

A decorative graphic on the left side of the cover. It features a cluster of white-outlined numbers (0-9) and mathematical symbols (+, -, *, /) at the top. Below this, a series of 3D cubes are arranged in a stepped, staircase-like pattern, receding into the distance. The cubes are rendered in various shades of blue and grey, with some faces highlighted in a lighter blue. The background is a solid dark blue.

MANUAL TÉCNICO

Invocaciones EJB de un cliente remoto utilizando JNDI

MAYO 2018



Invocaciones EJB de un cliente remoto utilizando JNDI

Dirección responsable:

Dirección de Registros Administrativos

Realizado por:

Víctor Espinoza Ochoa

Analista de Registros Administrativos

Aprobado por:

Jose Villota

Jefe de Gestión de Innovación para Automatizar la Producción Estadística

Tabla de contenido

A. INTRODUCCIÓN	7
B. ESPECIFICACIONES TÉCNICAS.....	8
C. CONFIGURACIONES	9
1. SERVIDOR DE APLICACIONES REMOTAS (CONTENEDOR EJB)	9
1.1. Creación de usuario de Aplicación	9
1.2. Habilitar Wildfly para aceptar conexiones remotas	12
2. SERVIDOR DE APLICACIONES CLIENTE	14
2.1. Dominio de Seguridad.....	14
2.2. Subsistemas remotos.....	14
2.2.1. Grupo de sockets standard.....	15
2.2.2. Conector Remoto.....	16
2.2.3. Socket de salida remota	18
2.2.4. Conexiones de salida remota	19
2.3. Configuración de parámetros	21
3. CREACIÓN EJB REMOTO	23
3.1. Creación de interface.....	24
3.2. Implementación del servicio	25
4. CLIENTE - WEB APPLICATION	28
4.1. Librería Jboss-client.jar	28
4.2. Librería EJB remoto.....	28
4.3. Referencia a las conexiones de salida.....	28
4.4. Referencias EJB externas	29
4.5. Contexto JNDI	29
4.6. Lookup	30
4.7. Declaración del Servicio en el controlador.....	31
5. DESPLIEGUE DE EJB'S – JAR'S.....	31
5.1. Implementación en el Servidor de Aplicaciones.....	31
5.2. Implementación en el Servidor Web	32
5.3. Arquitectura lógica proyectos.....	33

Índice de Figuras

Figura 1. Archivo add-user.bat.....	9
Figura 2: Usuario de Aplicación	9
Figura 3: Nombre de usuario	10
Figura 4: Contraseña de usuario.....	10
Figura 5: Proceso de conexión remota al usuario.	11
Figura 6: Valor Secreto	11
Figura 7. Interfaces de conexión	12
Figura 8: Reemplazo de IP	13
Figura 9: Reinicio de Wildfly.....	13
Figura 10: Grupo de sockets standard	15
Figura 11: Agregar socket.....	15
Figura 12: Socket agregado.....	16
Figura 13: Configurar Conector Remoto	16
Figura 14: Añadir Conector Remoto.....	17
Figura 15: Atributos de Conector Remoto	17
Figura 16: Sockets de salida remota.....	18
Figura 17: Añadir Socket de salida remota	18
Figura 18: Conexiones de Salida Remota.....	19
Figura 19: Agregar Conexiones de Salida Remota	19
Figura 20: Atributos de Conexión de Salida Remota.....	20
Figura 21: Propiedad "SASL_POLICY_NOANONYMOUS"	20
Figura 22: Propiedad "SSL_ENABLED"	21
Figura 23: Archivo parametros.properties	21
Figura 24: Configuración parametros.properties	22
Figura 25: Creación EJB Module	24
Figura 26: Paquete de Interfaces.....	24
Figura 27: interface del Servicio.....	25
Figura 28. Creación Session Bean.....	25
Figura 29: Servicio creado	26
Figura 30: Interface Local	26
Figura 31: Interfaz Remota	27
Figura 32: Implementación del servicio.....	27
Figura 33: Referencia a las conexiones de salida	28
Figura 34: Referencia EJB externa - web.xml	29
Figura 35: Contexto y Paramtros JNDI	30
Figura 36: Lookup.....	30
Figura 37: Declaración del Servicio.....	31
Figura 38: Esquema lógico Contenedor WEB.....	33
Figura 39: Esquema lógico Servidor Aplicación	33

Índice de Tablas

TABLA 1: REQUERIMIENTOS MÍNIMOS DE HARDWARE 8

TABLA 2: REQUERIMIENTOS MÍNIMOS DE SOFTWARE 8

A. INTRODUCCIÓN

JNDI (Java Naming and Directory Interface) es una API especificada en la tecnología Java que proporciona funcionalidades de nombrado y directorio para aplicaciones escritas en el lenguaje de programación Java. Está diseñado especialmente para la plataforma Java utilizando el modelo de objetos de Java. Usando JNDI, las aplicaciones basadas en tecnología Java pueden almacenar y recuperar objetos Java nombrados de cualquier tipo. Además, JNDI proporciona métodos para realizar operaciones de directorio estándar, como asociar atributos con objetos y buscar objetos utilizando sus atributos.

JNDI también se define independientemente de cualquier implementación específica de nombres o servicios de directorio. Permite a las aplicaciones acceder a diferentes, o múltiples servicios de directorio y nombres diferentes utilizando una API común, por ejemplo es utilizado por EJB para encontrar objetos remotos. Diferentes proveedores de nombres y servicios de directorio pueden conectarse sin problemas detrás de esta API común. Esto permite que las aplicaciones basadas en la tecnología Java aprovechen la información en una variedad de servicios de nombres y directorios existentes, tales como LDAP, NDS, DNS, NIS (YP), CORBA y RMI, además de permitir que las aplicaciones coexistan con el software y los sistemas heredados.

Utilizando JNDI como herramienta, puede crear nuevas aplicaciones potentes y portátiles que no solo aprovechen el modelo de objetos de Java, sino que también estén bien integradas con el entorno en el que se implementan.

En el contexto de EJB, hay dos términos:

- Binding: se refiere a la asignación de un nombre a un objeto EJB.
- Lookup: esto se refiere a buscar y obtener un objeto de EJB.

La API JNDI permite escribir código Java que realice operaciones sobre directorios. Es una API uniforme a todos los tipos de servicios de directorios. Es similar a JDBC.

B. ESPECIFICACIONES TÉCNICAS

TABLA 1: REQUERIMIENTOS MÍNIMOS DE HARDWARE

CARACTERÍSTICAS	
Procesador Mínimo:	Intel (R) Core(TM)2 Quad CPU 2.66GHz 2.67GHz
Memoria RAM Mínimo:	4 GB
Disco Duro:	100 GB

TABLA 2: REQUERIMIENTOS MÍNIMOS DE SOFTWARE

CARACTERÍSTICAS	
IDE de desarrollo	NetBeans 8.2 o Superior
Servidor de Aplicaciones	Wildfly-10.1 o superior
Lenguaje de Programación	Java SE Development Kit 8 o superior.

C. CONFIGURACIONES

1. Servidor de aplicaciones remotas (contenedor EJB)

1.1. Creación de usuario de Aplicación

- En primer lugar, se crea un usuario de aplicación en el Servidor (Wildfly-10.1 o superior), para ello se ejecuta el archivo add-user.bat localizado en \${WildFly-10.1_HOME}\bin\add-user.bat, Figura 1.

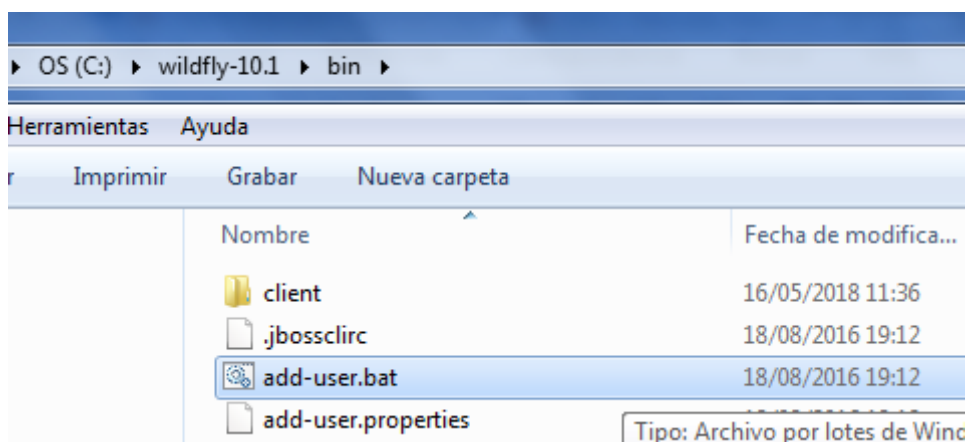


Figura 1. Archivo add-user.bat

- Se selecciona la opción **b)** para Usuario de Aplicación.

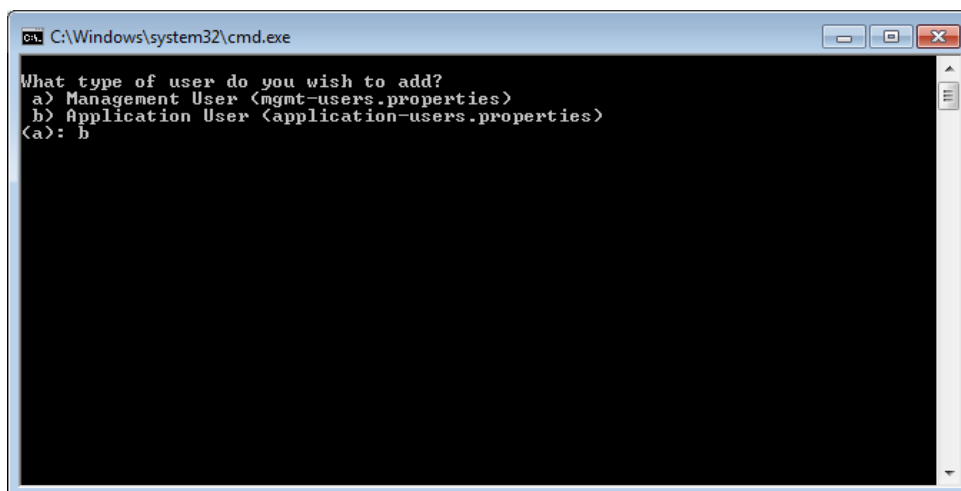


Figura 2: Usuario de Aplicación

- Se ingresa el nombre del usuario.

