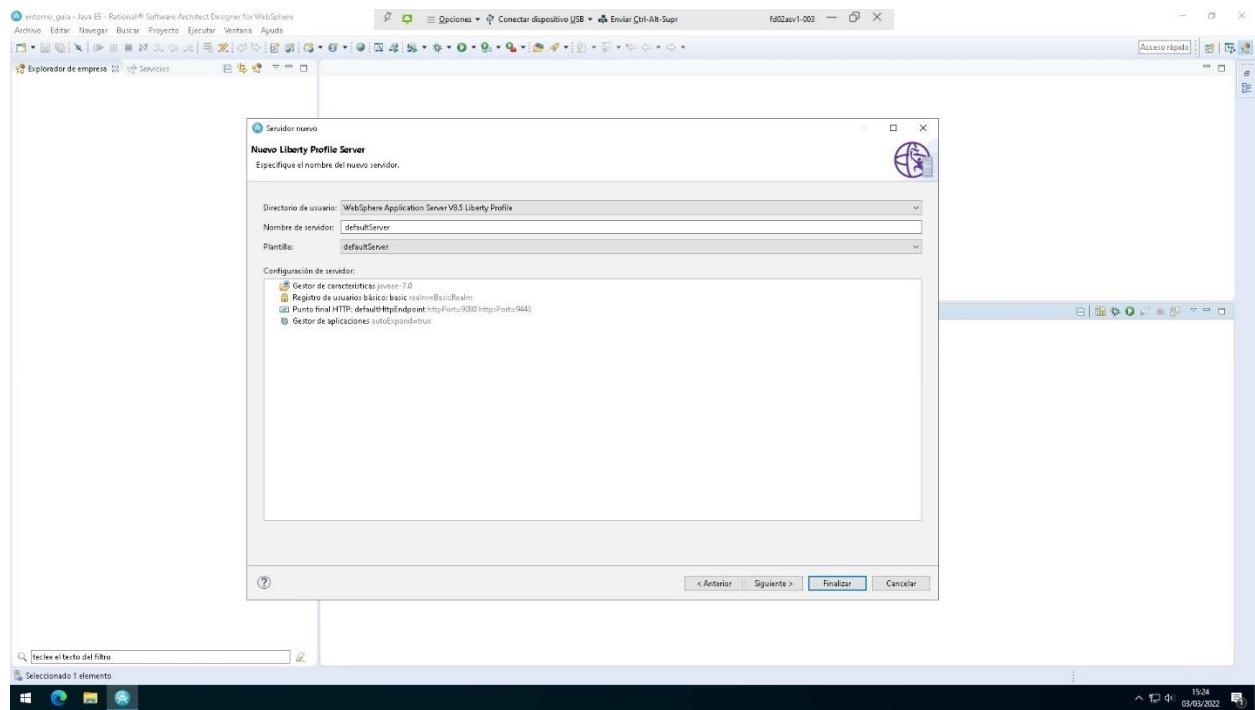
Seleccionamos la opción WebSphere Application Server V8.5 Liberty Profile



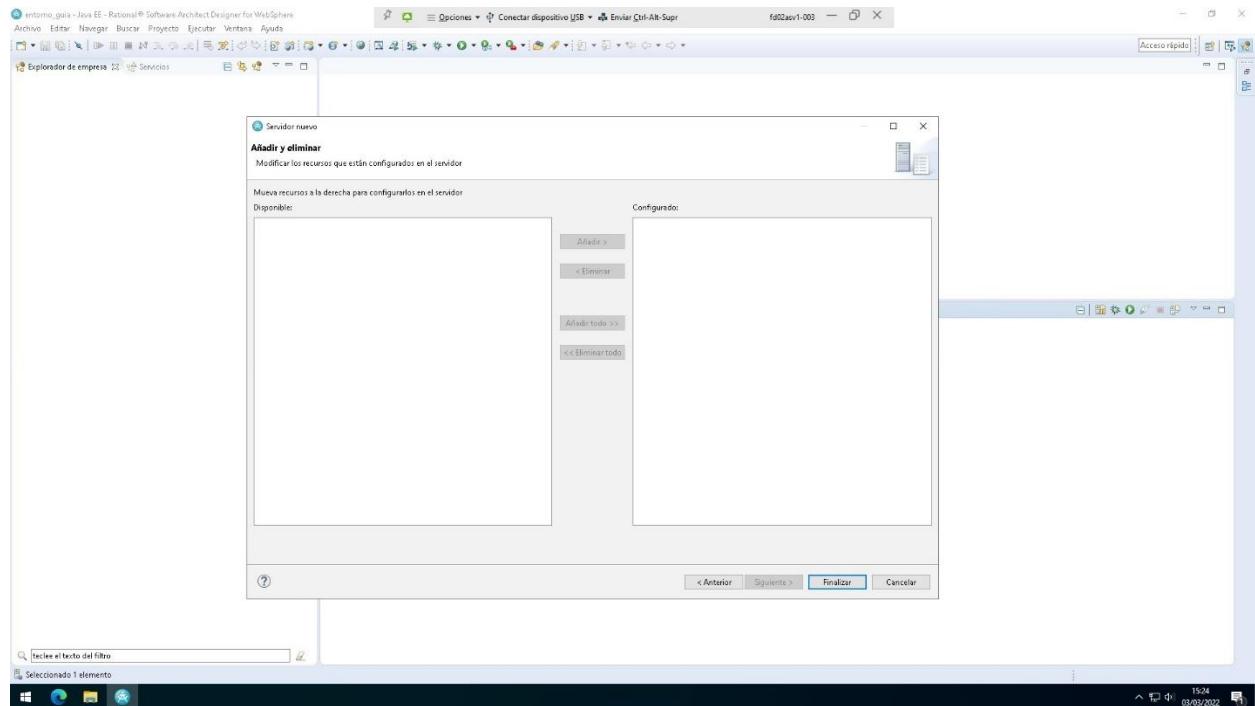
E indicamos la ruta en la que hemos desempaquetado Liberty y el jre sobre el que se ejecutará.



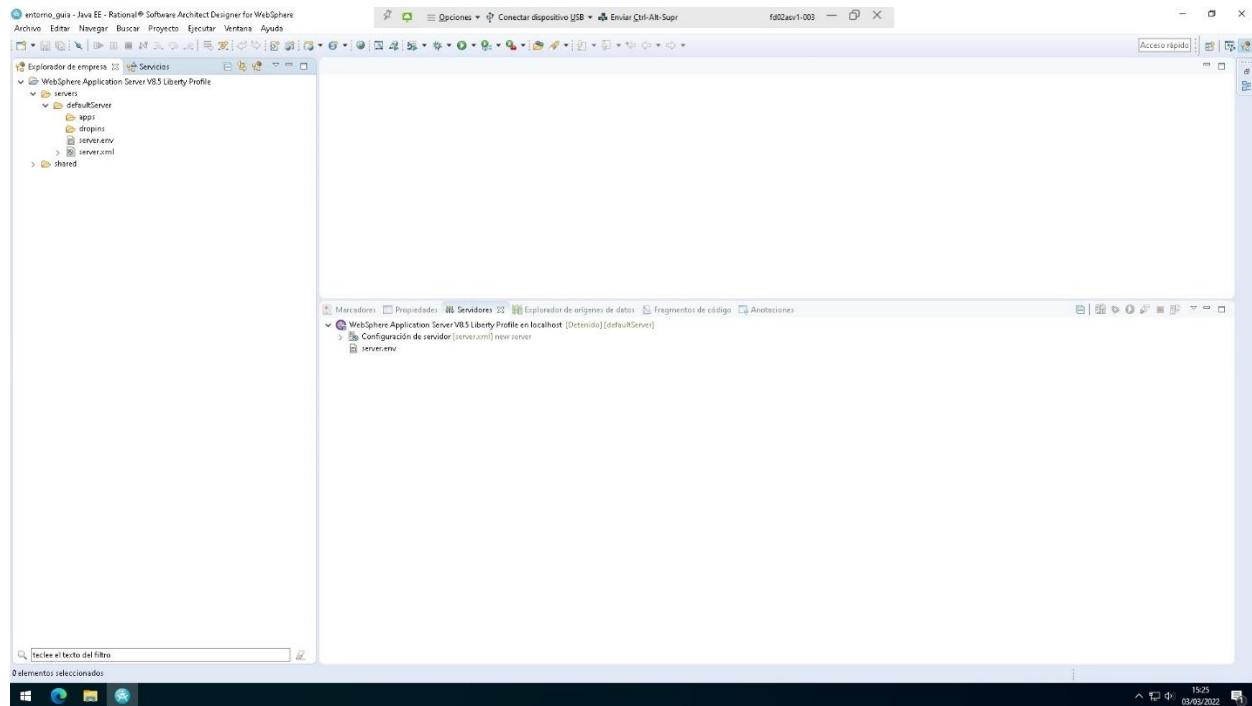
En la siguiente pantalla elegimos el nombre del servidor. No nos preocupamos, por ahora, por el contenido del server.xml



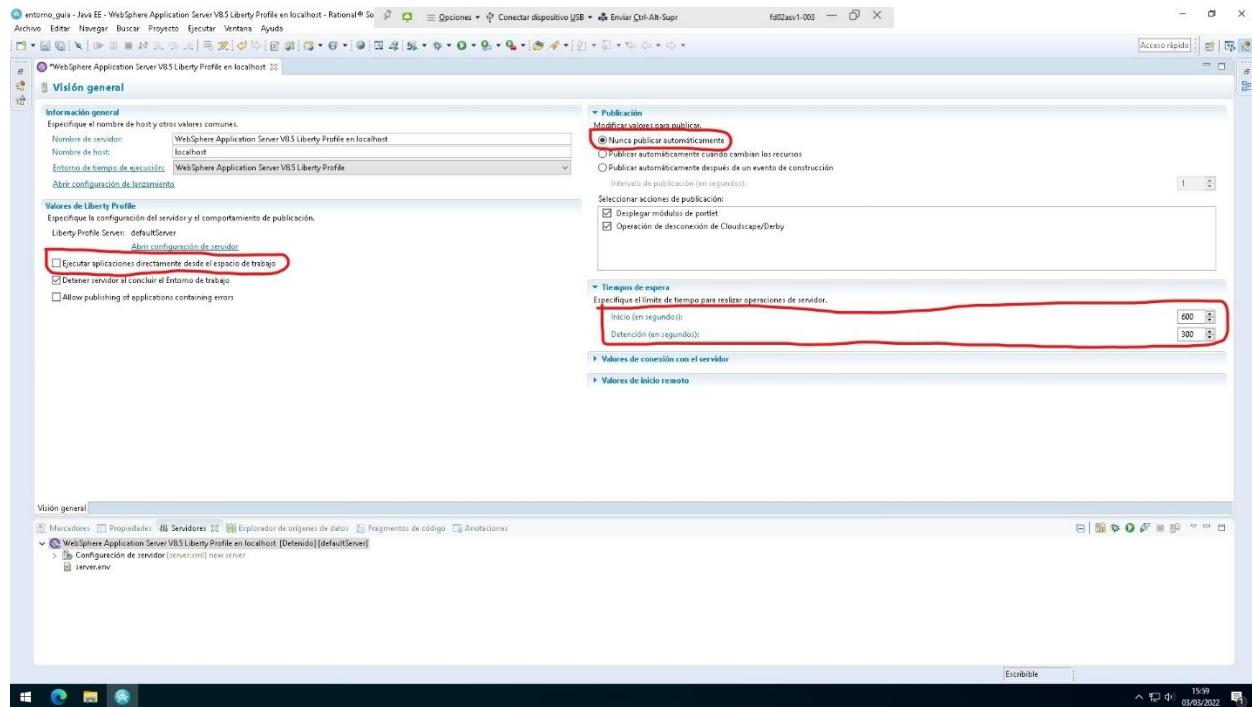
Llegamos a la pantalla de despliegues que, como con el server.xml, no nos preocupamos por ella.
Pinchamos en finalizar



Si toda ha ido bien, deberías tener algo semejante a esta captura



Ahora vamos a terminar de configurar el servidor, para ello damos doble click sobre el mismo y nos saltará una pantalla semejante a la siguiente, donde quitaremos el check a “Ejecutar aplicaciones directamente desde el espacio de trabajo”, seleccionaremos “Nunca publicar automáticamente” y subiremos los tiempos de arranque y parada (los de la captura sin orientativos)



1.4.5 Server.xml

Sólo nos quedaría poner al día el server.xml para que todo funcione correctamente.

Este fichero lo podemos descargar del repositorio, carpeta ConfiguracionServidor.

- Hay una parte común:
 - **server.xml**
- Otra parte específica por entorno (AT, DG...) que contiene los Datasources y JndiUrlEntries:
 - **server_<AT/DG>_ds_url.xml**
- Y otra con las entradas que dependen de rutas locales o de la configuración del entorno local de cada desarrollador:
 - **server_local.xml**

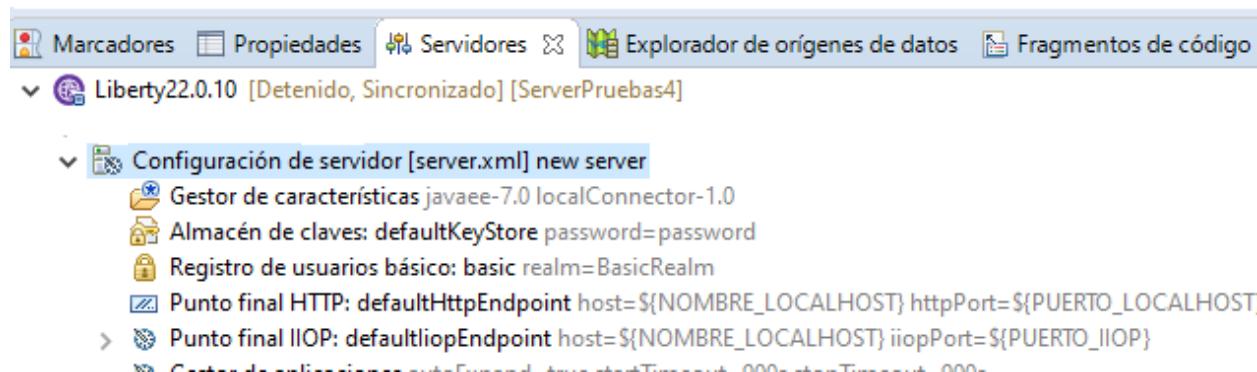
Descargamos el común, el local y el específico que necesitemos en nuestro caso.

Para que el común (server.xml) incluya adecuadamente al específico (AT o DG) que hayamos descargado, dejamos el específico renombrado a “server_ds_url.xml”.

Abrimos el server.xml que viene por defecto en el Liberty (<carpeta de instalación de Liberty>/usr/servers/defaultServer) y lo sobre-escribimos con el contenido del server.xml que hemos descargado del repositorio.

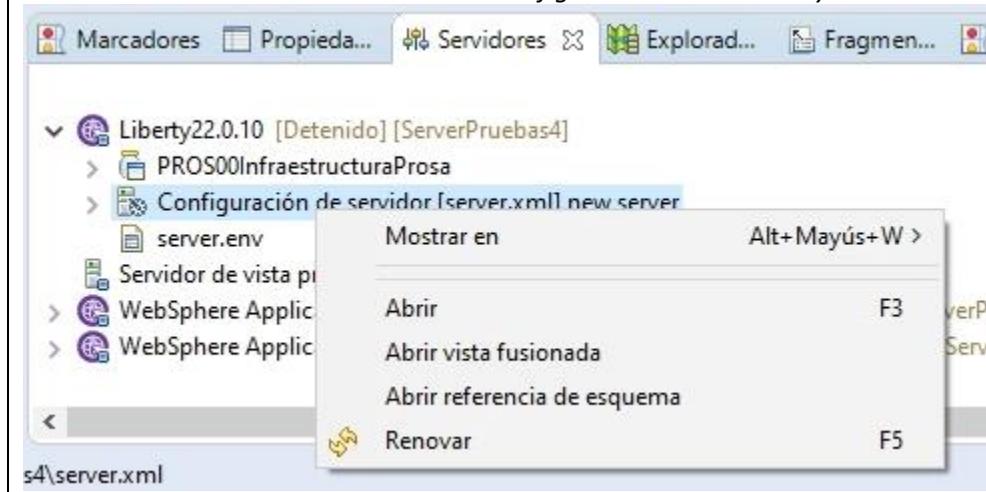
NOTA: El archivo server.xml ya existe previamente y hemos de sobre-escribirlo. Asegurarse de que el RSA está cogiendo el server.xml descargado y que no continua el original. Si se copia / pega el fichero en la carpeta, a veces hace caso omiso. Es mejor copiar el contenido del server.xml del repositorio, abrir el server.xml original y sustituir el contenido.

Si todo está bien, en la vista de Servidores, en el nodo de “Configuración de servidor”, si lo desplegamos, debemos ver algo similar a la siguiente imagen:



Si vemos que los elementos que cuelgan de Configuración de servidor, no se corresponden con la configuración del server.xml editado, es cuando RSA nos restaurará el server.xml anterior. En este caso,

hacer click con el botón derecho sobre “Configuración de servidor” y seleccionar “Renovar”:



NOTA: El archivo `server.xml` hace un *include* del archivo `server_ds_url.xml` y del `server_local.xml`

En el arranque del Liberty veremos una traza como la siguiente:

El proceso ha incluido el recurso de configuración: <path>/server_ds_url.xml

Lo siguiente es abrir el archivo `server_local.xml` y revisar la configuración para que se correspondan con nuestro entorno local.

Como mínimo, tendremos que e irnos a la sección “Variables”, variable NOMBRE_LOCALHOST, y poner el nombre de host que hayamos configurado previamente en el archivo hosts (xxxxx.vdi.ss)

```
<!-- ===== -->
<!-- Variables -->
<!-- ===== -->

<!-- Variables -->
<variable name="ORACLE JDBC DRIVER PATH" value="C:\oracle\product\11.2.0\client_1\jdbc\lib"/>
<variable name="NOMBRE_LOCALHOST" value=<nombre que configuramos en el paso 3.3 de la guía>.vdi.ss" />
<variable name="PUERTO_LOCALHOST" value="9081" />
<variable name="PUERTO_LOCALHOST_SEGURO" value="8443" />
<variable name="PUERTO_AJENO" value="9080" />
<variable name="PUERTO_AJENO_SEGURO" value="9443" />
<variable name="PUERTO_IIOP" value="1809" />
<variable name="PUERTO_IIOP_SEGURO" value="1833" />
```

NOTA: el servidor Liberty arrancará usando los puertos indicados en las variables PUERTO_LOCALHOST y PUERTO_LOCALHOST_SEGURO. Si nos diese un error al arrancar porque dichos puertos están en uso, cambiarlos en dichas variables por unos libres

Y con esto hemos terminado de configurar el servidor Liberty de desarrollo.

NOTA: En alguna ocasión, puede que el RSA restaure el `server.xml` a una versión anterior, sobre todo puede ocurrir cuando se añaden o eliminan aplicaciones para desplegar en el Liberty. Se recomienda

hacer una copia de seguridad de los archivos server.xml, por si pasa este comportamiento anómalo, poder restaurarlo sin pérdida

1.4.6 Keystore de Liberty

La primera vez que se arranca el Liberty, creará un almacén de certificados en la ruta (<carpeta de instalación de Liberty>/usr/servers/defaultServer/resources/security, con un password de nuestra elección.

Esta password quedará reflejada en el server.xml en una línea como la siguiente:

```
<keyStore password="xxxxx"/>
```

Si esta línea no existe previamente en el server.xml, al arrancar el Liberty saldrá un popup solicitando la entrada de la contraseña. Podemos poner la que queramos y automáticamente insertará dicha línea en el server.xml con la contraseña introducida.

Si antes de arrancar, ya tenemos esa línea informada, utilizará la contraseña indicada y no la pedirá en el popup.

NOTA: Si al arrancar el servidor vemos trazas con este error:

[ERROR] CWPKI0033E: El almacén de claves situado en C:/IBM/wlp_2022.10/wlp/usr/servers/defaultServer/resources/security/key.p12 no se ha cargado debido al siguiente error: parseAlgParameters failed: ObjectIdentifier()

...

[ERROR] CWWKS9582E: Los atributos sslRef [defaultSSLConfig] que el elemento orb con el ID defaultOrb necesita no se han resuelto en 10 segundos. En consecuencia, las aplicaciones no se iniciarán.

Entrar en el servidor, ir la carpeta "resources", eliminarla y re-arrancar. Si todo va bien, deberíamos ver que se regenera la carpeta "resources" y salen las siguientes trazas:

The screenshot shows the RAD interface with the 'ServerPruebas' server selected. In the left pane, the 'resources' folder under 'security' is expanded, showing 'key.p12' and 'ltppa.keys'. In the center pane, the 'Código fuente' tab of the 'server.xml' editor is open, displaying XML code related to security settings. The right pane shows the 'Logs' tab with several audit log entries in Spanish. One entry from 'AUDIT [] CMWK0082A: El proceso ha incluido el recurso de configuración: C:/RWTTOOLS/IBM/wlp_2022.10/usr/servers/ServerPruebas2/server.ds.url.xml' is highlighted. Another entry from 'AUDIT [] CMWK100011: El servidor de nombres CORBA está ahora disponible en corbaloc://localhost:2809/NameService' is also visible. The logs indicate the successful creation of the LTPA key and the SSL certificate.

Y ya no saldrá el error

Si se necesitase añadir algún certificado en este almacén, se puede hacer con el comando keytool disponible en la carpeta bin de la OpenJDK

Ej:

```
C:\OpenJDK\bin\keytool -import -alias RAIZ_DES_ACGISS2021.crt -file  
C:\CERTIFICADOS_COPIA_AUTENTICA_LIBERTY\RAIZ_DES_ACGISS2021.crt -storetype JKS -keystore  
C:\Liberty\usr\servers\defaultServer\resources\security\key.p12 -storepass password -noprompt
```

En verde, el alias del certificado que deseamos instalar.

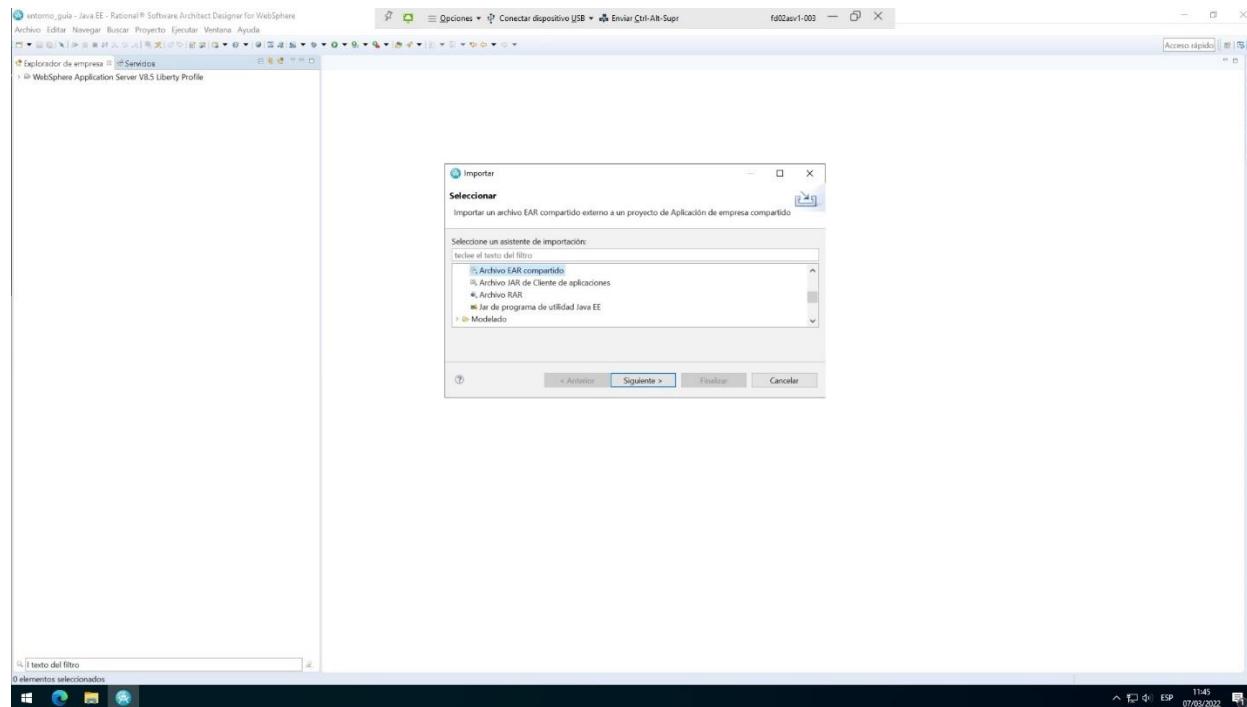
En azul, la ruta absoluta del certificado que deseamos instalar, incluyendo el propio nombre del archivo del mismo.

En rosa, la ruta absoluta del almacén de certificados.

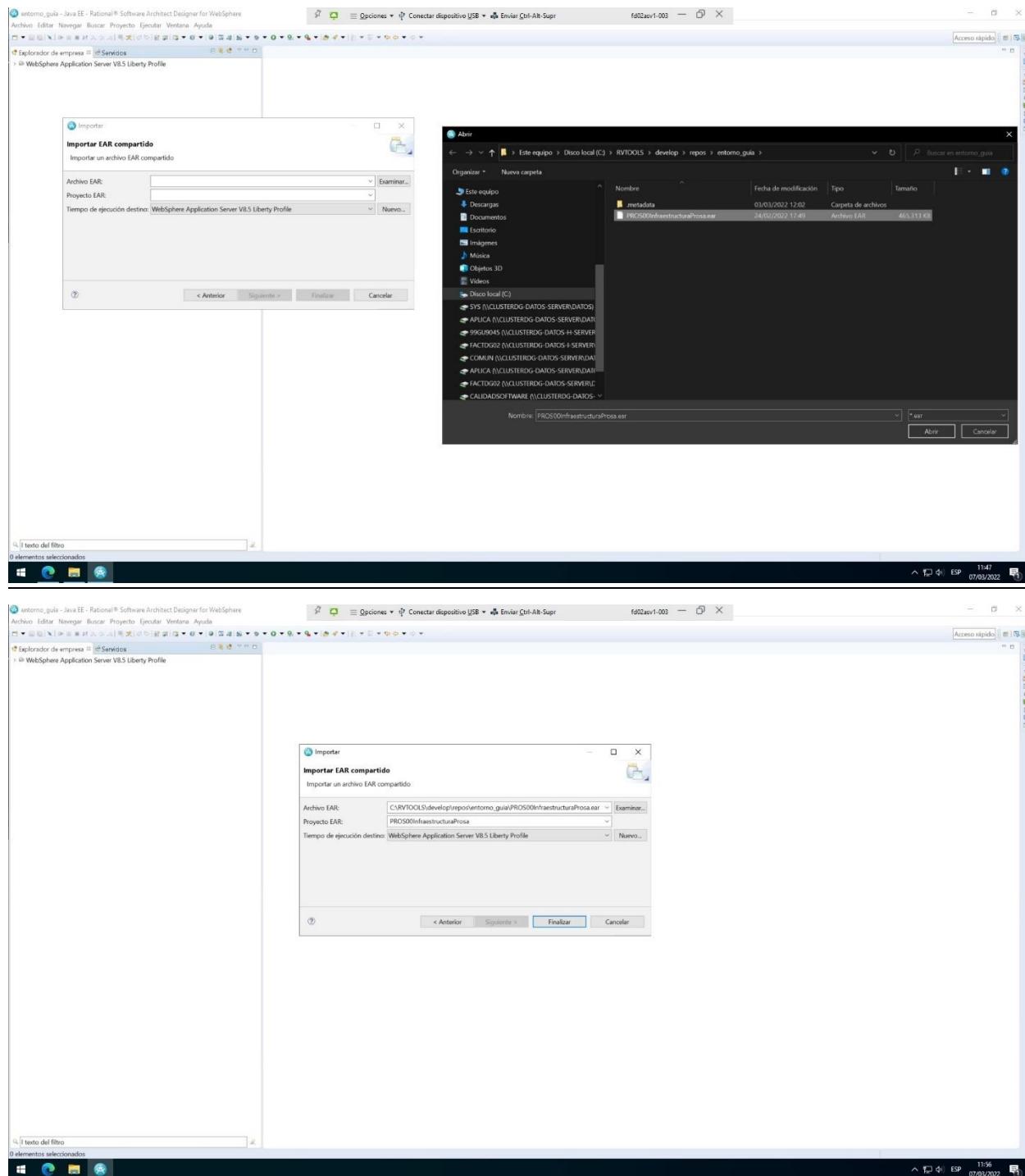
1.4.7 Importar EAR de infraestructura

Ahora vamos a importar el ear de arquitectura.