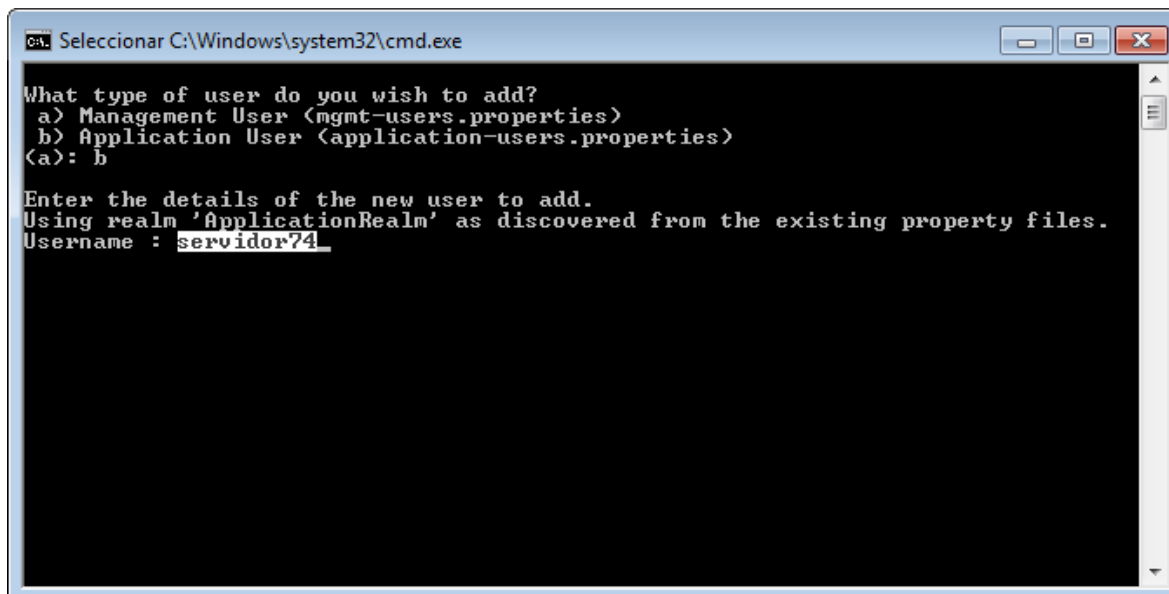


Figura 3: Nombre de usuario

- Ingrese la contraseña y re-confírmela.

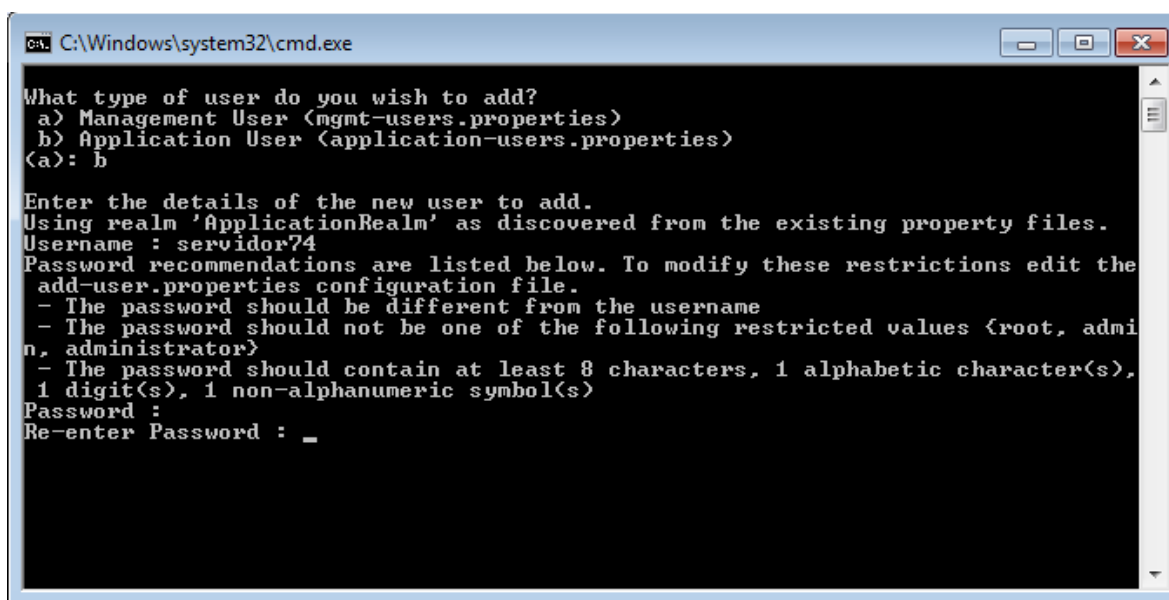
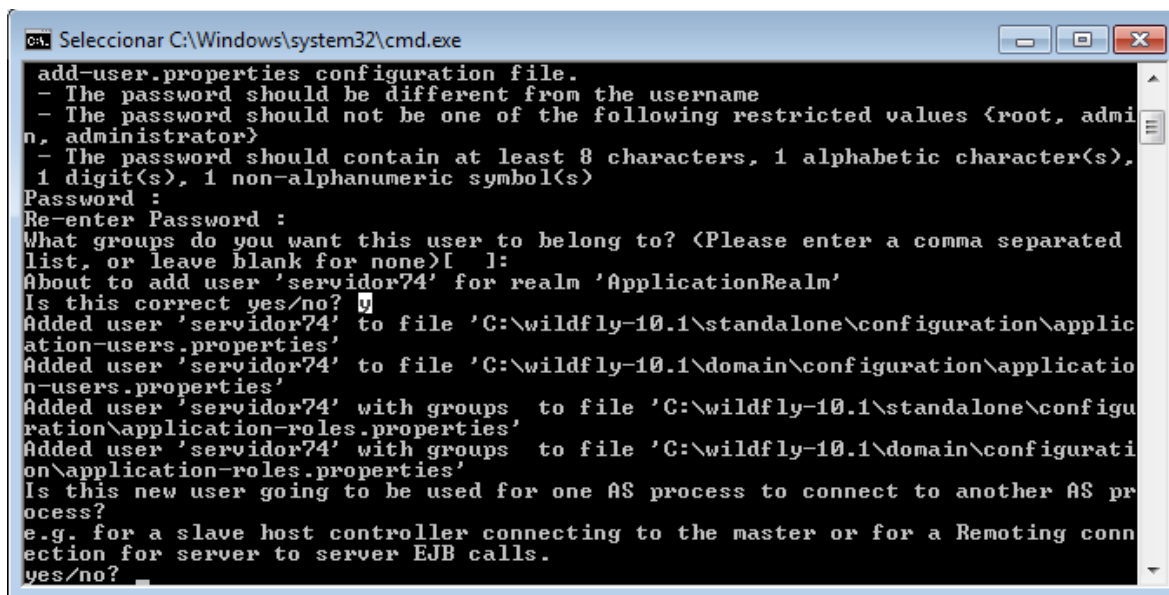


Figura 4: Contraseña de usuario

- Dejar en blanco a la pregunta "¿A qué grupos desea que pertenezca este usuario?" y responder **Sí** para agregar el usuario a "ApplicationRealm".



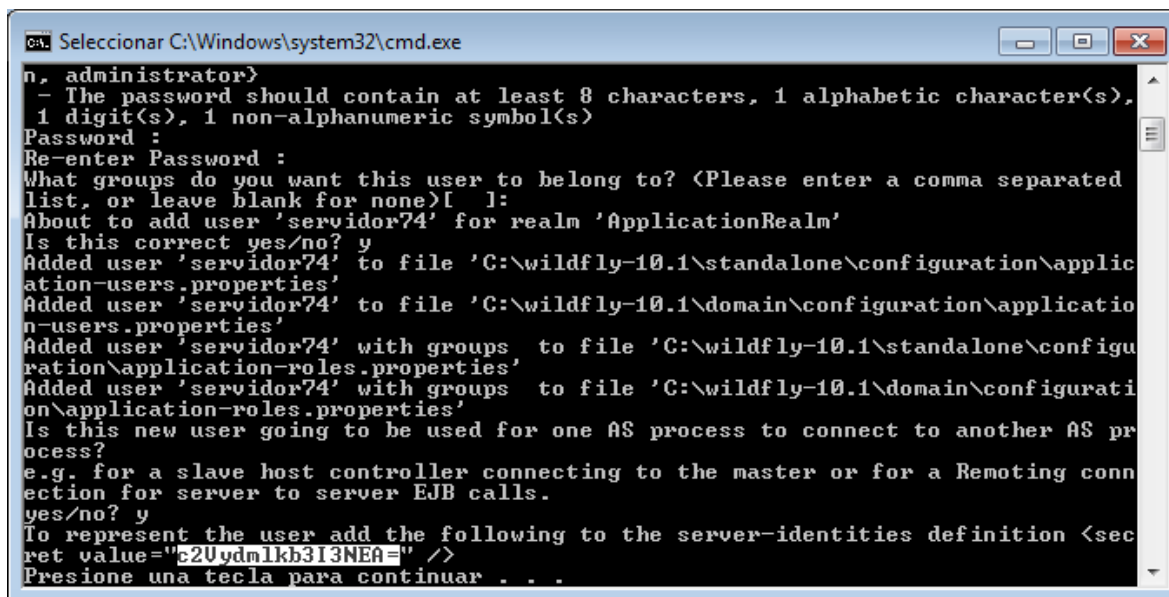
```

C:\Windows\system32\cmd.exe
add-user.properties configuration file.
- The password should be different from the username
- The password should not be one of the following restricted values {root, administrator}
- The password should contain at least 8 characters, 1 alphabetic character(s), 1 digit(s), 1 non-alphanumeric symbol(s)
Password :
Re-enter Password :
What groups do you want this user to belong to? (Please enter a comma separated list, or leave blank for none)[]
About to add user 'servidor74' for realm 'ApplicationRealm'
Is this correct yes/no? y
Added user 'servidor74' to file 'C:\wildfly-10.1\standalone\configuration\application-users.properties'
Added user 'servidor74' to file 'C:\wildfly-10.1\domain\configuration\application-users.properties'
Added user 'servidor74' with groups to file 'C:\wildfly-10.1\standalone\configuration\application-roles.properties'
Added user 'servidor74' with groups to file 'C:\wildfly-10.1\domain\configuration\application-roles.properties'
Is this new user going to be used for one AS process to connect to another AS process?
e.g. for a slave host controller connecting to the master or for a Remoting connection for server to server EJB calls.
yes/no?

```

Figura 5: Proceso de conexión remota al usuario.