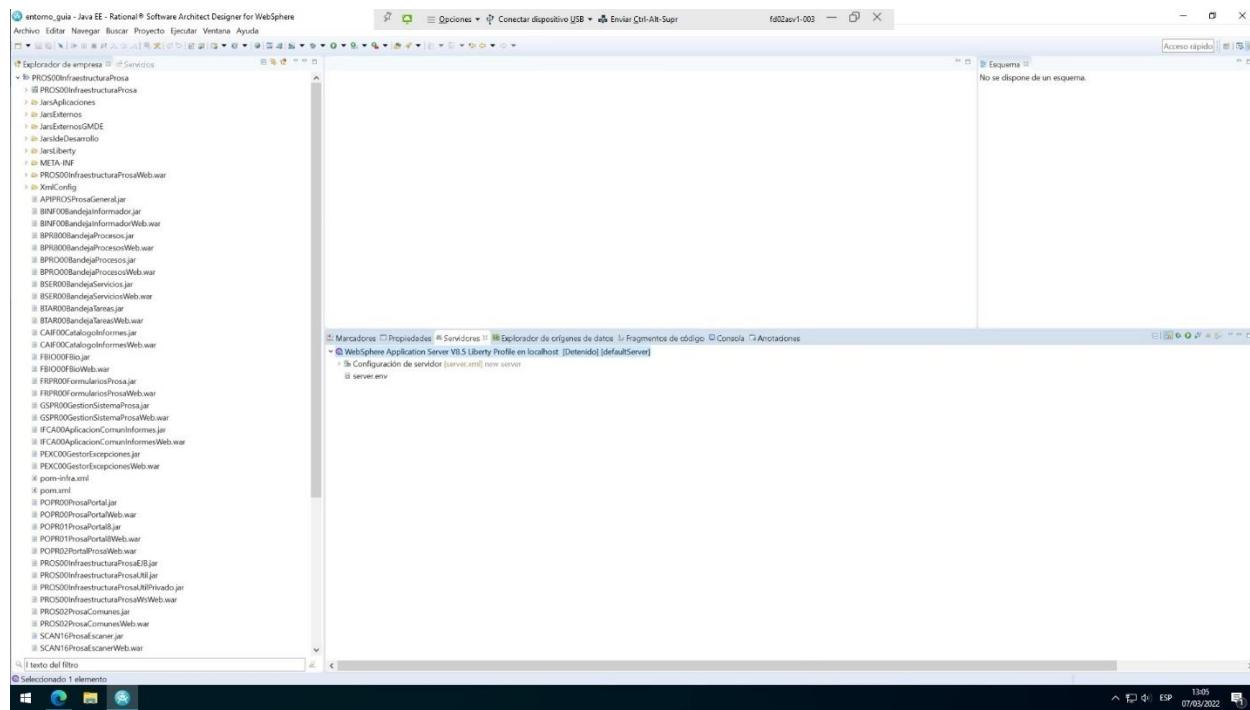
Nos descargamos el EAR desde el **repositorio**, carpeta Infraestructura/<versión>, y lo ponemos en la carpeta del Workspace. Ahora vamos a Archivo > Importar y nos saldrá la siguiente ventana, en la que buscaremos “EAR Compartido”



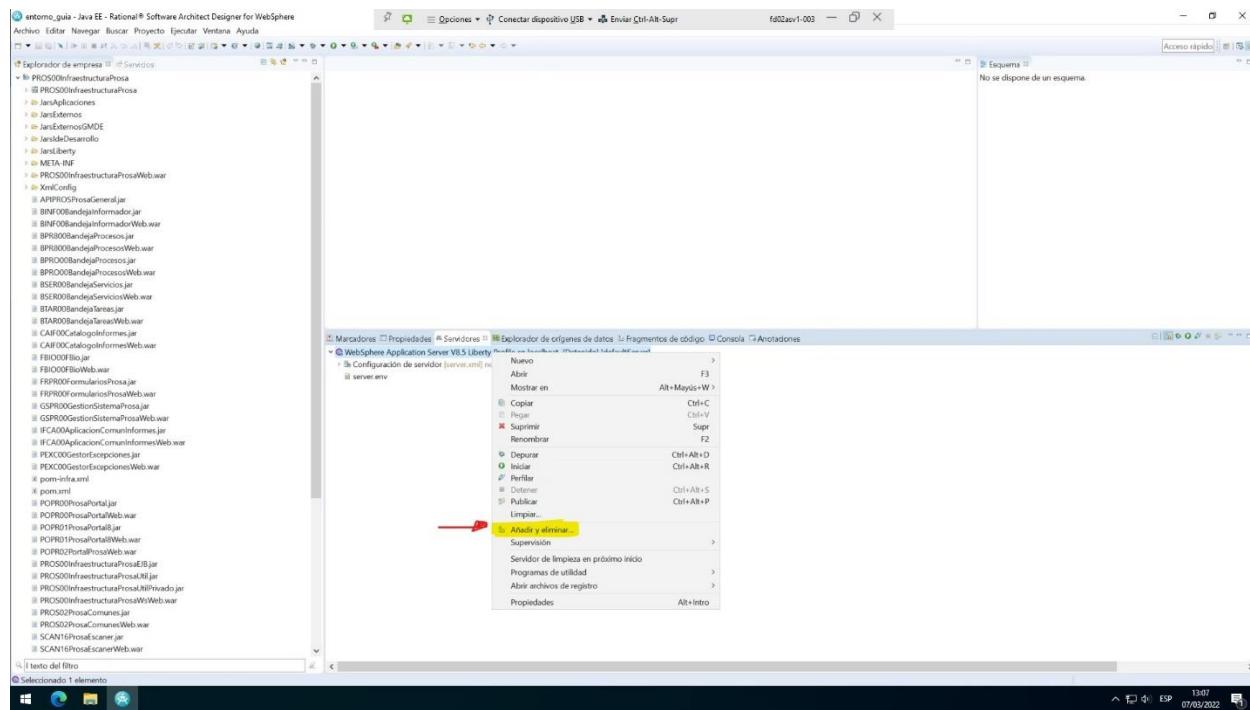
Navegamos hasta la carpeta del workspace y lo importamos el EAR



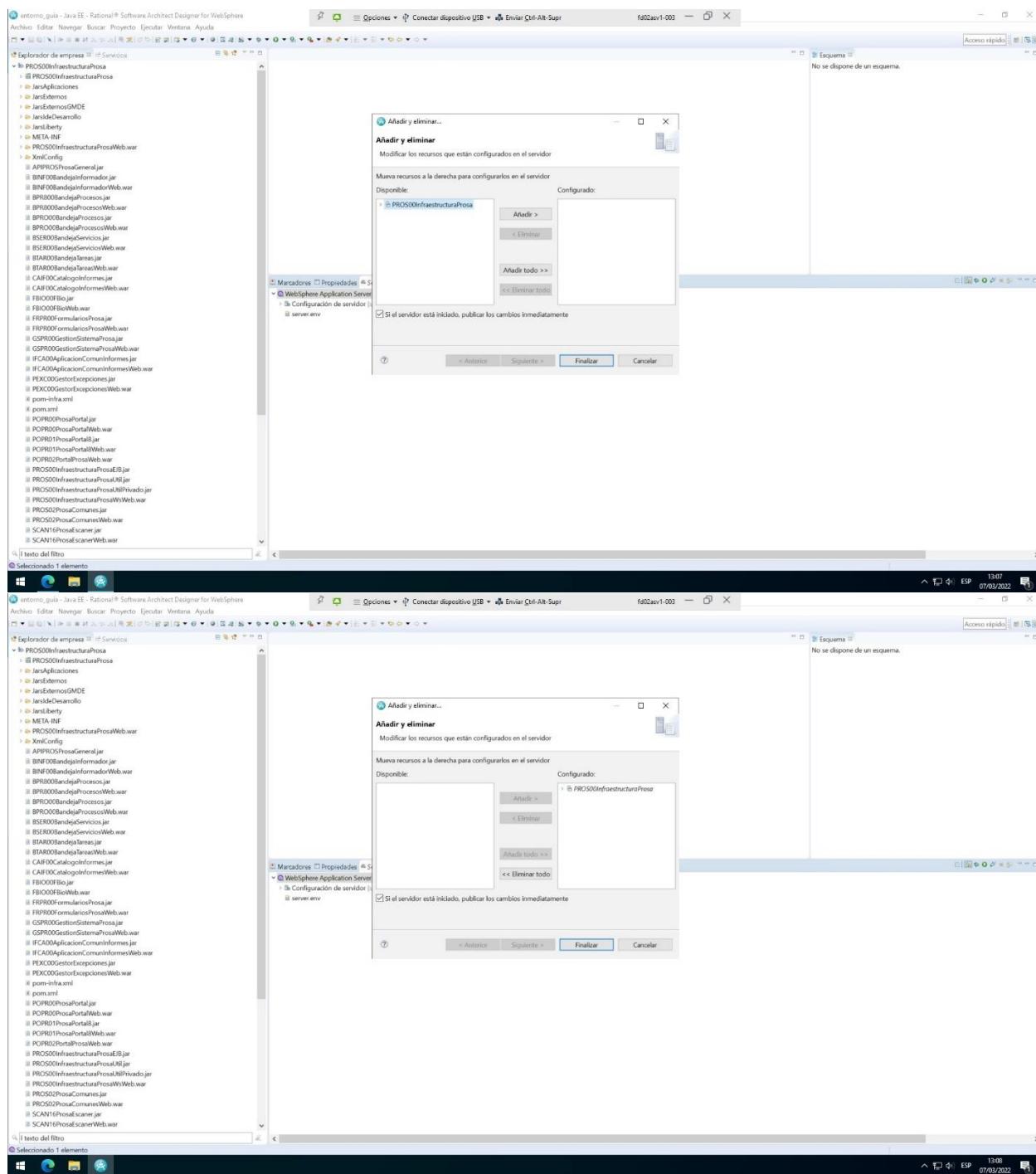
Una vez importado el EAR al Workspace no tendrá quedará una cosa así



Ahora ya estamos en condición de añadir la infraestructura a Liberty, para ello nos vamos a la pestaña de Servidores y hacemos click con el botón secundario sobre el Servidor Liberty y nos vamos a "Añadir y Eliminar"



Añadimos el EAR al Servidor



Y ya tenemos la infraestructura montada en el servidor.

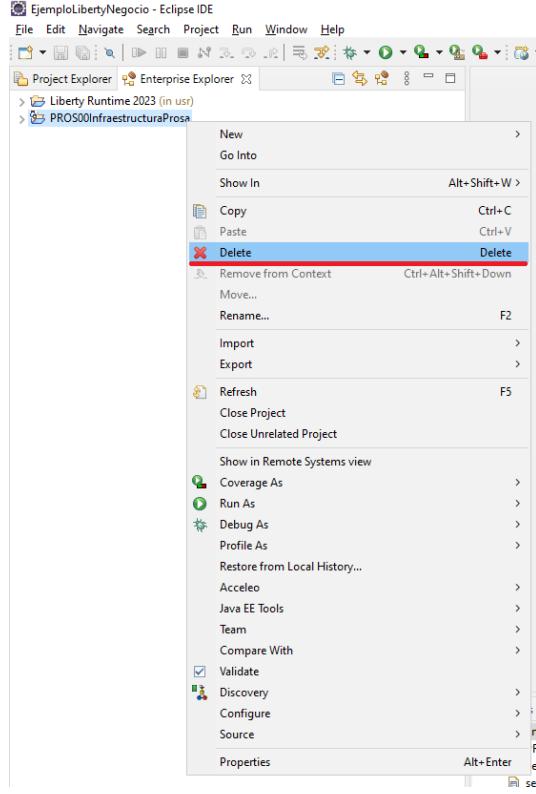
Al arrancar la infraestructura, podemos observar que aparecen unas trazas con un mensaje de advertencia "SRVE9967W", de la forma:

Se deben a una característica interna de la propia arquitectura y no afectan en absoluto al buen funcionamiento de la misma.

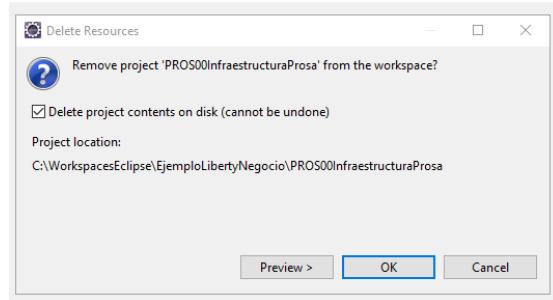
1.4.8 Migrar/cambiar a otra versión de EAR de infraestructura.

Para migrar/cambiar a otra versión de ear de infraestructura, el proceso será muy similar al del apartado anterior y habrá que seguir los siguientes pasos:

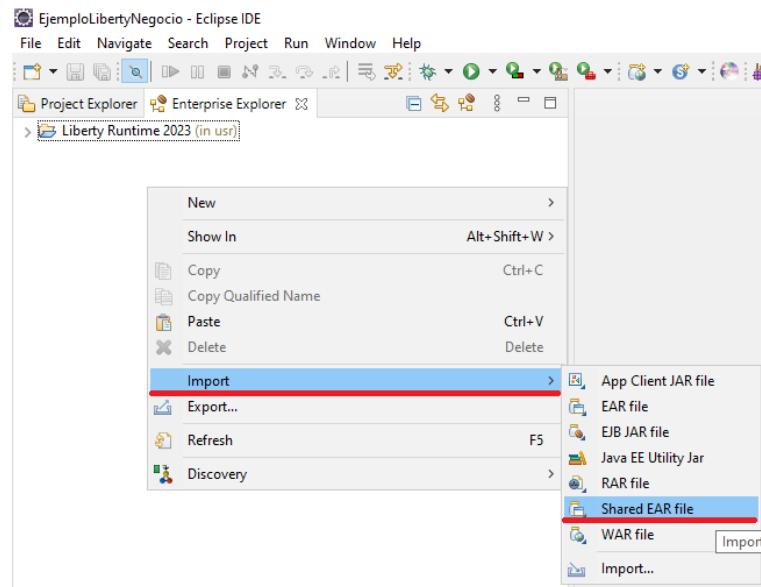
Eliminamos la infraestructura actual, haciendo click derecho sobre la misma y pulsando "Delete":



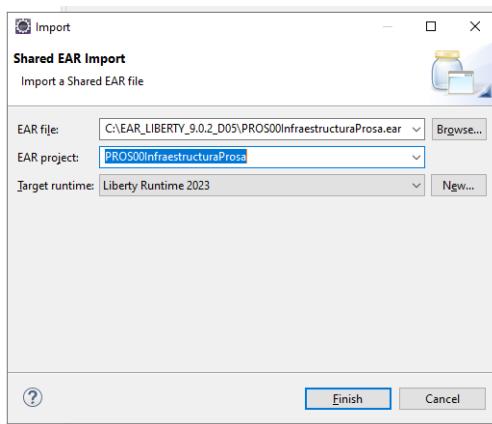
Eliminamos también la infraestructura del disco duro:



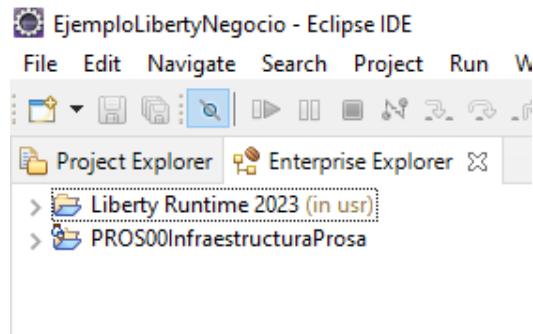
Importamos la nueva versión de infraestructura deseada, haciendo click derecho sobre la perspectiva “Enterprise Explorer”, importamos “Shared EAR file”:



Seleccionamos el nuevo ear de infraestructura:



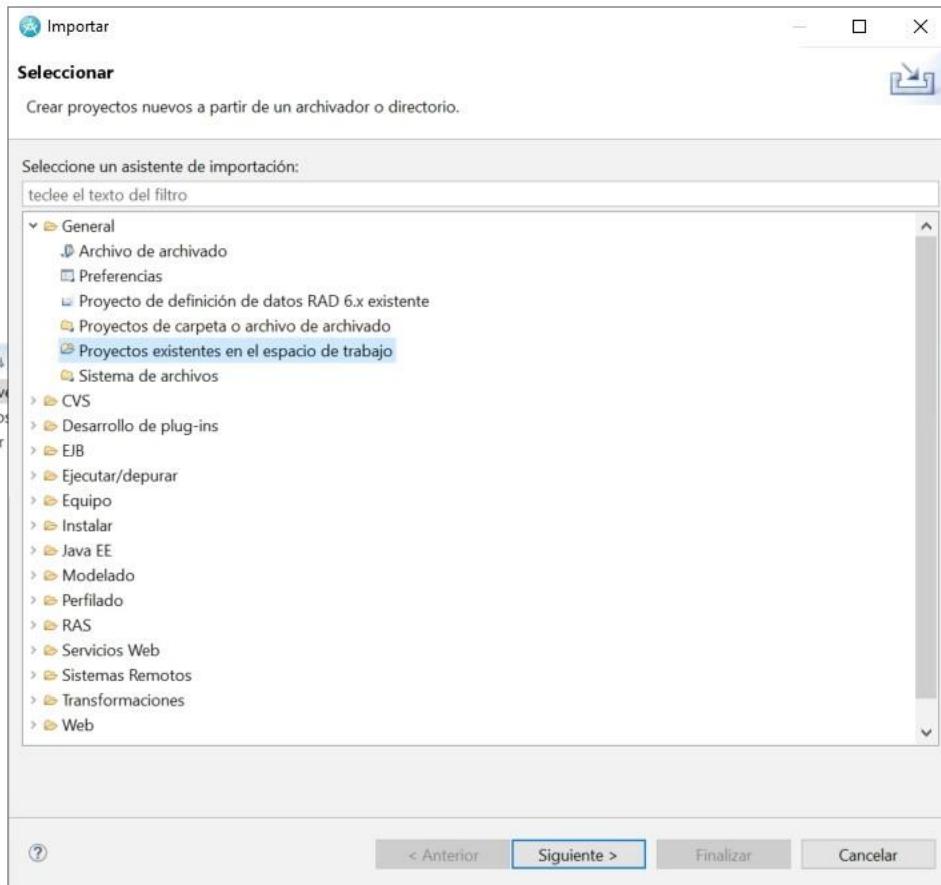
Lo tendremos en nuestro workspace:

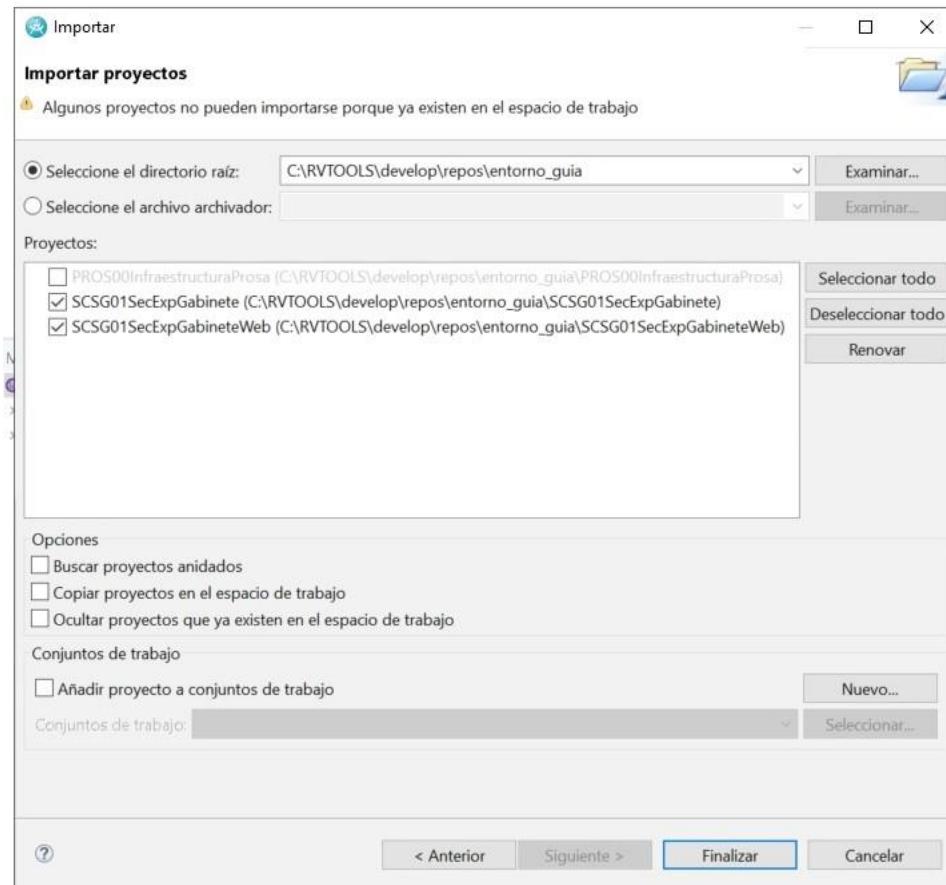


1.4.9 Importar aplicación de trabajo.

Y por fin podemos importar nuestra aplicación de trabajo (en nuestro caso SCSG01SecExpGabinete y su compañero SCSG01SecExpGabineteWeb). Ponemos ambas carpetas en el Workspace y las importamos.

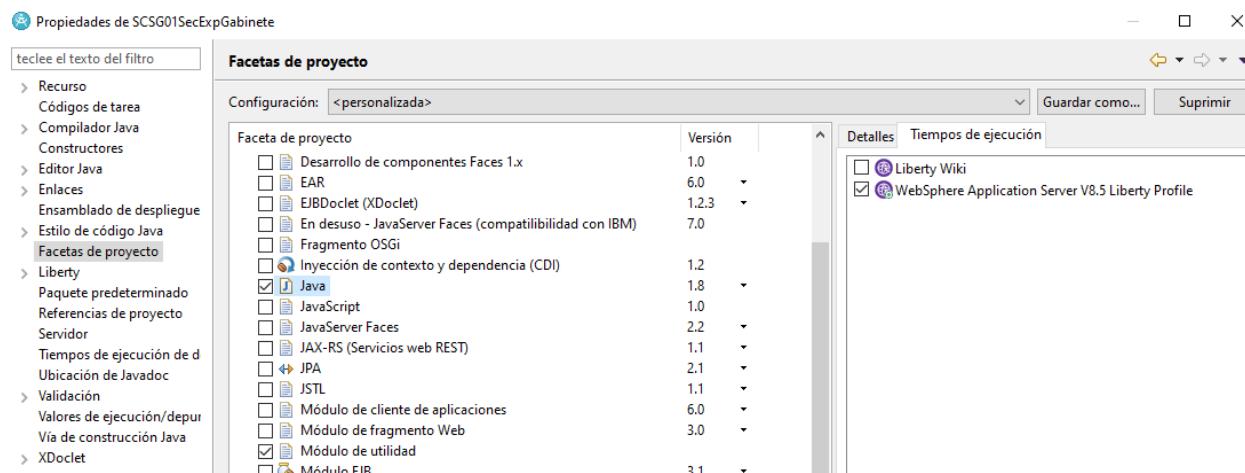
Archivo > Importar > General > Proyectos existentes en el espacio de trabajo



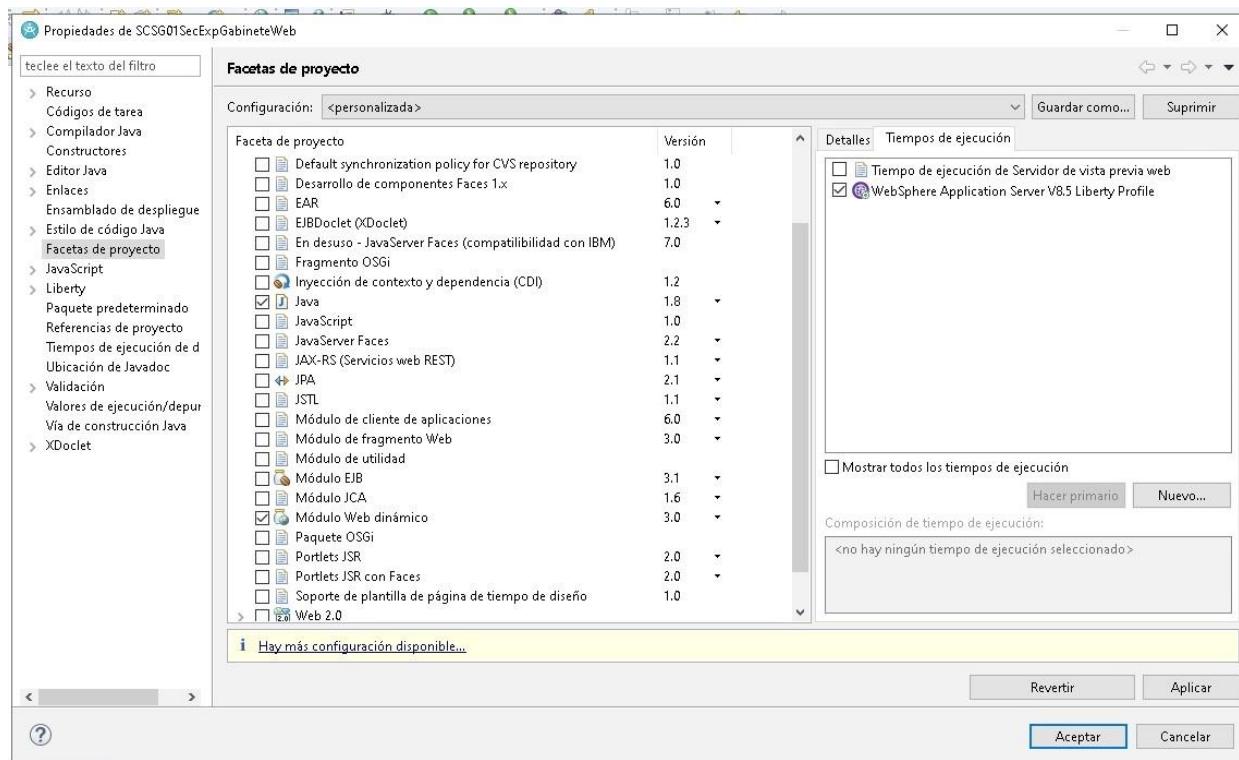


1.4.9.1 Facetas

Ahora pasamos a definir las facetas de ambos proyectos. Botón secundario sobre SCSG01SecExpGabinete > Propiedades > Facetas de proyecto y seleccionamos Java versión 1.8 y Módulo de utilidad. En Tiempos de ejecución, seleccionamos el Liberty



Para SCSG01SecExpGabineteWeb debemos seleccionar las facetas Java y Módulo Web dinámico



NOTA: Puede ser que los proyectos importados ya contengan configuración de las facetas, y dé otros errores de facetas o no deje editarlas en el RSA.