- Responda **sí** a la pregunta "¿Este nuevo usuario va a ser usuario de un proceso AS para conectarse a otro proceso AS?".



```

C:\Windows\system32\cmd.exe
n, administrator}
- The password should contain at least 8 characters, 1 alphabetic character(s), 1 digit(s), 1 non-alphanumeric symbol(s)
Password :
Re-enter Password :
What groups do you want this user to belong to? (Please enter a comma separated list, or leave blank for none)[]
About to add user 'servidor74' for realm 'ApplicationRealm'
Is this correct yes/no? y
Added user 'servidor74' to file 'C:\wildfly-10.1\standalone\configuration\application-users.properties'
Added user 'servidor74' to file 'C:\wildfly-10.1\domain\configuration\application-users.properties'
Added user 'servidor74' with groups to file 'C:\wildfly-10.1\standalone\configuration\application-roles.properties'
Added user 'servidor74' with groups to file 'C:\wildfly-10.1\domain\configuration\application-roles.properties'
Is this new user going to be used for one AS process to connect to another AS process?
e.g. for a slave host controller connecting to the master or for a Remoting connection for server to server EJB calls.
yes/no? y
To represent the user add the following to the server-identities definition <secret value="s2Uydm1kb3I3NEA=" />
Presione una tecla para continuar . . .

```

Figura 6: Valor Secreto

- El **valor secreto** generado se lo utilizará para la configuración en el servidor del Cliente e indica la contraseña codificada en **Base 64**. Se puede recuperar el valor secreto de la contraseña en la siguiente página: <https://www.base64encode.org/>.

1.2. Habilitar Wildfly para aceptar conexiones remotas

- Para habilitar el acceso, se lo puede realizar desde la consola de administración del servidor de aplicaciones.
- Una vez que se haya ingresado a la consola, ir a **Configuration > Interfaces** (Figura 7).

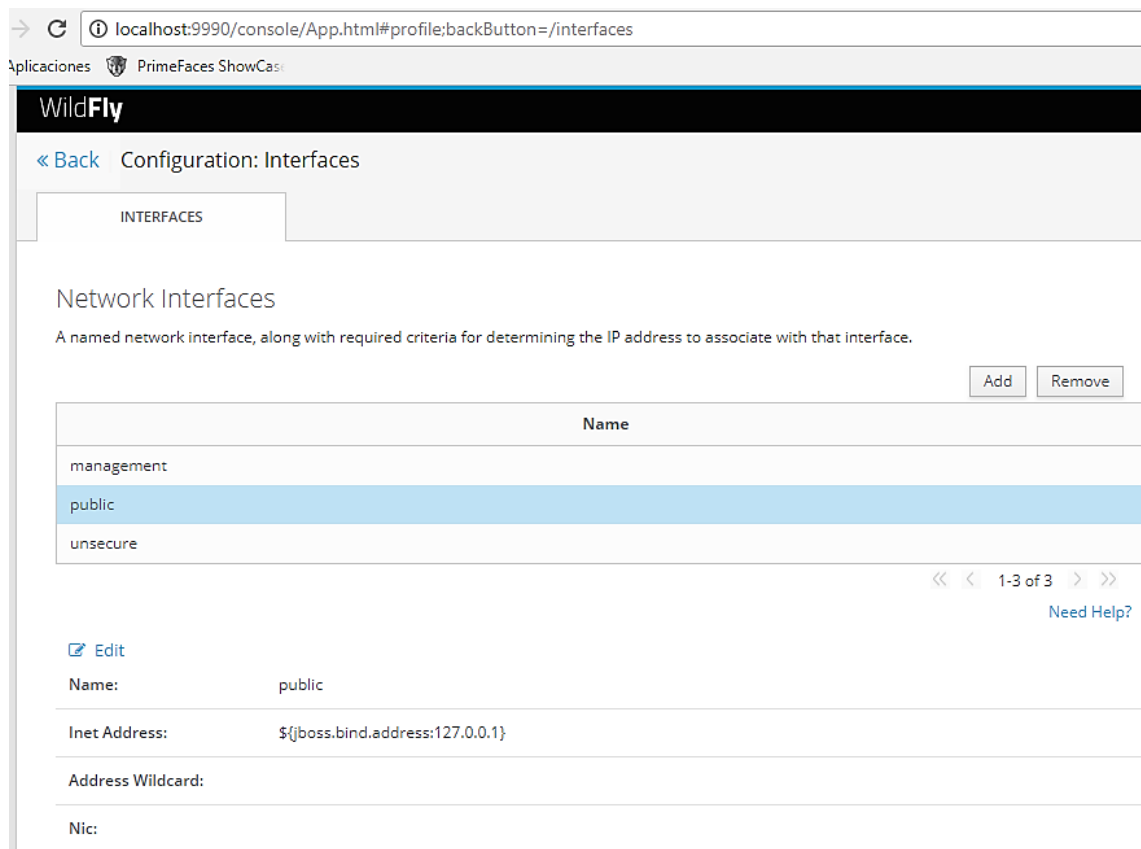


Figura 7. Interfaces de conexión

Esto significa que el servidor en cuestión declara dos interfaces: una se conoce como "management"; y la otra "public". La interfaz de "management" se utiliza para todos los componentes y servicios que requiere la capa de gestión (es decir, HTTP Management Endpoint). La vinculación de la interfaz "public" se utiliza para cualquier comunicación de red relacionada con la aplicación (es decir, Web, mensajería, etc.).

- Editar la interfaz public en el campo Inet Adress: "\${jboss.bind.address:0.0.0.0}" o si se desea vincularse a una dirección particular, reemplace 0.0.0.0 con la ip. Ver (Figura 8).

Edit

Name*: public

Inet Address: \${jboss.bind.address:0.0.0.0}

Address Wildcard:

Nic:

Nic Match:

Loopback*: ☐

Loopback Address:

Required fields are marked with an asterisk (*).

► Advanced

Cancel Save

Figura 8: Reemplazo de IP

- Asegúrese de guardar los cambios y reiniciar Wildfly, ver Figura 9.

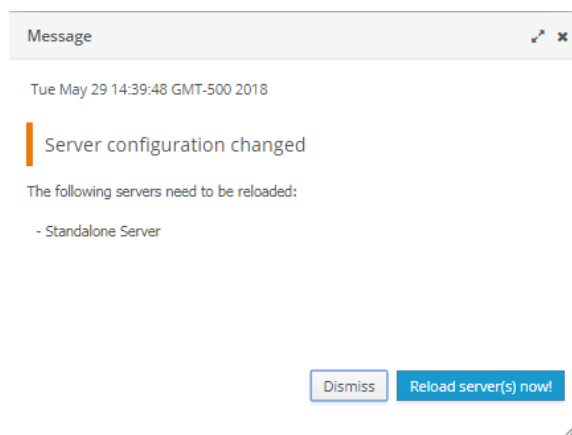


Figura 9: Reinicio de Wildfly

También puede habilitarse la conexión remota editando el archivo standalone.xml en la siguiente sección, guardar los cambios y reiniciar Wildfly.

```
<interfaces>
  <interface name="management">
    <inet-address value="${jboss.bind.address.management:127.0.0.1}"/>
  </interface>
  <interface name="public">
    <inet-address value="${jboss.bind.address:0.0.0.0}"/>
  </interface>
  <interface name="unsecure">
    <inet-address value="${jboss.bind.address.unsecure:127.0.0.1}"/>
  </interface>
</interfaces>
```

2. Servidor de Aplicaciones Cliente

2.1. Dominio de Seguridad

El elemento `<security-realms />` se usa para mantener la definición de uno o más dominios de seguridad. El elemento `<server-identities />` define cómo el servidor puede identificarse. También se configura la identidad secreta, que es la contraseña que usa el servidor cuando se comunica con otro servidor.

- Se procede a editar el archivo `standalone.xml` localizado en:

`${WildFly10_HOME}\standalone\configuration\standalone.xml`

- Aquí se está usando el valor secreto "c2Vydm1kb3I3NEA=", la contraseña codificada en base 64 del usuario "servidor74" que se creó anteriormente. Lo estamos agregando como un dominio de seguridad y le damos el nombre "ejb-security-realmSer74".

```
<security-realms>
  ...
  <security-realm name="ejb-security-realmSer74">
    <server-identities>
      <secret value="c2Vydm1kb3I3NEA=" />
    </server-identities>
  </security-realm>
</security-realms>
```

2.2. Subsistemas remotos

Permite configurar ajustes para conexiones entrantes y salientes para servicios remotos.

2.2.1. Grupo de sockets standard

Una vez ingresado en la consola de administración del servidor ir a: Configuration > Socket Binding > View standard-sockets > Inbound.

The screenshot shows the WildFly administration console. The breadcrumb trail is Configuration > Socket Binding. The left sidebar shows 'Socket Binding Groups' with 'Inbound' selected. The main area is titled 'Socket Bindings: Group standard-sockets' and contains a table of socket configurations.

Name	Port	MCast Port
ajp	\$(jboss.ajp.port:8009)	
http	\$(jboss.http.port:8080)	
https	\$(jboss.https.port:8443)	
iiop	3528	
iiop-ssl	3529	
management-http	\$(jboss.management.http.port:9990)	
management-https	\$(jboss.management.https.port:9993)	
old	4777	

At the bottom of the table, there is a pagination control showing '1-8 of 10' and a 'Need Help?' link.

Figura 10: Grupo de sockets standard

Click en [add] para agregar el socket con el nombre “old” y el puerto 4777.

The screenshot shows the 'Create Socket Binding' dialog box. It has three input fields: 'Name *' with the value 'old', 'Port *' with the value '4777', and 'Group *' with the value 'standard-sockets'. There is a 'Need Help?' link and a note at the bottom stating 'Required fields are marked with an asterisk (*).'. At the bottom right, there are 'Cancel' and 'Save' buttons.