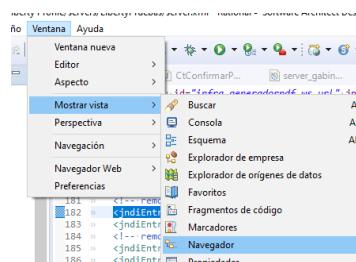
Lo importante es dejar la faceta de Java con la versión 1.8 y reemplazar en la pestaña de Tiempos de ejecución si teníamos otro servidor como WAS, por el Liberty

Si nos da error por estas facetas, las podemos deshabilitar:

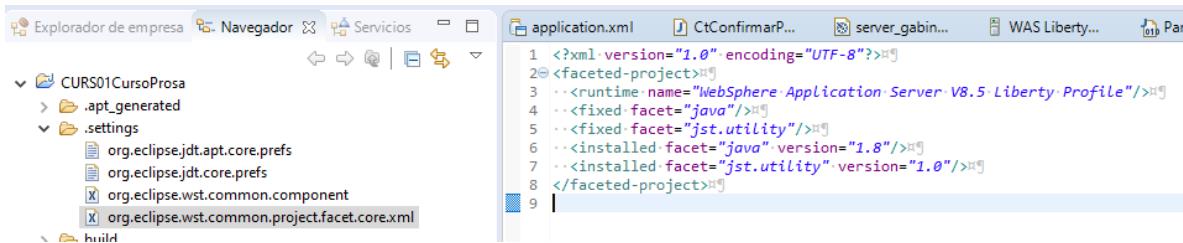
-  Web de WebSphere (Ampliado)
-  Web de WebSphere (Coexistencia)

En caso que no nos deje editar las facetas desde el RSA, podemos editar los ficheros a mano y renovarlos luego en el entorno:

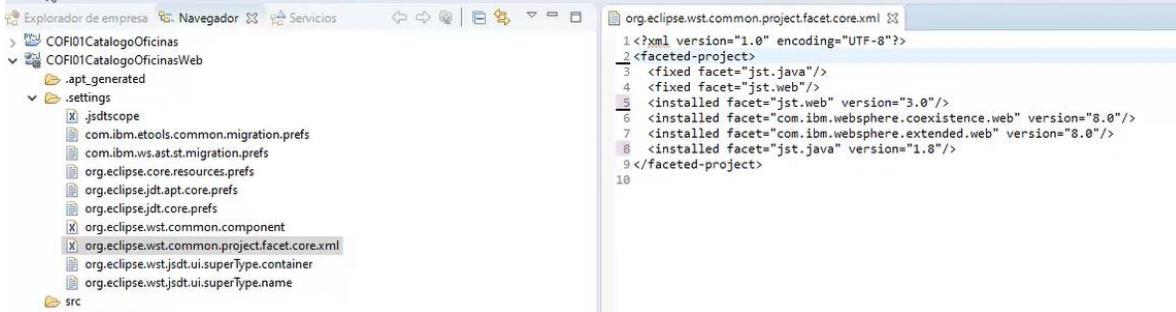
- Habilitamos la vista de navegador en el RSA



- Estando en el proyecto de negocio, desde la vista de Navegador, abrimos el archivo *.facet.core.xml en la carpeta .settings y lo dejamos como en la imagen

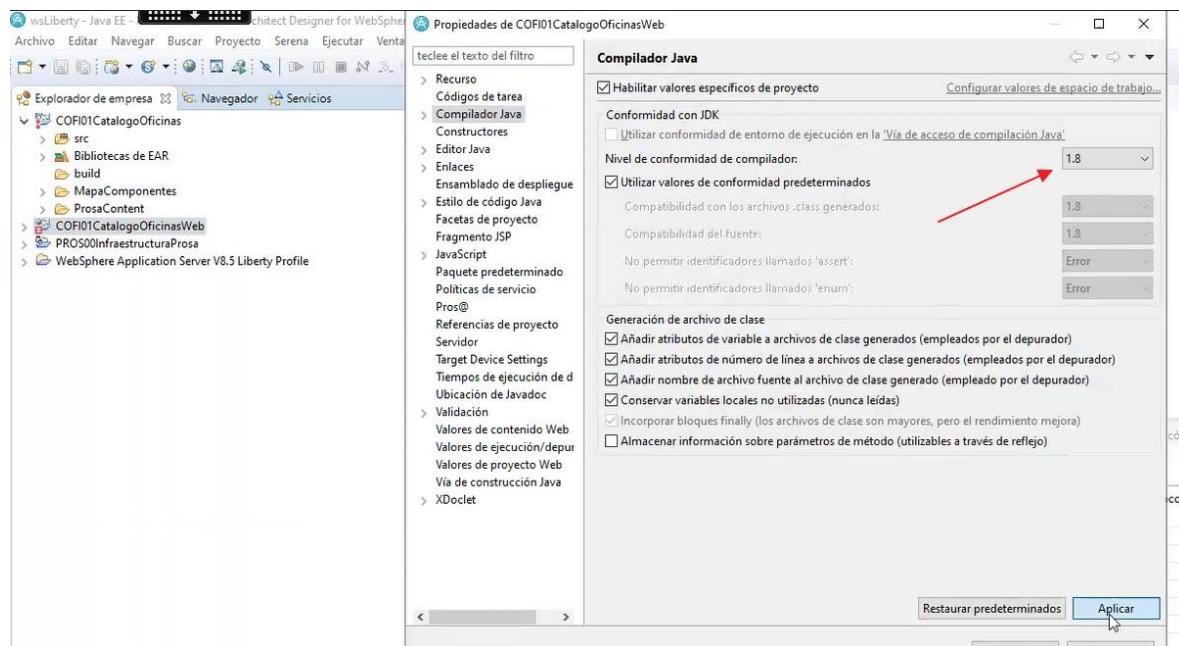


- Puede que haya que cambiar la versión de java a la 1.8
 - Cambiar el runtime de “Websphere Application Server v8.0” por el de Liberty: “WebSphere Application Server V8.5 Liberty Profile”
 - Eliminar las facetas de prosa.facet.ext si no se utilizan
- Igualmente para el proyecto web. Dejarlo como en la imagen:



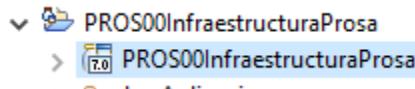
- Puede que haya que cambiar la versión de la faceta jst.web a la 3.0

Tras estos cambios, verificamos que está seleccionada la versión 1.8 en el Compilador Java de ambos proyectos:

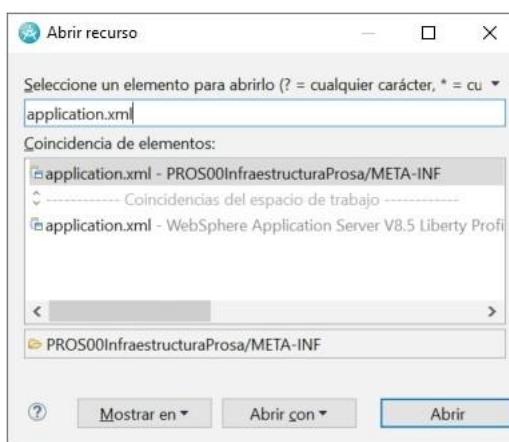


1.4.9.2 Application.xml.

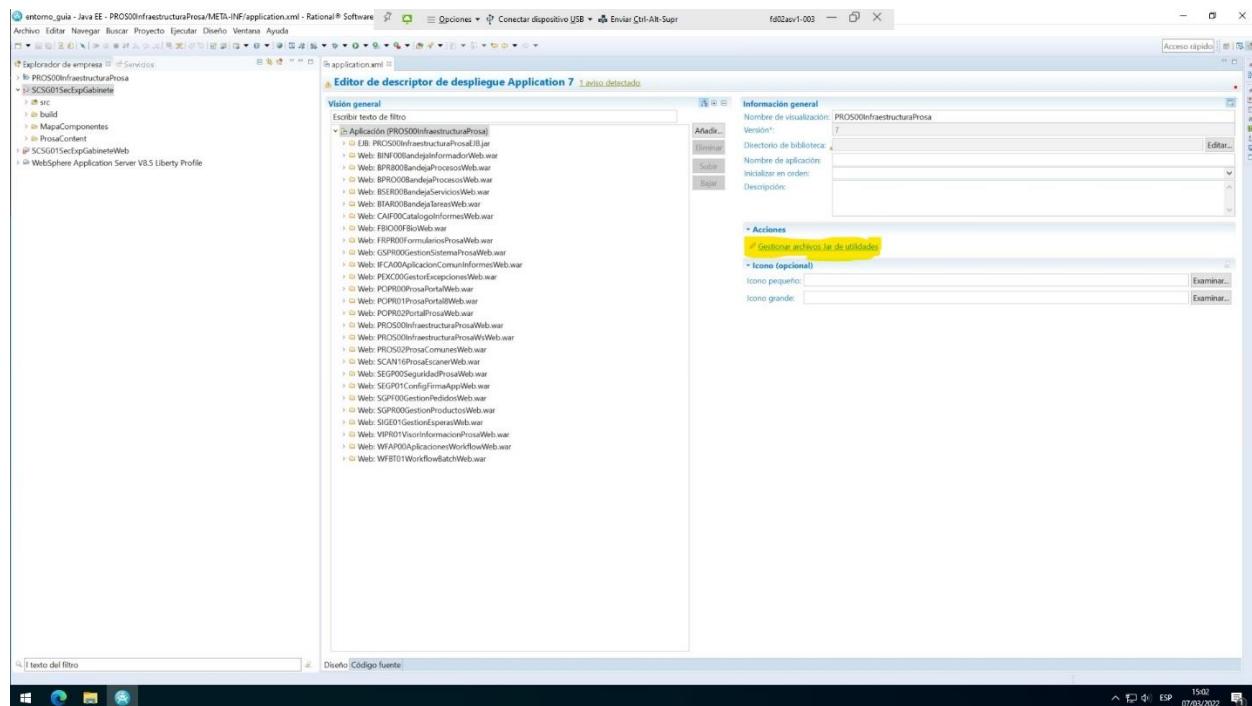
Con toda seguridad nos va dar errores de compilación en la aplicación, para poder solucionarlos primero tenemos que hacer que las aplicaciones vean el EAR de arquitectura. Para ello, hacemos doble click sobre el application.xml situado aquí:



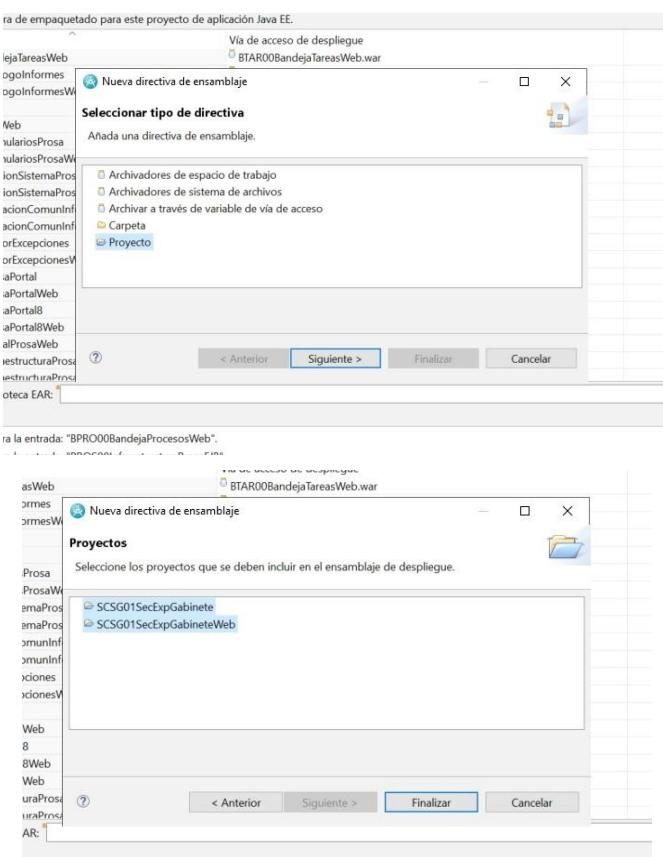
o presionamos Ctrl + Shift + R y buscamos application.xml.



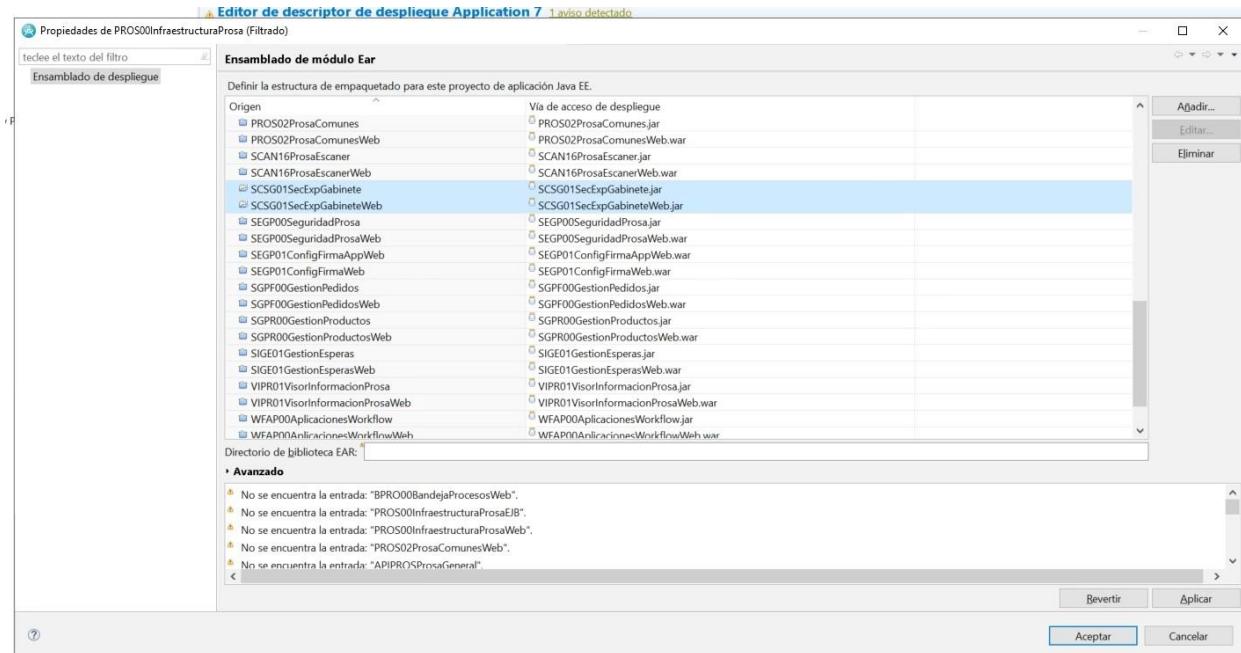
En el application.xml nos vamos a “Gestionar archivos Jar de utilidades” teniendo seleccionado el nodo raíz:



Una vez en este menú pinchamos en Añadir > Proyecto y seleccionamos nuestro proyecto



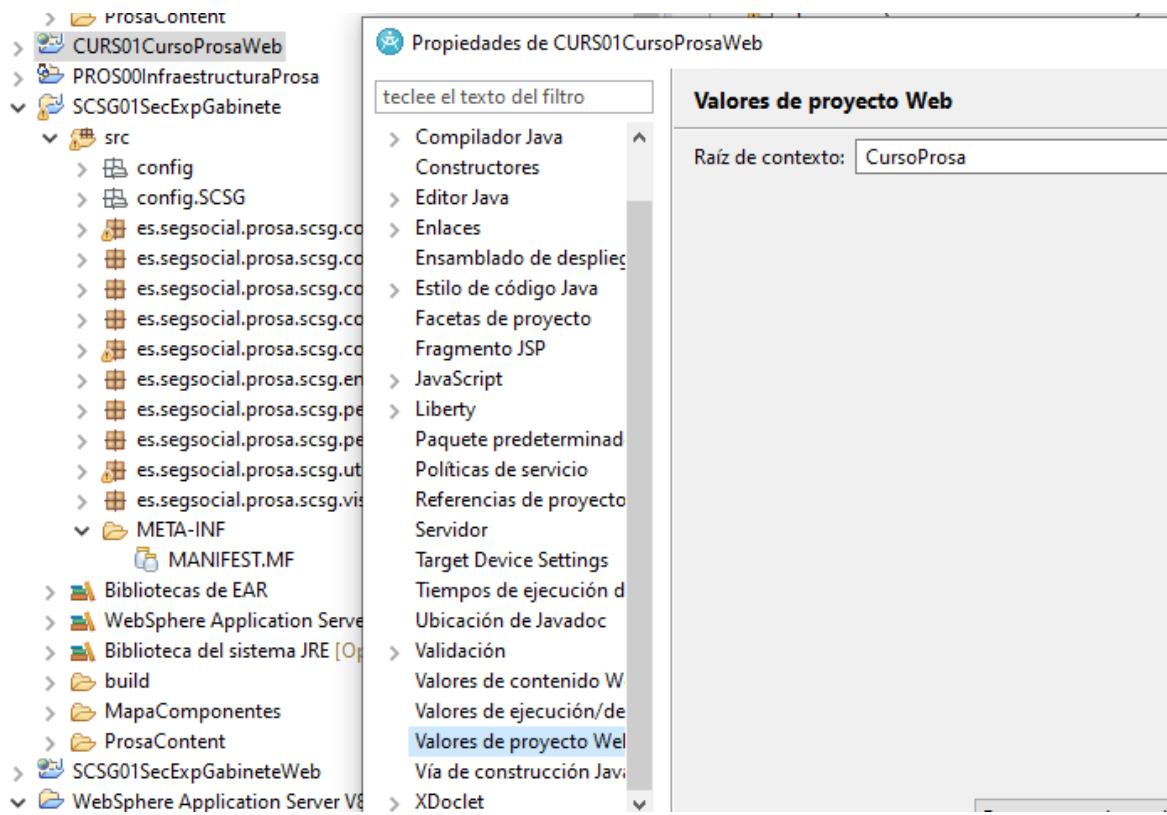
El resultado final será algo tal que así



NOTA: Verificar antes de añadir el proyecto que la caja de texto de “Directorio de biblioteca EAR” está vacía. Si pone “lib”, lo quitamos.