Figura 11: Agregar socket

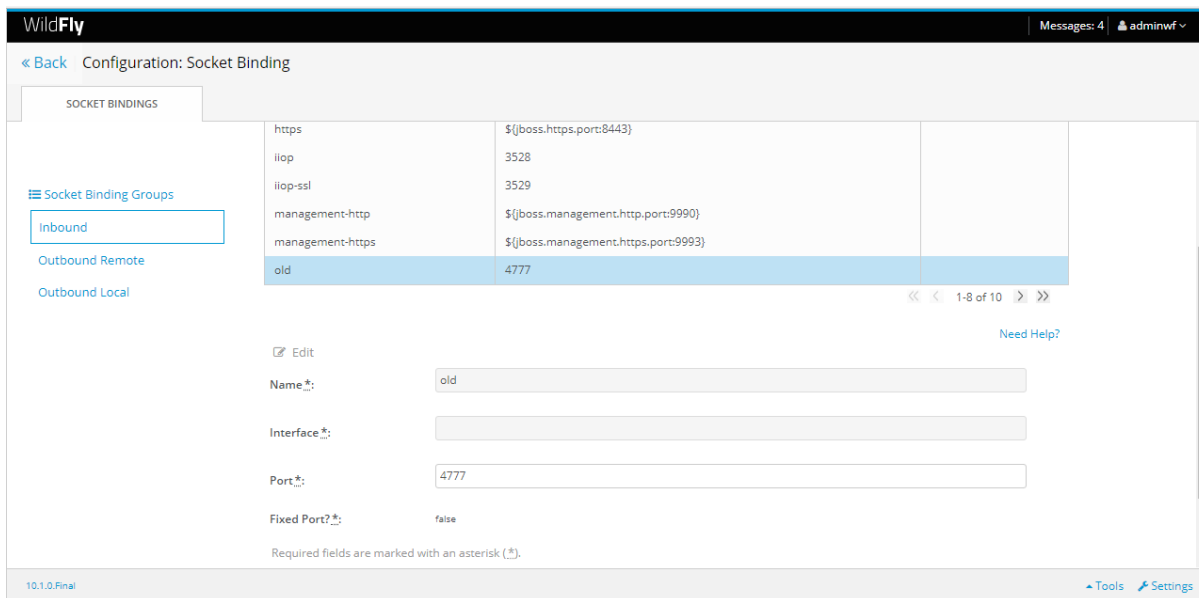


Figura 12: Socket agregado

2.2.2. Conector Remoto

Para configurar un conector remoto, una vez ingresado en la consola de administración del servidor ir a: Configuration > Subsystems > Remoting > pestaña REMOTE CONNECTORS > Conectores.

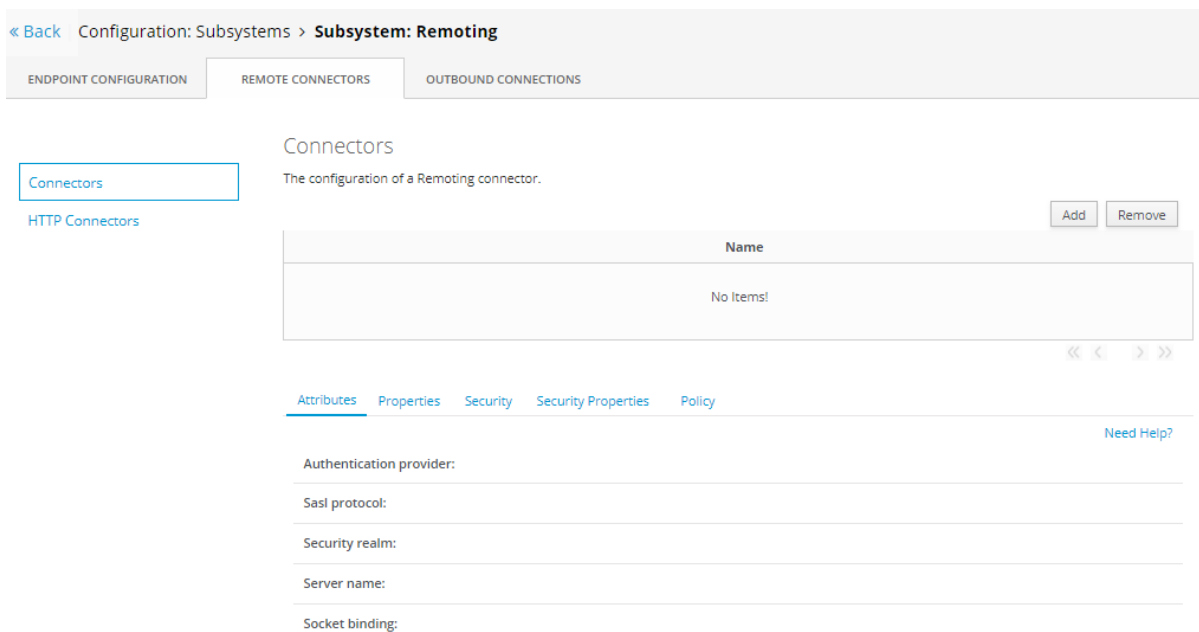


Figura 13: Configurar Conector Remoto

- Clic en [Add] para Añadir el nombre y la conexión de socket (nombre del socket de la sección 2.2.1).

Figura 14: Añadir Conector Remoto

- Luego editar los atributos e ingresar el Dominio de Seguridad “ApplicationRealm” y guardar los cambios.

Figura 15: Atributos de Conector Remoto

2.2.3. Socket de salida remota

Se añade la Información de configuración para el destino remoto, enlace de socket de salida, una vez ingresado en la consola de administración del servidor ir a: Configuration > Socket Binding > View standard-sockets > Outbound Remote.

Remote Socket Bindings: Group standard-sockets
Configuration information for a, remote destination, outbound socket binding.

Socket Binding Groups

- Inbound
- Outbound Remote**
- Outbound Local

Add Remove

Name
mail-smtp
remote-ejb
remote-ejb
remote-ejbR

1-4 of 4

Edit Need Help?

Name: remote-ejb

Host: localhost

Port: 8080

Source Interface:

Source Port: 0

Fixed Source Port?: false

Figura 16: Sockets de salida remota

- Clic en [Add] para Añadir el nombre, el host y el puerto http.

Create Remote Socket Binding

Need Help?

Name*: remote-ejbSer74

Host*: 10.10.10.74

Port*: 8080

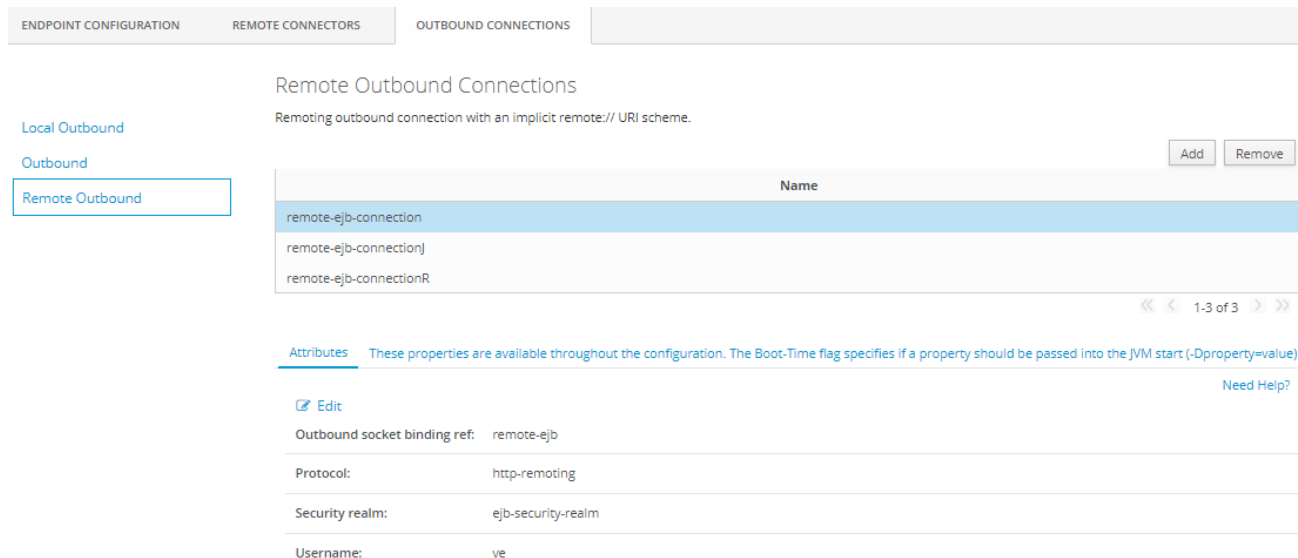
Required fields are marked with an asterisk (*).

Cancel Save

Figura 17: Añadir Socket de salida remota

2.2.4. Conexiones de salida remota

Para configurar las conexiones de salida remota, una vez ingresado en la consola de administración del servidor ir a: Configuration > Subsystems > Remoting > pestaña OUTBOUND CONNECTIONS > Remote Outbound.



The screenshot shows the 'Remote Outbound Connections' configuration page. On the left, there are tabs for 'Local Outbound', 'Outbound', and 'Remote Outbound'. The 'Remote Outbound' tab is selected. The main area shows a table with the following connections:

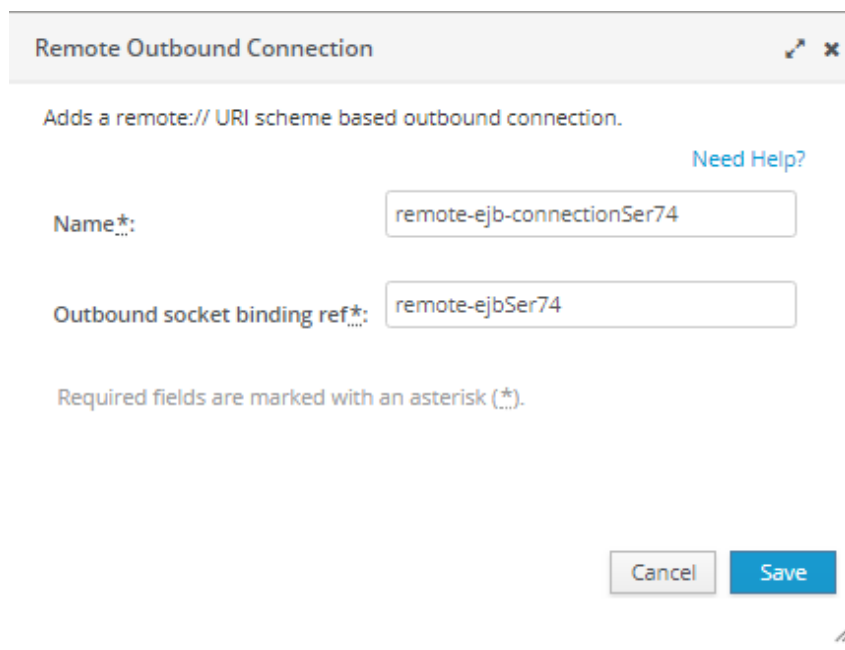
Name
remote-ejb-connection
remote-ejb-connectionj
remote-ejb-connectionR

Below the table, there is a section for 'Attributes' with the following properties:

- Outbound socket binding ref: remote-ejb
- Protocol: http-remoting
- Security realm: ejb-security-realm
- Username: ve

Figura 18: Conexiones de Salida Remota

- Clic en [Add] para añadir la conexión, ingrese el nombre y referencia del socket de la conexión de salida (Sección 2.2.3).



The screenshot shows the 'Remote Outbound Connection' dialog box. It contains the following fields:

- Name*:** remote-ejb-connectionSer74
- Outbound socket binding ref*:** remote-ejbSer74

At the bottom, there are 'Cancel' and 'Save' buttons.

Figura 19: Agregar Conexiones de Salida Remota

- Clic en [Edit] de la conexión creada, para así editar el nombre del dominio de seguridad “ejb-security-realmSer74” (2.1 Dominio de Seguridad) y el usuario de aplicación creado en el servidor contenedor del EJB remoto.