Hecho esto, tenemos que revisar en el fichero application.xml la configuración de nuestro proyecto para ver que está todo correcto. Comprobaremos que el RSA ha incluido el módulo de la aplicación y que los valores de web-uri y context-root están correctamente establecidos.

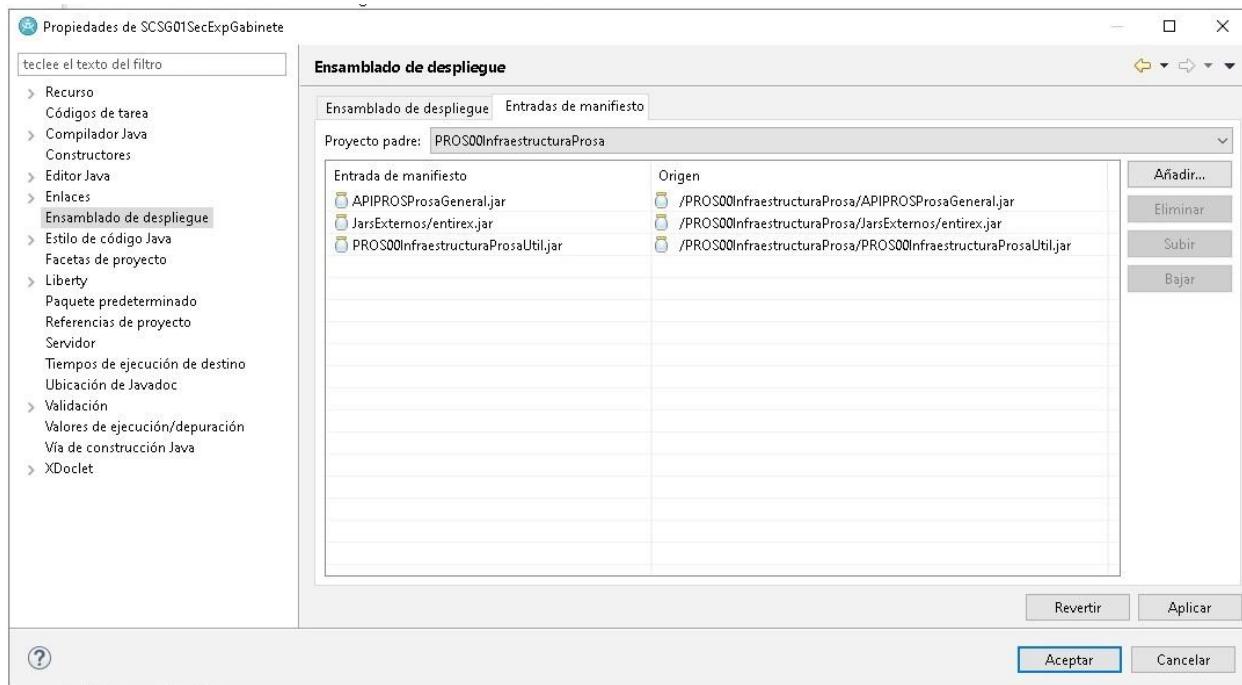
Para comprobar el Context root de la aplicación web:



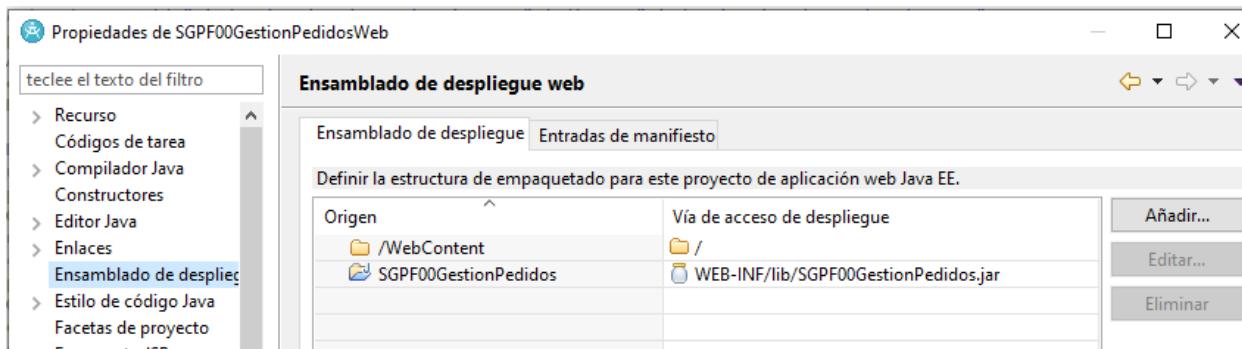
1.4.9.3 Ensamblado de despliegue y Manifest

Ahora ya podemos solventar los errores de compilación.

Clicamos con el botón secundario sobre el proyecto Java de la aplicación y vamos a Propiedades. En el nuevo menú vamos a Ensamblado de despliegue > Entradas de manifiesto. En este punto tenemos que redefinir las librerías dependientes de nuestro proyecto con la infraestructura.

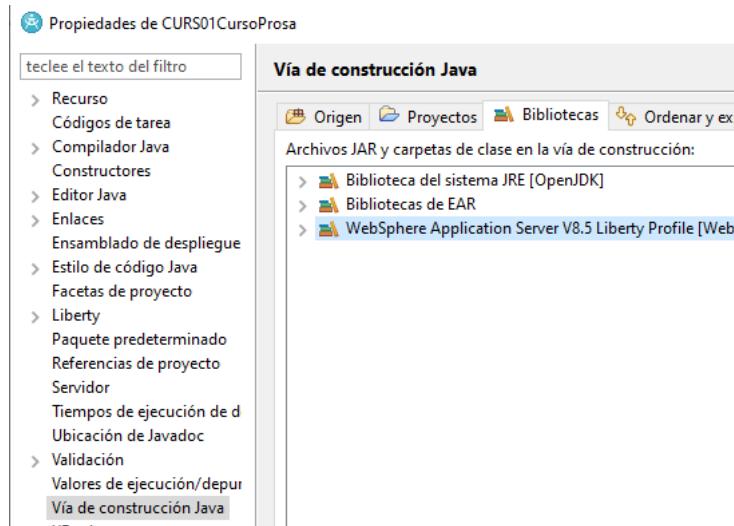


Si en nuestro proyecto, vamos a publicar **web services**, necesitaremos incluir proyecto java en el ensamblado de despliegue del proyecto web para conseguir que se publique correctamente el WS:

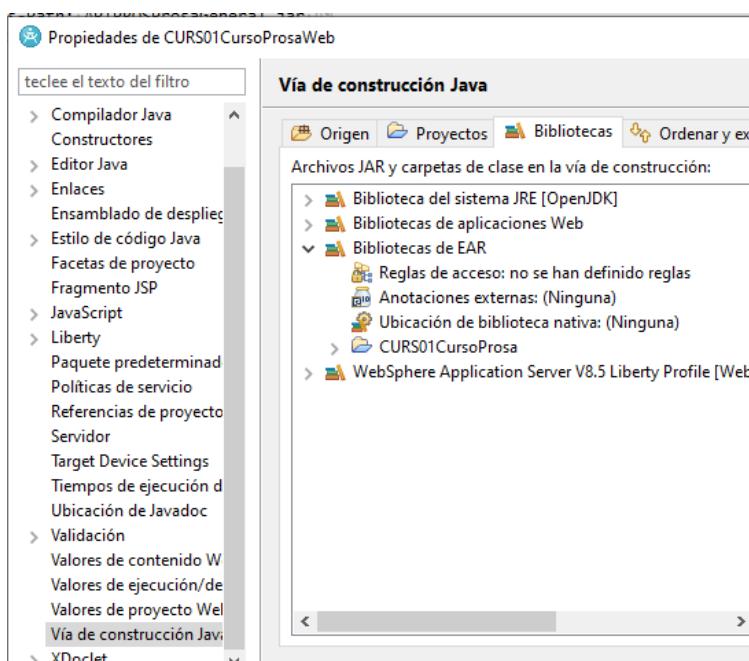


1.4.9.4 Bibliotecas de la vía de construcción Java

Revisamos las bibliotecas del proyecto de negocio. Para ello, vamos a las propiedades del proyecto, en Vías de construcción Java, pestaña Bibliotecas, y debería quedar como en la imagen:



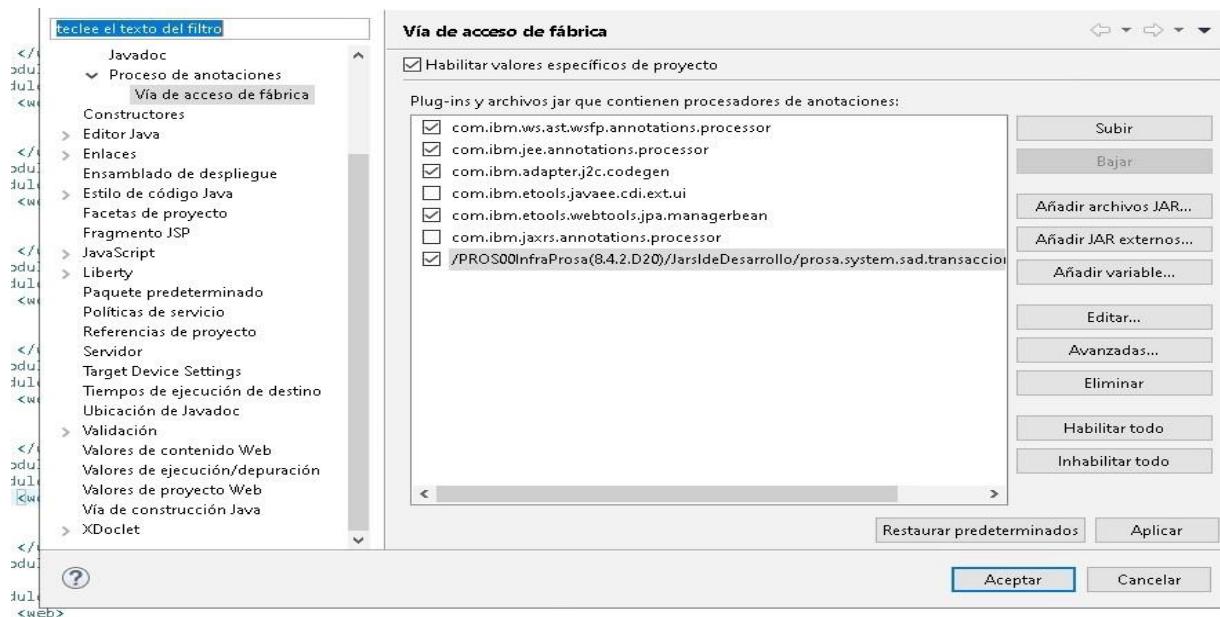
Y en el proyecto web:



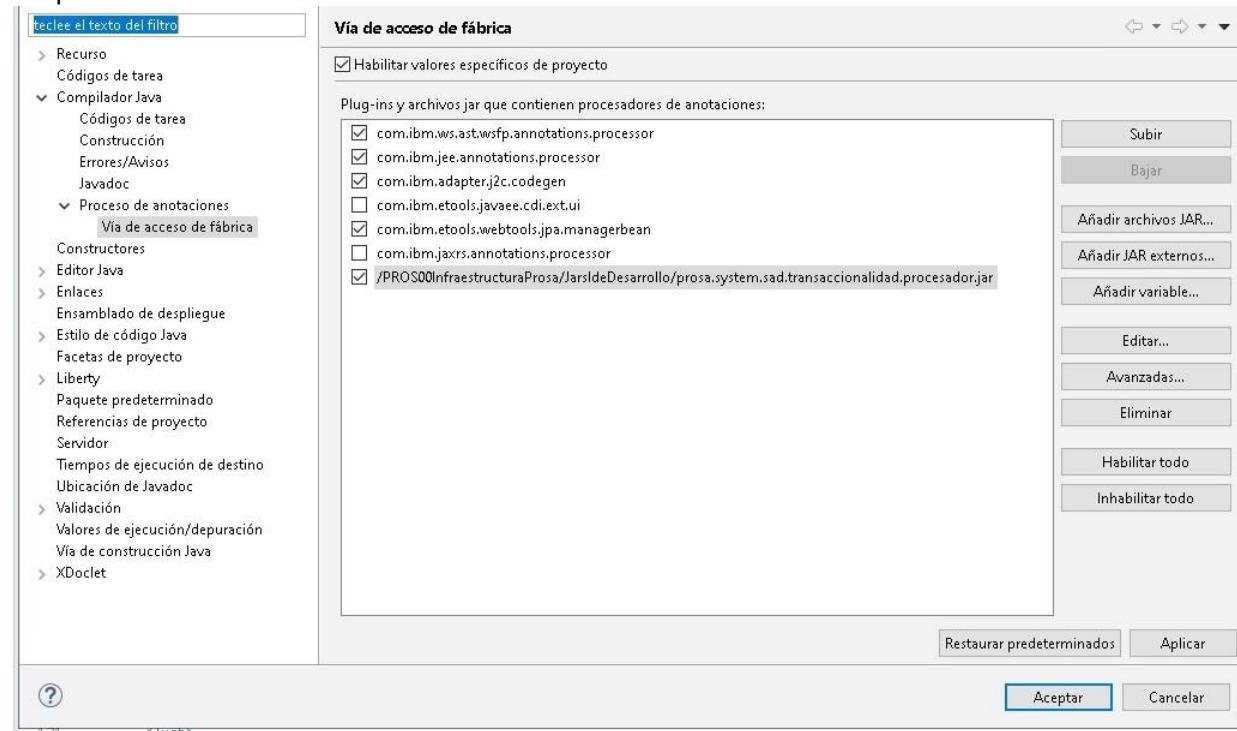
1.4.9.5 Procesador de anotaciones

Puede ocurrir que también haya un error de APT. Para solventarlo debemos hacer lo siguiente: botón secundario sobre el proyecto > Propiedades > Compilador Java > Procesadores de anotaciones > Vía de acceso de fábrica y modificamos la última entrada para dejar como las capturas:

Antes



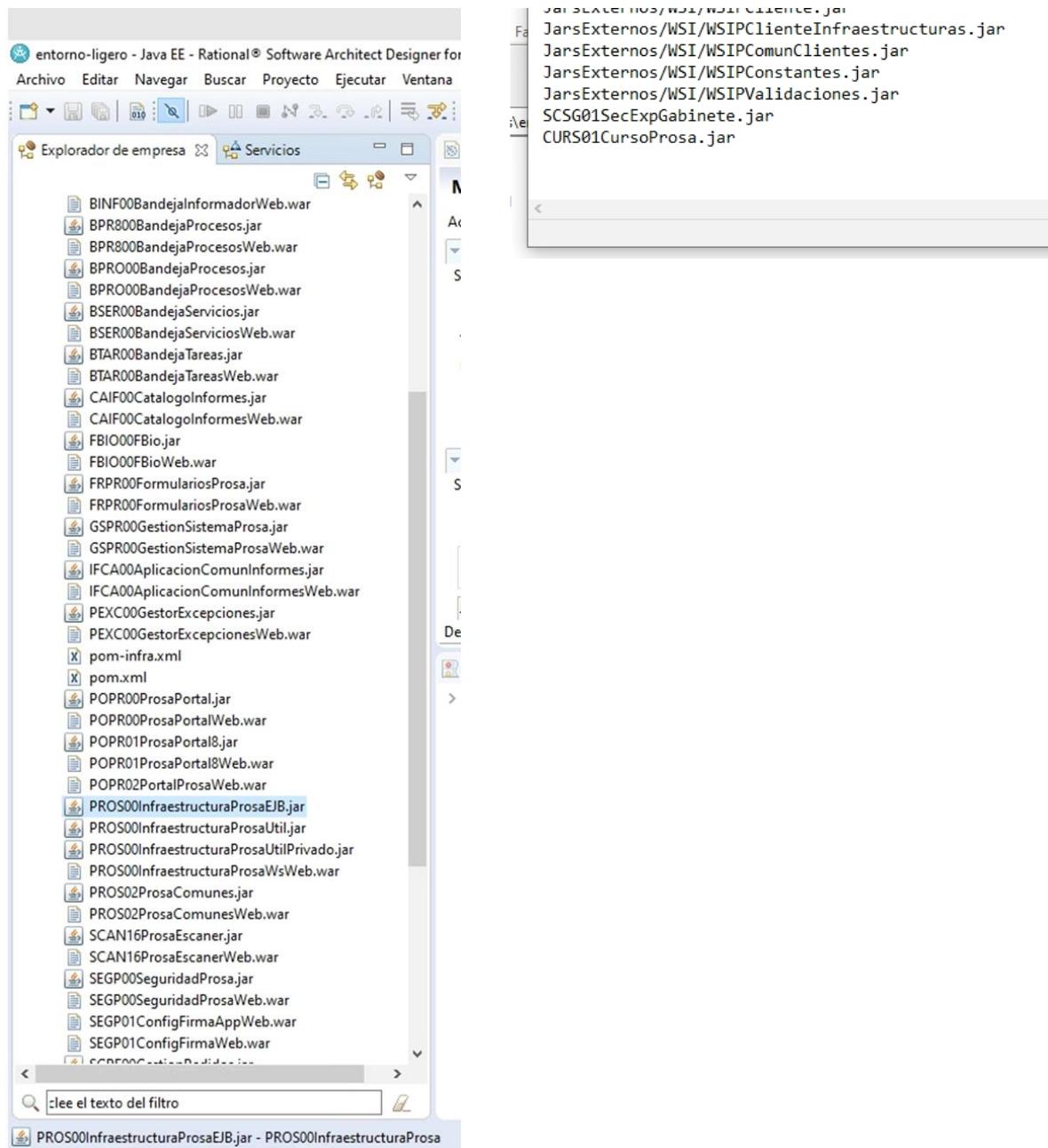
Después:



En el proyecto Web tenemos que hacer exactamente lo mismo.

1.4.9.6 MANIFEST de InfraestructuraProsaEJB.

También necesitaremos modificar el manifiesto del jar del EJB de Prosa Infraestructura y añadir el jar de la aplicación que estemos desarrollando, de lo contrario, nos darán errores a la hora de lanzar el servicio de la aplicación, ya que la infraestructura no verá la aplicación.



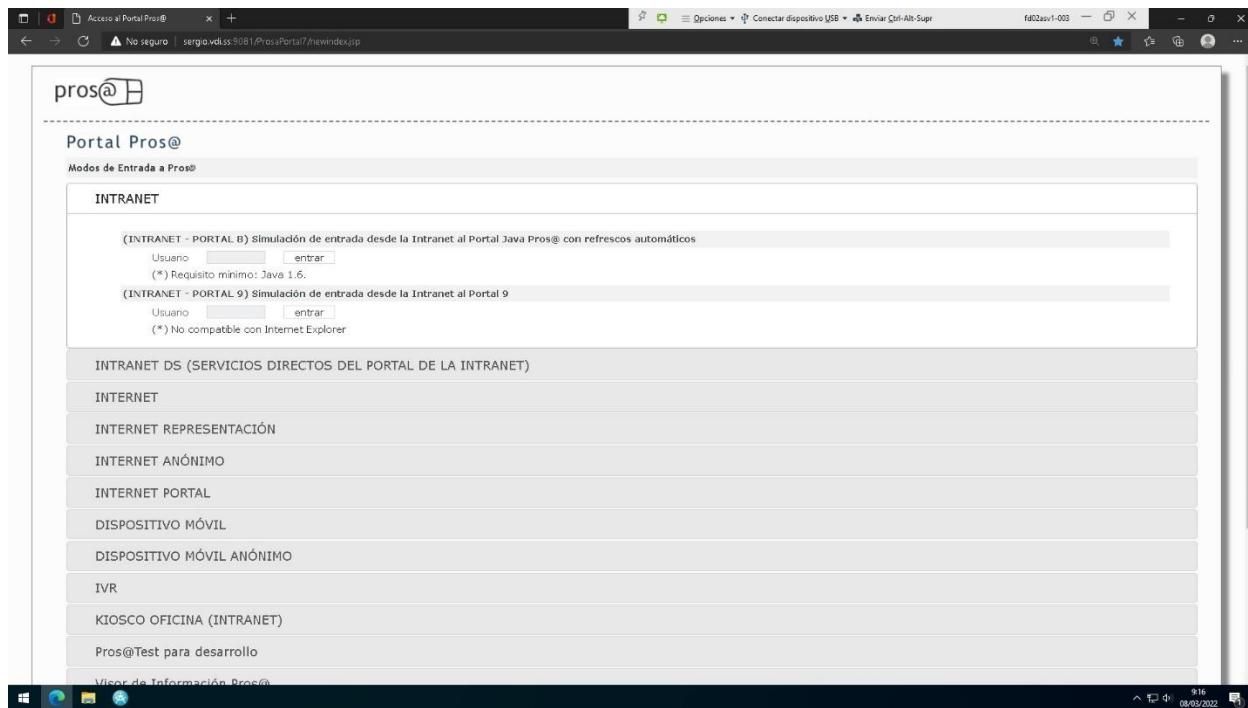
Para ello tendremos que abrir el jar de EJB como un zip, navegar al archivo manifest y añadir la aplicación al final. Tened en cuenta que todas las líneas deben tener el mismo formato (Espacios al principio y al final de cada línea) y guardar los cambios en el jar.

1.4.9.7 Arranque del portal.

Construimos y republicamos el proyecto, arrancamos el servidor y deberíamos entrar al portal sin problemas.

Con la siguiente url:

http://<NOMBRE_LOCALHOST>:<PUERTO_LOCALHOST>/ProsaPortal7/newindex.jsp



2. APÉNDICES

2.1 Breve descripción del archivo server.xml

En el archivo server.xml es donde van colocadas todas las variables necesarias para configurar nuestro servidor Liberty, y también donde colocaremos todos los valores dependientes del entorno que necesite nuestra aplicación. Tiene varios apartados, pero los que más nos importan a nosotros son los siguientes:

1. Entradas de tipo **jndiEntry**: en este tipo de entradas definiremos valores “timeout”, rutas del sistema de ficheros, nombres de usuarios, contraseñas, etc...
2. Entradas de tipo **jndiURLEntry**: aquí definiremos url para llamar a servicios externos, ya sean Restful o SOAP

Estos dos tipos de entradas tienen una estructura similar, sólo cambia el nombre de la etiqueta xml

```
<jndiEntry id="identificador_único" jndiName="nombre_por_el_que_buscaremos_el_valor"  
value="valor_de_la_entrada">
```

3. Entradas de tipo **dataSource**: aquí es donde definiremos las bases de datos que usará nuestra aplicación.

```
<dataSource id="identificador_único" jndiName="nombre_por_el_que_buscaremos_el_valor">  
    <jdbcDriver libraryRef="referencia_a_los_drivers_de_bbdd_configurados">  
        <properties.oracle URL="cadena_deConexion" password="contraseña" user="usuario">  
</dataSource>
```

Las passwords deben estar codificadas con el **algoritmo de cifrado de Websphere XOR**, disponible en esta url:

<https://strelitzia.net/wasXORdecoder/wasXORdecoder.html>

NOTA: En el server_local.xml está definida esta variable JNDI:

```
<jndiEntry id="log_fichero_directorio" jndiName="prosa/config/log/fichero/directorio"  
value="C:/temp/Prosa"/>
```

La ruta indicada en el value, debe existir previamente. Si no es así, crearla y reiniciar

2.2 Instalación de nuevas características (features) Liberty

2.2.1 Creación archivo repositories.properties

Ahora nos vamos a <Carpeta donde tengamos Liberty>\etc donde hay un archivo llamado repositories.properties_bkp, borramos el bkp de la extensión del archivo para dejarlo como repositories.properties

2.2.2 Ejecución comando instalación

Para ejecutar el comando de instalación debemos abrir un CMD o PowerShell y ejecutamos el siguiente comando <Carpeta donde tengamos Liberty>\bin\installUtility install feature-1 feature-2 ...

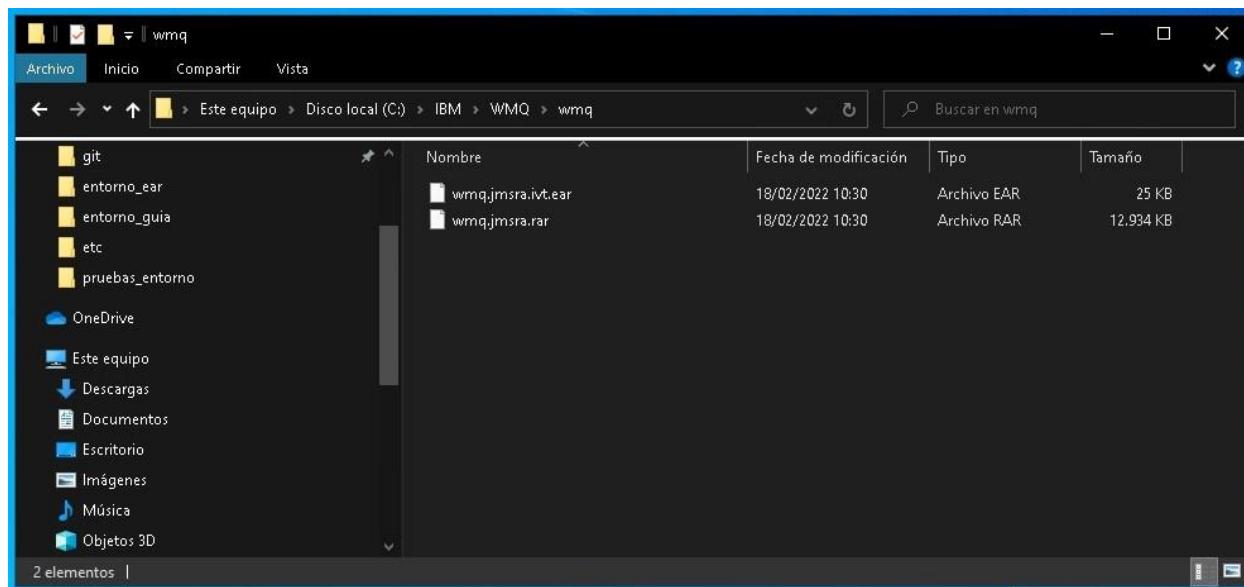
2.3 Instalación bibliotecas IBM MQ

El servidor base que te has descargado no tiene instalada la funcionalidad de IBM MQ.

En caso de necesitar trabajar con colas locales, primero deberías realizar el punto anterior para instalar esa funcionalidad (comando installUtility install wmqJmsClient-2.0).

Además de esto, te debes descargar la librería MQ del repositorio, carpeta Software/Libreria_MQ y descomprimirla.