Figura 20: Atributos de Conexión de Salida Remota

Luego ingresar las propiedades de configuración dando clic en [These properties are available throughout the configuration...]. En WildFly 11 clic en pestaña [Properties].

Clic en [Add] para añadir la propiedad “SASL_POLICY_NOANONYMOUS” con el valor “false”

Figura 21: Propiedad “SASL_POLICY_NOANONYMOUS”

Clic en [Add] para añadir la propiedad “SSL_ENABLED” con el valor “false”.

Figura 22: Propiedad “SSL_ENABLED”

2.3. Configuración de parámetros

Configuración de parámetros iniciales para la conexión con EJB de Administración en el servidor de aplicaciones remotas, para ello dirigirse a la ruta:

C:\wildfly-11.0.0.Final\standalone\configuration

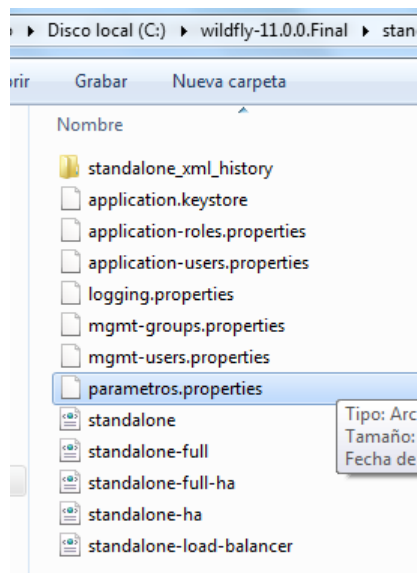


Figura 23: Archivo parametros.properties

Abrir el archivo **[parametros.properties]** y configurar lo siguiente:

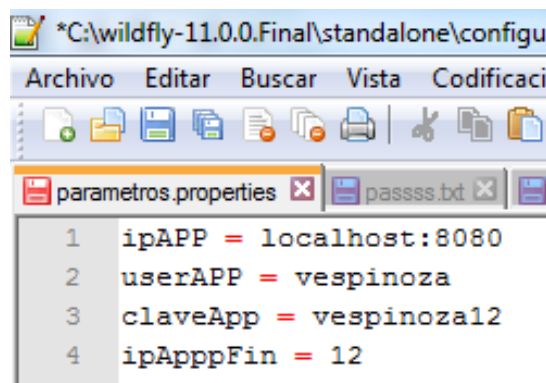


Figura 24: Configuración parametros.properties

ipAPP = 172.16.6.12:8080	IP de servidor de aplicaciones
userAPP = vespinoza	usuario creado en el servidor de aplicaciones
claveApp = vespinoza12	Clave creada en el servidor de aplicaciones
ipAppFin = 12	valor asignado como sufijo para identificar los ejb's o jar's alojados en servidor de aplicaciones

Para el caso del Proyecto SIPE, en el cual ya se disponga de EJB's remotos - jar's y de la base de datos habilitada, se debe realizar la inserción de los EJB'S creados, el cual contiene el catálogo de los mismos con sus parámetros de conexión para su administración.

Para tal efecto ejecutar la siguiente función:

```
SELECT administracion.fn_insert_ejbs(
    'http-remoting://172.16.6.12:8080',
    'vespinoza',
    'vespinoza12',
    12
);
```

Los parámetros son los mismos configurados en el archivo **[parametros.properties]**.

3. CREACIÓN EJB REMOTO

Para crear un EJB remoto se necesita crear una interface, una clase que la implemente y que, además, referencie a la interfaz por medio de la anotación @Remote.

Todo EJB de sesión puede tener dos interfaces, una local y otra remota:

- La interfaz **local** define los métodos del EJB que se pueden invocar desde cualquier otro componente (otro EJB, un Servlet, etc) que se encuentre en el mismo servidor de aplicaciones donde está desplegado el EJB.
- La interfaz **remota** define los métodos del EJB que se pueden invocar por componentes que se encuentren fuera del servidor de aplicaciones donde esté desplegado el EJB. La interfaz remota sólo será necesaria si vamos a llamar al EJB desde otro sitio que no sea el propio servidor de aplicaciones donde está desplegado el EJB.

Cuando se despliega un EJB en el servidor de aplicaciones, este hace un registro de ellos vía JNDI -llamado binding- con los siguientes datos:

`java:global/[<application-name>]/<module-name>/<bean-name>!<fully-qualified-bean-interface-name>`

Dónde:

- application-name: es el nombre de la aplicación. **Solo necesario si se ha desplegado un EAR**. Se deduce por:
 - el nombre del EAR
 - el valor de `<display></display>` del archivo `application.xml` dentro de `META-INF`.
- module-name: es el nombre del módulo al cual pertenece el bean (archivos `.jar`, `.war`) sin la extensión. Esto puede ser sobrescrito en el `ejb-jar.xml`.
- bean-name: indica el nombre simple del bean.
- fully-qualified-bean-interface-name: indica el nombre calificado (con todo y paquete) del bean o de la interface (en caso sea un EJB remoto).

Esta estructura fue impuesta a partir de la especificación 3.1 de EJB y la puedes utilizar en Java EE 7+. Antes de esto, cada proveedor tenía su propia estructura JNDI, no permitiendo hacer las aplicaciones totalmente portables.

Existen también dos tipos más de JNDI:

- `java:module/ejbName!fullEjbInterfaceName`: se usa cuando el EJB remoto está en el mismo módulo.
- `java:app/moduleName/ejbName!fullEjbInterfaceName`: se usa cuando el EJB remoto se encuentra en otro módulo pero en la misma aplicación.

Para empezar se crea un EJB Module normal.

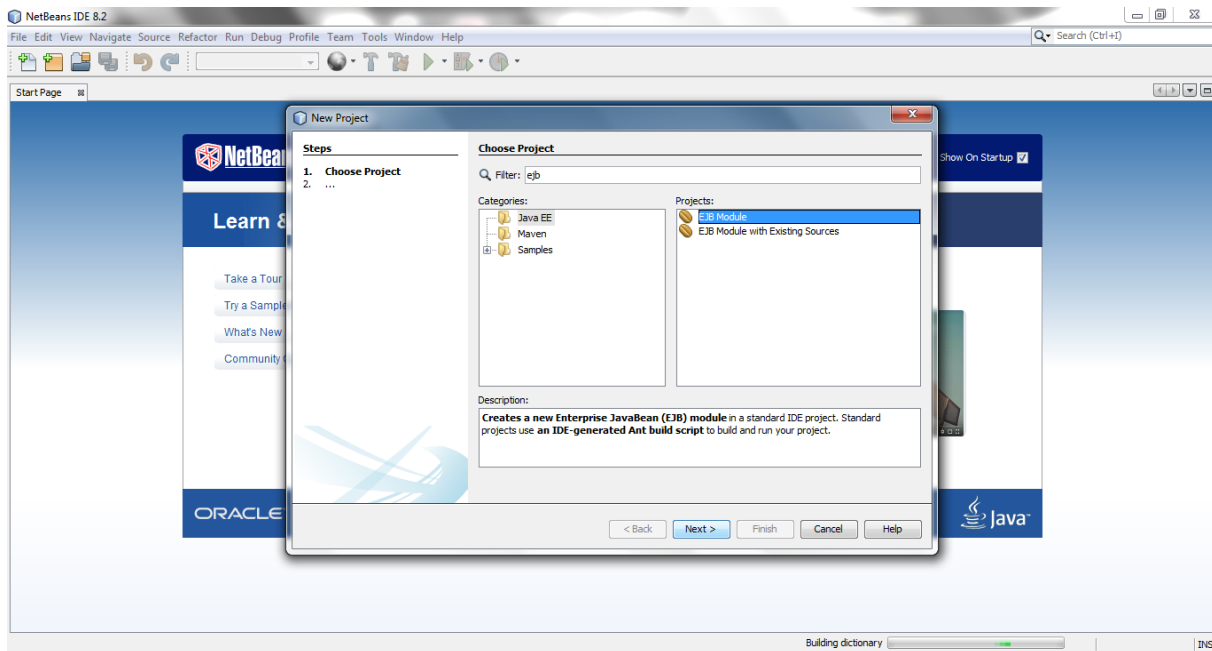


Figura 25: Creación EJB Module

3.1. Creación de interface

Creación de Paquete de Interfaces.

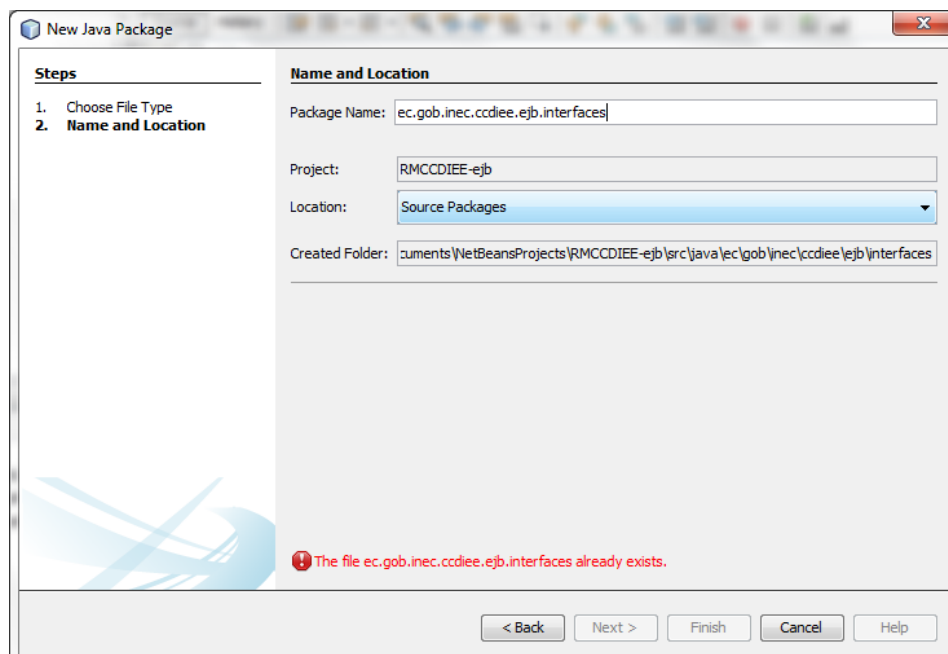


Figura 26: Paquete de Interfaces

- Se crea una Java interface y en ella se definen los métodos que se van a utilizar.