Creamos la siguiente carpeta C:\IBM\WMQ y ejecutamos el jar anterior con un java -jar 9.1.5.0-IBM-MQ-Java-InstallRA.jar (seguimos todos los pasos que nos vaya indicando). Una vez ejecutado el comando comprobamos que se ha creado una carpeta en la ruta anterior con nombre wmq en minúscula y cuyo contenido es el siguiente:



Y añadir esta feature en el server.xml dentro de la sección <features>:

```
<feature>wmqJmsClient-2.0</feature>
```

Y esta entrada en el server_local.xml con la ruta en la que se haya descargado el rar:

```
<variable name="wmqJmsClient.rar.location" value="C:/IBM/WMQ/wmq/wmq.jmsra.rar"/>
```

2.4 Instalación de la feature local-connector

En el caso de tener ya un entorno de Liberty instalado y necesitar instalar la feature de local-connector, veremos un error como el siguiente en la consola:

```
[ERROR] CWWKF0042E: No se puede encontrar una definición de característica para la característica localconnector-1.0.
```

En este caso se puede instalar o bien por comando:

```
<Carpeta donde tengamos Liberty>/bin/installUtility install localconnector-1.0
```

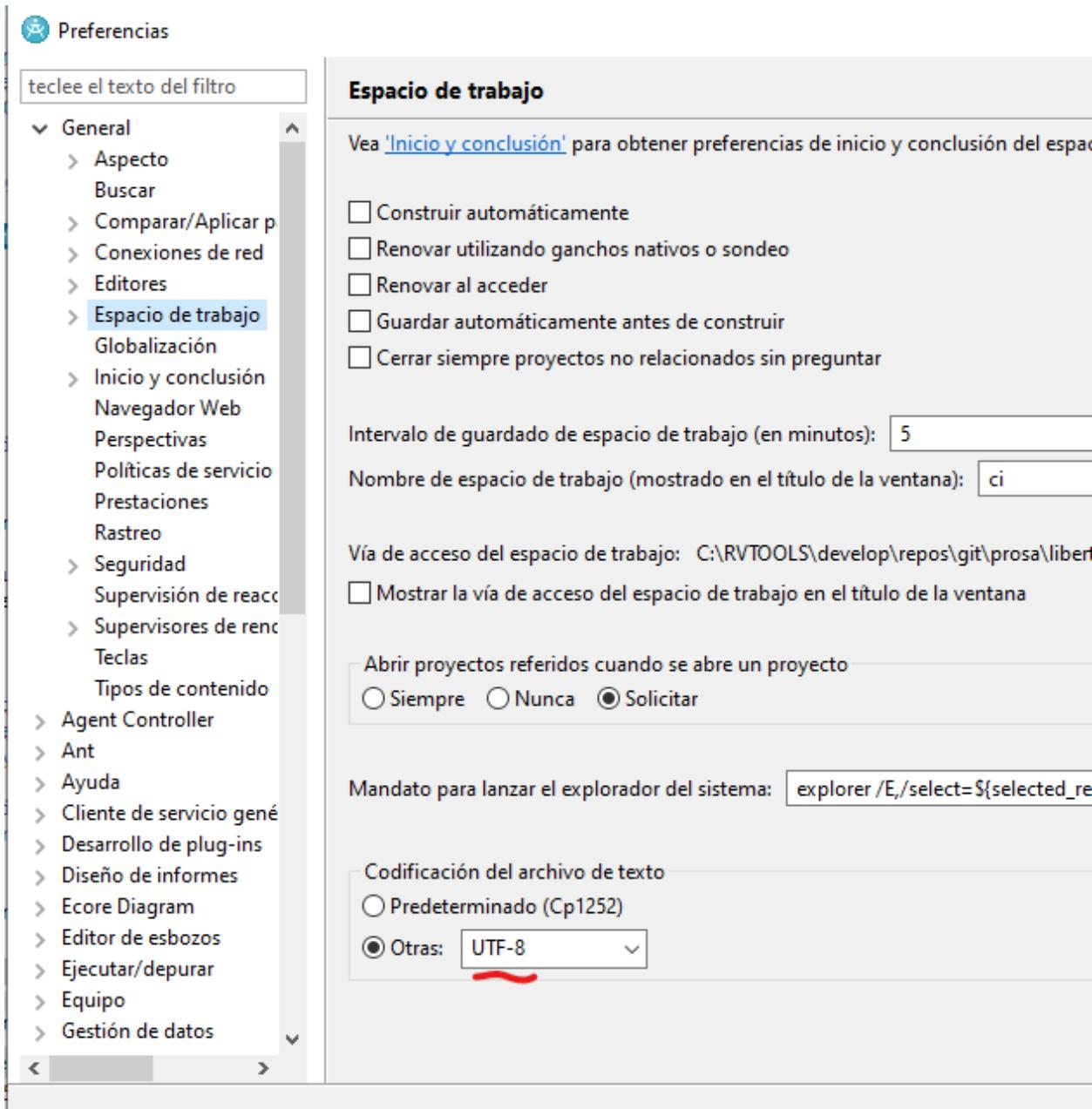
o bien con los siguientes pasos:

- Descargar el zip **localConnector-1.0.zip** de en el repositorio, carpeta Software/Liberty
- Para el servidor Liberty en caso de estar arrancado
- Descomprimir el zip en el directorio “lib” en la ruta donde se haya instalado el Liberty
- Descomprimirá los siguientes archivos:
 - o lib/com.ibm.ws.jmx.connector.local_1.0.59.jar
 - o lib/features/com.ibm.websphere.appserver.localConnector-1.0.mf
- Eliminamos el zip y rearrancamos el Liberty

- Ahora deberíamos ver una traza como la siguiente con una lista de features instaladas y entre ellas debería aparecer la de localConnector:
- [AUDIT] CWWKF0012I: El servidor ha instalado las características siguientes: [..., localConnector-1.0,...]

2.5 Encoding UTF-8 en el Workspace

Podemos verificar o configurar UTF-8 como encoding en Ventana – Preferencias – General – Espacio de trabajo:

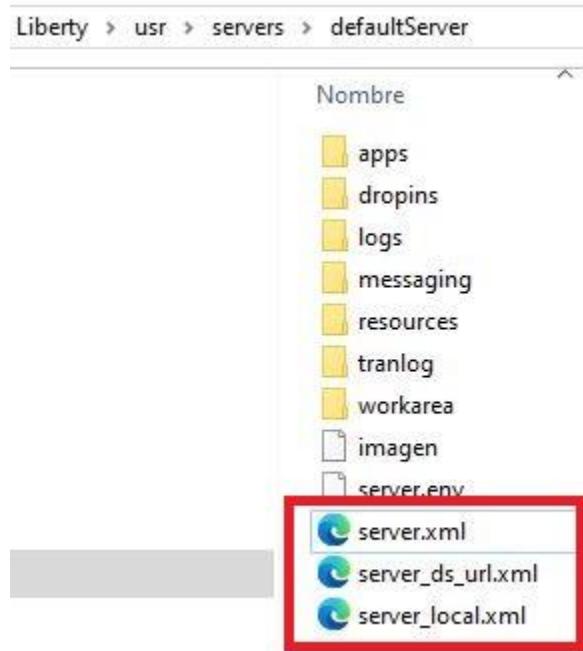


2.6 Cómo usar un Almacén de Certificados concreto (key.p12)

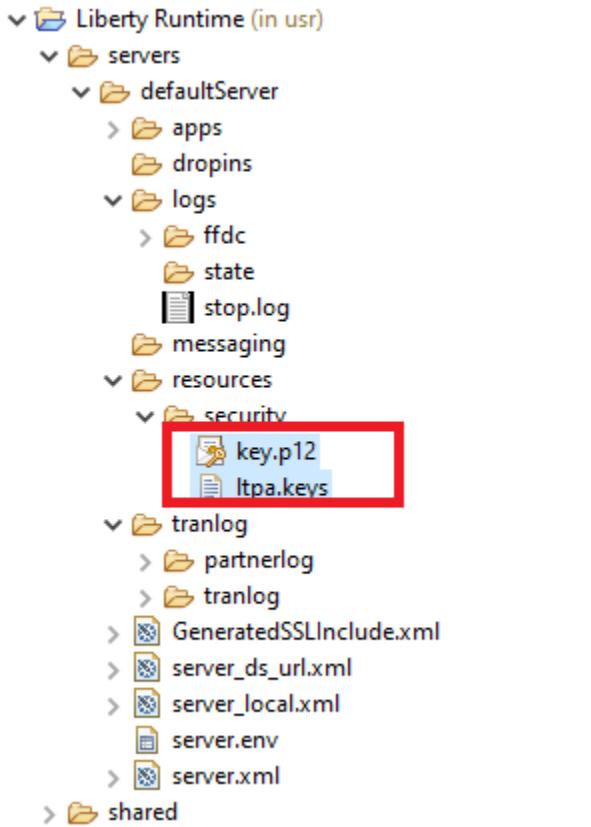
Definimos en el archivo “hosts” el nombre del servidor que corresponda al utilizado al generar el almacén de certificados, en este caso “prosadesa.portal.ss”. Debe aparecer como primera línea cuya IP sea distinta a 127.0.0.1, con el valor real exacto de IP de la máquina en cuestión:

```
# localhost name resolution is handled within DNS itself.  
127.0.0.1      localhost  
10.198.XXX.XXX  prosadesa.portal.ss  
#10.198.XXX.XXX  mipc.portal.ss
```

Descargamos de Nexus los archivos de configuración del servidor (server.xml, server_ds_url.xml y server_local.xml) y los copiamos en la ruta apropiada de instalación de Liberty:



Con el servidor detenido elimino los archivos:



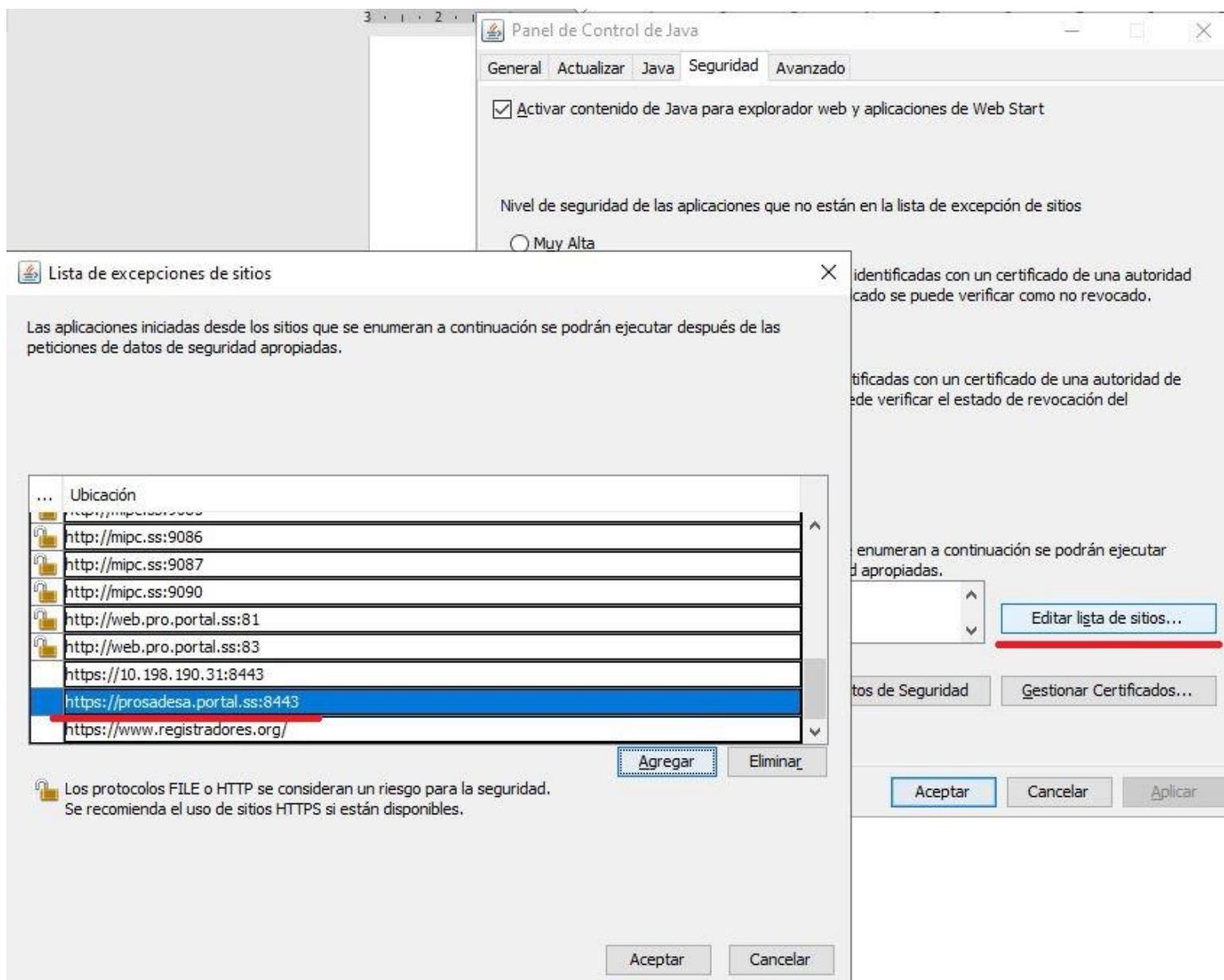
Y colocar en su lugar el archivo key.p12 que se encuentra en Nexus, en la localización:

The screenshot shows the 'Nexus Repository Manager' interface with the URL 'entorno_ligero_liberty'. The 'HTML View' section displays a file tree under 'ConfiguracionServidor'. The 'Certificados' folder contains several certificate files, including 'key.p12', which is highlighted with a red box.

En los archivos host (fuera del EV) tendremos que poner:

10.198.XXX.XXX prosadesa.portal.ss

Ejecutando, como administrador, el panel de control de java, agregaremos el sitio (con https y el puerto seguro que tengamos en el server_local.xml):



Ejecutamos Prosa con https (<https://prosadesa.portal.ss:8443/ProsaPortal7/newindex.jsp>).
En el caso de Firefox, avisará de que el certificado es autofirmado, y podremos verlo:

Certificado



The screenshot shows a certificate dialog box with two main sections: "Nombre del asunto" (Subject) and "Nombre del emisor" (Issuer). Both sections show the same information: País: us, Organización: ibm, Unidad organizativa: defaultServer, and Nombre común: prosadesa.portal.ss. The "Nombre común" field is highlighted with a green border.

Nombre del asunto	
País	us
Organización	ibm
Unidad organizativa	defaultServer
Nombre común	prosadesa.portal.ss

Nombre del emisor	
País	us
Organización	ibm
Unidad organizativa	defaultServer
Nombre común	prosadesa.portal.ss

Aceptamos el riesgo y continuamos.