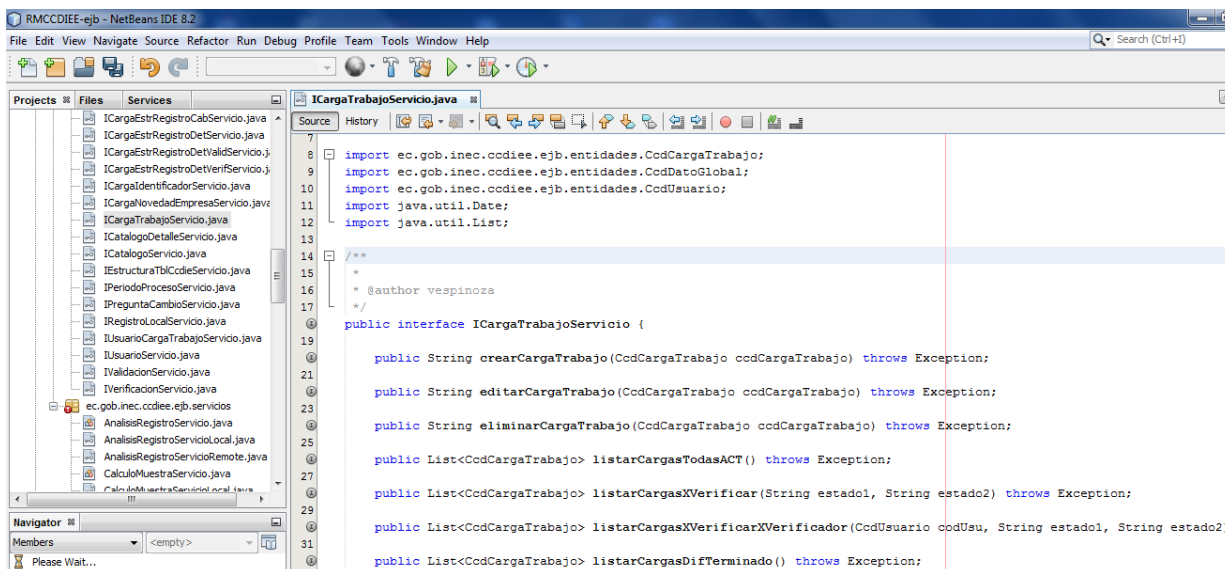


Figura 27: interface del Servicio

3.2. Implementación del servicio

La Implantación de los métodos se la realiza en el servicio, para ellos crear un Session Bean tipo Stateless con Interface Local.

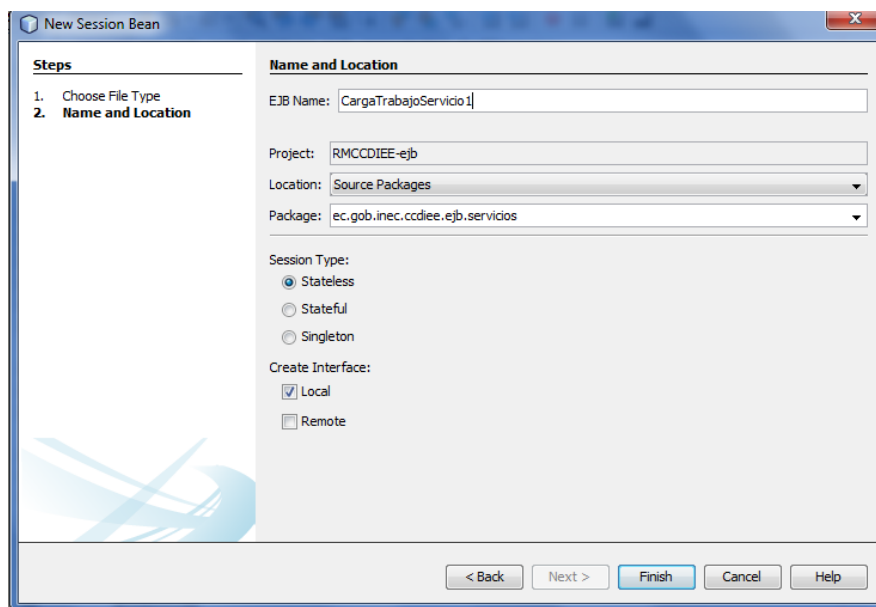


Figura 28. Creación Session Bean

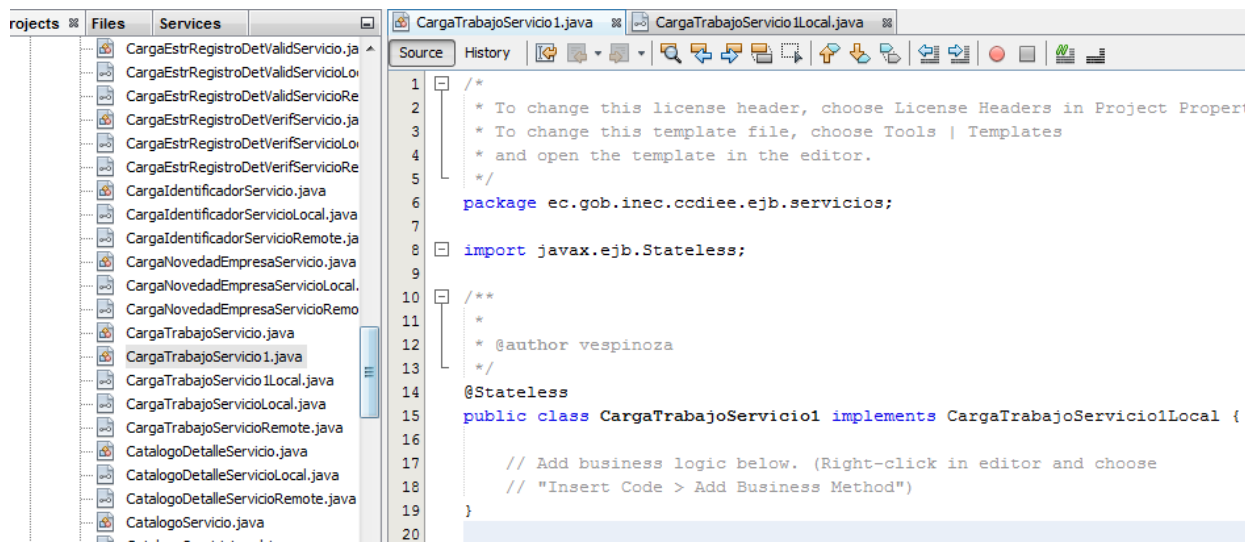


Figura 29: Servicio creado

Heredar a la interfaz local la interfaz creada.

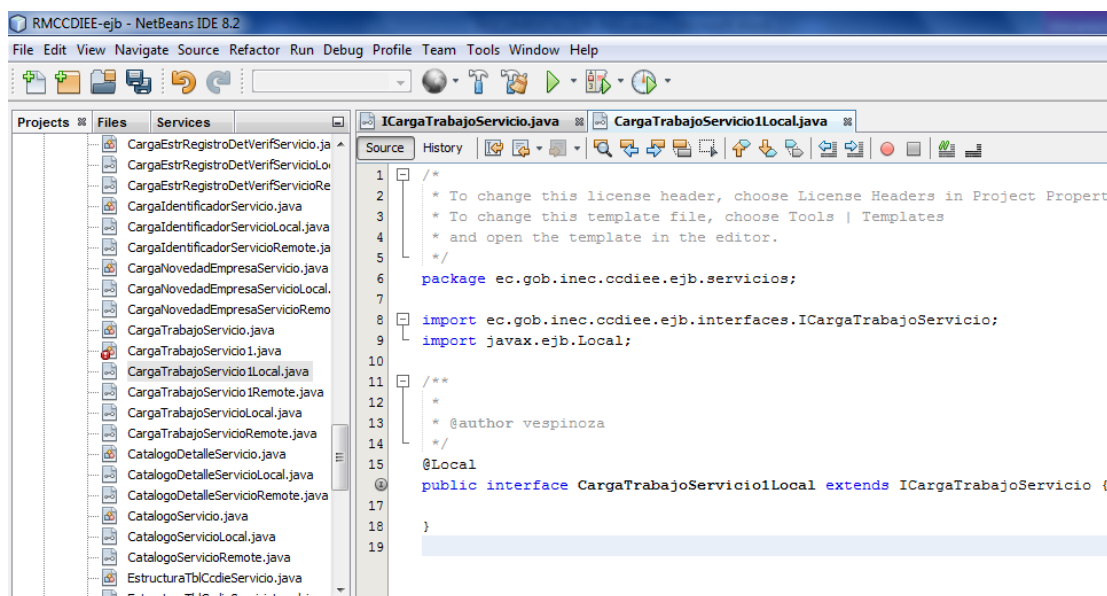


Figura 30: Interface Local

Se crea manualmente la interfaz Remota y de igual forma se hereda a la interfaz creada.

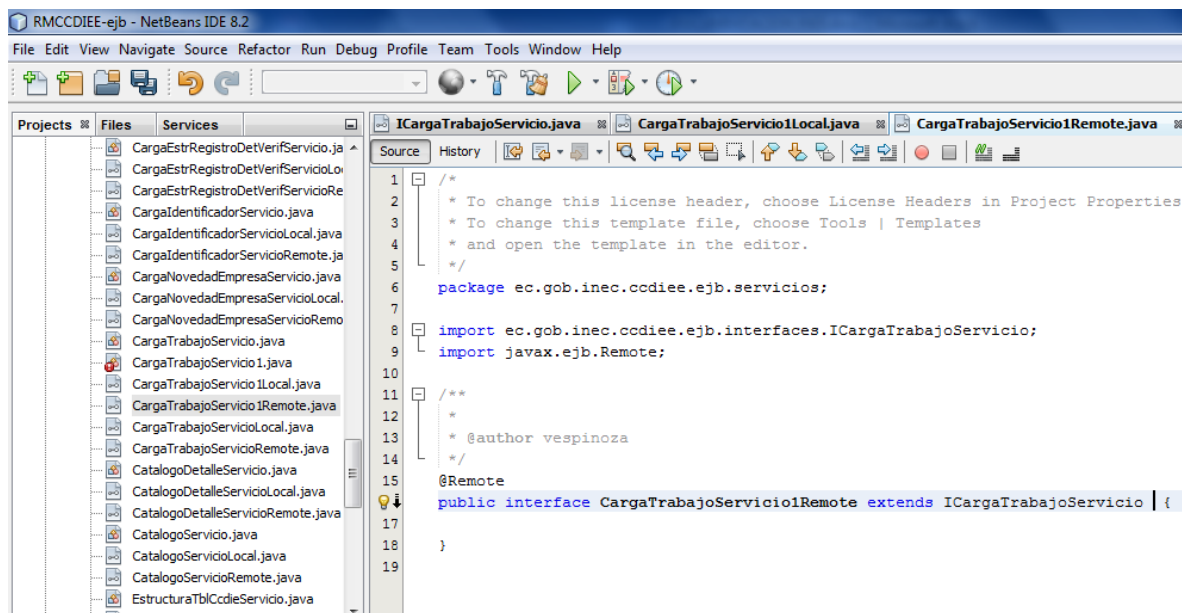


Figura 31: Interfaz Remota

Por último, al servicio creado realizar el “implements” a las interfaces Local y Remota, los cuales heredan los métodos creados. Aquí en el servicio realizar la implementación de los métodos y añadir la notación @Override.

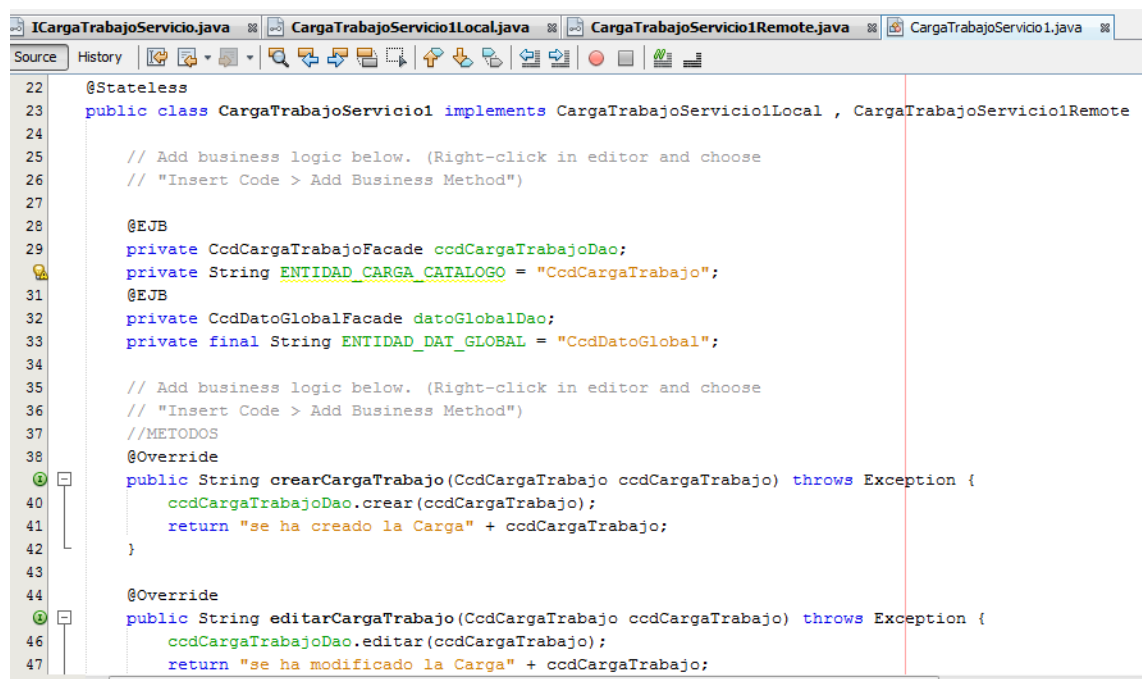


Figura 32: Implementación del servicio

Una vez desplegado el EJB en el servidor, el registro JNDI para el servicio de ejemplo es el siguiente:

```
java:global/RMCCDIEE-ejb/CargaTrabajoServicio!ec.gob.inec.ccdiee.ejb.servicios.CargaTrabajoServicioRemote
java:app/RMCCDIEE-ejb/CargaTrabajoServicio!ec.gob.inec.ccdiee.ejb.servicios.CargaTrabajoServicioRemote
java:module/CargaTrabajoServicio!ec.gob.inec.ccdiee.ejb.servicios.CargaTrabajoServicioRemote
java:jboss/exported/RMCCDIEE-ejb/CargaTrabajoServicio!ec.gob.inec.ccdiee.ejb.servicios.CargaTrabajoServicioRemote
java:global/RMCCDIEE-ejb/CargaTrabajoServicio!ec.gob.inec.ccdiee.ejb.servicios.CargaTrabajoServicioLocal
java:app/RMCCDIEE-ejb/CargaTrabajoServicio!ec.gob.inec.ccdiee.ejb.servicios.CargaTrabajoServicioLocal
java:module/CargaTrabajoServicio!ec.gob.inec.ccdiee.ejb.servicios.CargaTrabajoServicioLocal
```

4. CLIENTE - WEB APPLICATION

Una vez creada una Web Application (war) se procede a realizar las configuraciones necesarias por parte del lado del Cliente.

4.1. Librería Jboss-client.jar

Es necesario al añadir al proyecto Web Application la librería Jboss-client.jar localizada en \${WildFly-10.1_HOME}\bin\client

4.2. Librería EJB remoto

Adicionalmente añadir la librería del EJB remoto creado, para realizar casting a los objetos.

4.3. Referencia a las conexiones de salida

La aplicación del cliente EJB debe incluirse en la carpeta WEB-INF el archivo jboss-ejb-client.xml que contiene una referencia a las conexiones de salida, en este caso “remote-ejb-connectionSer74”, (Sección 2.2.4).

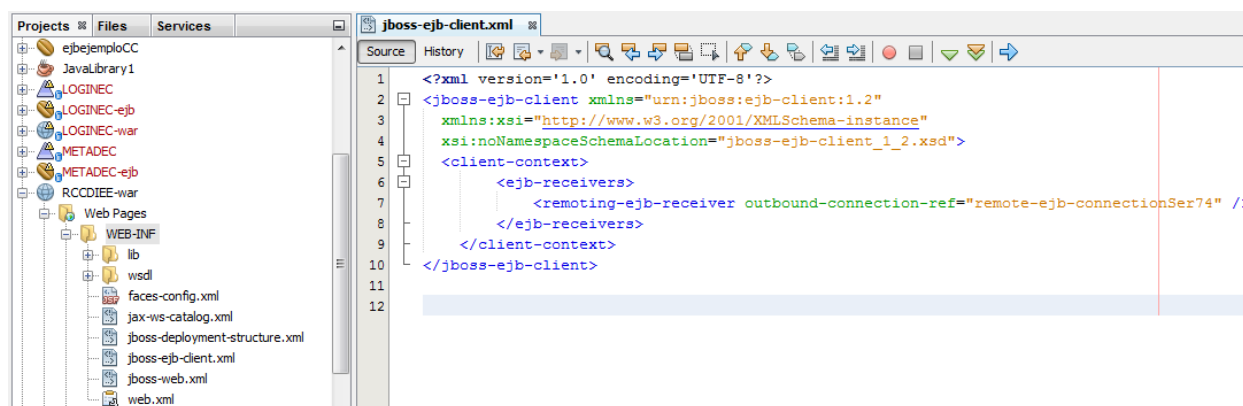


Figura 33: Referencia a las conexiones de salida

4.4. Referencias EJB externas

Una referencia EJB se llama externa cuando el bean proviene de otra unidad de aplicación, que incluso puede implementarse en otro servidor. Se debe proporcionar el nombre JNDI completo del bean en jboss-web.xml.

Web.xml con referencias de EJB externas:

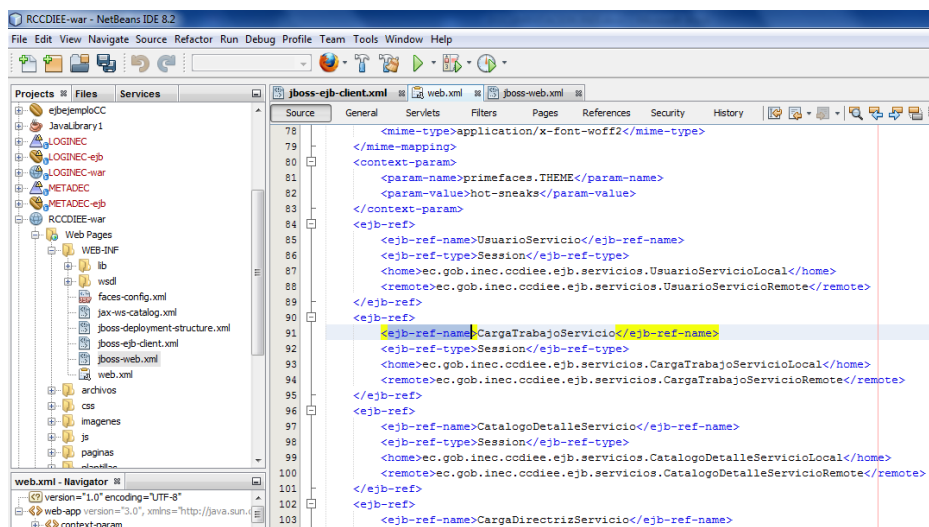


Figura 34: Referencia EJB externa - web.xml

Ejemplo: Referencia de Servicio Remoto

```
<ejb-ref>
  <ejb-ref-name>UsuarioServicio</ejb-ref-name>
  <ejb-ref-type>Session</ejb-ref-type>
  <home>ec.gob.inec.codiie.ejb.servicios.UsuarioServicioLocal</home>
  <remote>ec.gob.inec.codiie.ejb.servicios.UsuarioServicioRemote</remote>
</ejb-ref>
```

4.5. Contexto JNDI

Se definen las propiedades JNDI dependiendo del servidor de aplicaciones, además se define la IP donde se encuentra el EJB remoto, puerto http, el usuario de aplicación y su contraseña. Luego se inicia el contexto pasando como parámetros las propiedades JNDI.

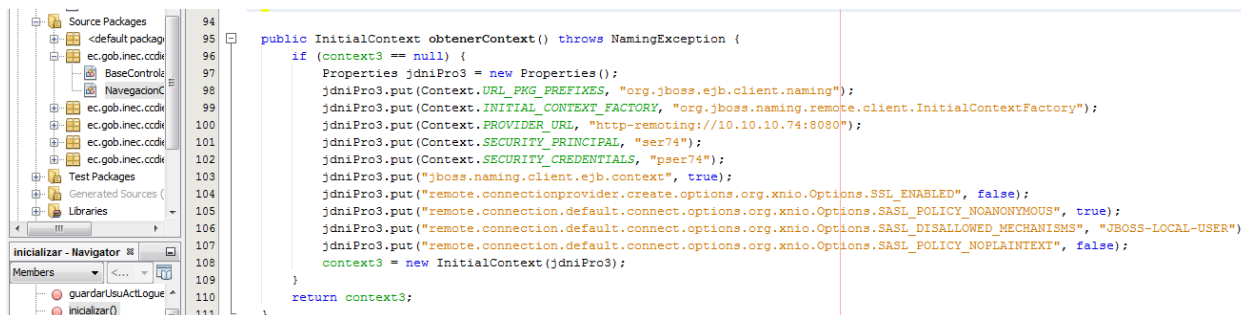


Figura 35: Contexto y Paramtros JNDI

Código:

```

InitialContext context3 = null;
public InitialContext obtenerContext() throws NamingException {
    if (context3 == null) {
        Properties jndiPro3 = new Properties();
        jndiPro3.put(Context.URL_PKG_PREFIXES, "org.jboss.ejb.client.naming");
        jndiPro3.put(Context.INITIAL_CONTEXT_FACTORY, "org.jboss.naming.remote.client.InitialContextFactory");
        jndiPro3.put(Context.PROVIDER_URL, "http-remoting://10.10.10.74:8080");
        jndiPro3.put(Context.SECURITY_PRINCIPAL, "ser74");
        jndiPro3.put(Context.SECURITY_CREDENTIALS, "pser74");
        jndiPro3.put("jboss.naming.client.ejb.context", true);
        jndiPro3.put("remote.connectionprovider.create.options.org.xnio.Options.SSL_ENABLED", false);
        jndiPro3.put("remote.connection.default.connect.options.org.xnio.Options.SASL_POLICY_NOANONYMOUS", true);
        jndiPro3.put("remote.connection.default.connect.options.org.xnio.Options.SASL_DISALLOWED_MECHANISMS", "JBoss-LOCAL-USER");
        jndiPro3.put("remote.connection.default.connect.options.org.xnio.Options.SASL_POLICY_NOPLAINTEXT", false);
        context3 = new InitialContext(jndiPro3);
    }
    return context3;
}

```

4.6. Lookup

Para recuperar datos, usamos el método `context.lookup()`, pasando como parámetro el registro JNDI. Este método devuelve un *Object*, así que es cosa nuestra hacer el *cast* al objeto real.

Registro JNDI ejemplo:

`context.lookup("RMCCDIEE-ejb/UsuarioServicioec.gob.inec.ccdiee.ejb.servicios.UsuarioServicioRemote");`

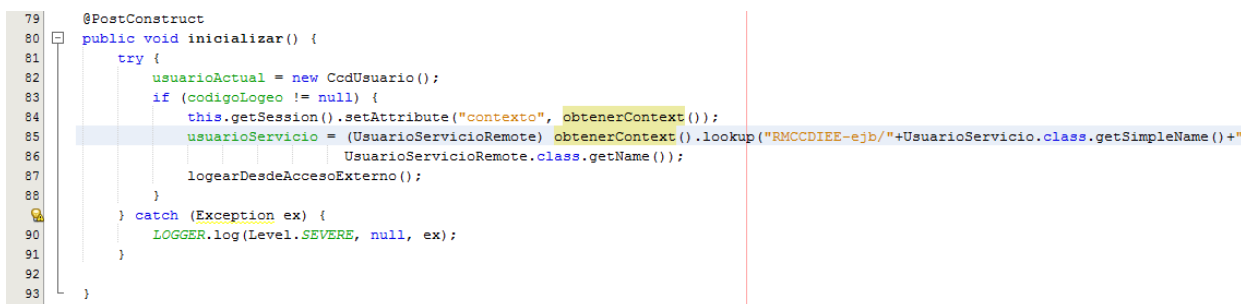


Figura 36: Lookup

4.7. Declaración del Servicio en el controlador.

```
//SERVICIOS
//@EJB
//private UsuarioServicio usuarioServicio;
private UsuarioServicioRemote usuarioServicio;
InitialContext context3 = null;

public NavegacionControlador() {
}

@PostConstruct
public void inicializar() {
    try {
        usuarioActual = new CodUsuario();
        if (codigoLogeo != null) {
            this.getSession().setAttribute("contexto", obtenerContext());
            usuarioServicio = (UsuarioServicioRemote) obtenerContext().lookup("RMCCDIEE-ejb/"+UsuarioServicio.class.getSimpleName()
                + UsuarioServicioRemote.class.getName());
            logearDesdeAccesoExterno();
        }
    } catch (Exception ex) {
        LOGGER.log(Level.SEVERE, null, ex);
    }
}
```

Figura 37: Declaración del Servicio

5. DESPLIEGUE DE EJB'S – JAR'S

Tome en cuenta que la implementación se puede realizar en un solo servidor o en varios servidores dependiendo de la cantidad de información y procesamiento que se realiza a la misma.

Una vez configurados los servidores, implementada la BDD, creado el Data Source en el servidor de aplicaciones (para proyecto SIPE: `jndi-name="java:/sipeDS"`), se deben desplegar los EJB'S – JAR'S en el siguiente orden:

5.1. Implementación en el Servidor de Aplicaciones

1. sipe-entidades-jar.jar
2. ldapjdk_4.1.jar
3. sipe-negocio-jar.jar
4. sipe-conexion.jar
5. sipe-dato.jar
6. sipe-administracion-ejb.jar
7. sipe-captura-ejb.jar
8. sipe-distribuciontrabajo-ejb.jar
9. sipe-geografia-ejb.jar
10. sipe-metadato-ejb.jar
11. sipe-muestra-ejb.jar
12. sipe-proceso-ejb.jar
13. sipe-reportes-ejb.jar

14. sipe-seguridad-ejb.jar

15. sipe-servicioSV-war.war (*)

- El orden del 1-5 es obligatorio, en adelante es indefinido.
- *Si se despliega el sipe-servicioSV-war.war en el servidor de aplicaciones, configurar la IP en los archivos [js_salto_mod, js_validacion_mod] en el servidor web en la ruta C:\wildfly-11.0.0.Final\welcome-content\js\saltosValidaciones:

Archivo	Línea
js_salto_mod	# 31 : var url = "http://172.16.2.240:8080/sipe-servicioSV-war/servicio/salto/form/" + codSeccion + "";
js_validacion_mod	# 283: var url = " http://172.16.2.240:8080/sipe-servicioSV-war/servicio/validacion/form/" + codSeccion + "";

5.2. Implementación en el Servidor Web

1. sipe-entidades-jar.jar
2. ldapjdk_4.1.jar
3. sipe-negocio-jar.jar
4. sipe-cache-ejb.jar
5. sipe-servicioSV-war.war (*)
6. sipe-administracion-war.war
7. sipe-captura-war.war
8. sipe-distribuciontrabajo-war.war
9. sipe-geografia-war.war
10. sipe-metadato-war.war
11. sipe-muestra-war.war
12. sipe-proceso-war.war
13. sipe-reportes-war.war
14. sipe-seguridad-war.war

- El orden del 1-5 es obligatorio, en adelante es indefinido.
- *Si se despliega el sipe-servicioSV-war.war en el servidor web, configurar el Data Source en el mismo (para proyecto SIPE: `jndi-name="java:/sipeDS"`).
- En la carpeta [welcome-content], de la raíz del contenedor web coloque los archivos necesarios en las siguientes carpetas (con el objetivo de realizar cambios sin necesidad de deployar nuevamente los WAR, y compartir esta información entre WAR)
 - js (Archivos JavaScript)
 - css (Archivos de estilos)
 - imagenes (Archivos de imágenes)

- En la carpeta [modules], de la raíz del contenedor web coloque los archivos necesarios (librerías para el despliegue de las aplicaciones).

5.3. Arquitectura lógica proyectos

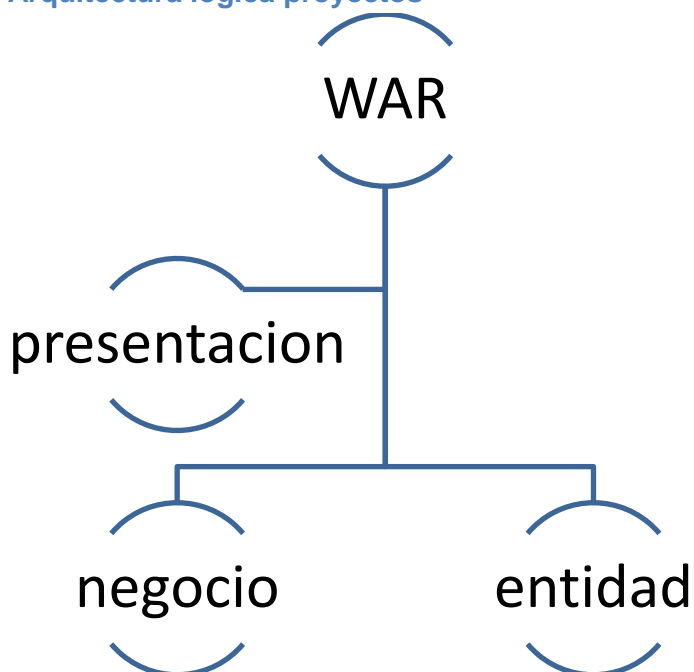


Figura 38: Esquema lógico Contenedor WEB

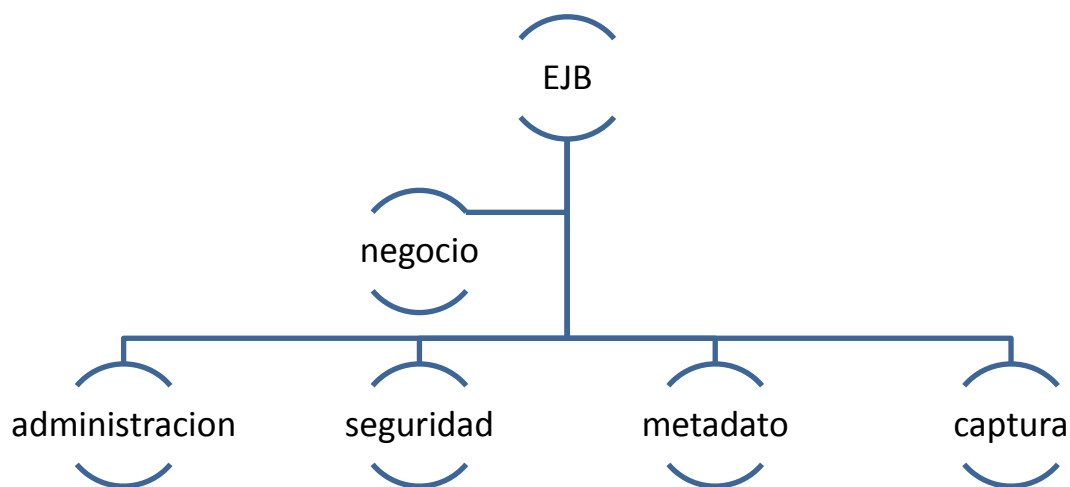


Figura 39: Esquema lógico Servidor Aplicación

Fecha de creación: 28 de mayo de 2018

Fecha de actualización: 20 de noviembre de 2018



www.ecuadorencifras.gob.ec



@ecuadorencifras



INEC/Ecuador



Inec



INECEcuador



INEC Ecuador