



Eusko Jaurlaritzaren Informatika Elkarte  
Sociedad Informática del Gobierno Vasco

UDA – Utilidades de desarrollo de aplicaciones

## Plugin UDA. Guía de uso del plugin

Fecha: 10/06/2011

Referencia:

EJIE S.A.  
Mediterráneo, 14  
Tel. 945 01 73 00  
Fax. 945 01 73 01  
01010 Vitoria-Gasteiz  
Posta-kutxatila / Apartado: 809  
01010 Vitoria-Gasteiz  
[www.ejje.es](http://www.ejje.es)



*UDA – Utilidades de desarrollo de aplicaciones* by [EJIE](#) is licensed under a [Creative Commons Reconocimiento-NoComercial-CompartirIgual 3.0 Unported License](#).

## Control de documentación

Título de documento: Plugin UDA. Guía de Uso del Plugin

### Histórico de versiones

| Código: | Versión: | Fecha:     | Resumen de cambios:  |
|---------|----------|------------|--|
|         | 1.0.0    | 10/06/2011 | Primera versión.   |
|         | 1.1.0    | 14/09/2011 | Cambios en las pantallas, por la nueva versión del plugin. |
|         |          |            |  |
|         |          |            |  |

### Cambios producidos desde la última versión

Cambios en las pantallas, por la nueva versión del plugin.

### Control de difusión

Responsable: Ander Martinez

Aprobado por:

Firma:

Fecha:

Distribución:

### Referencias de archivo

Autor:

Nombre archivo:

## Contenido

|     | Capítulo/sección                                | Página |
|-----|---|--------|
| 1   | Introducción                                    | 1      |
| 2   | Configuración de preferencias de UDA            | 2      |
| 3   | Asistentes UDA                                  | 5      |
| 3.1 | Crear nueva aplicación                          | 5      |
| 3.2 | Añadir un proyecto WAR a una aplicación         | 11     |
| 3.3 | Generar código para una aplicación              | 15     |
| 3.4 | Crear nuevo mantenimiento                       | 21     |
| 3.5 | Añadir un proyecto EJB a una aplicación         | 36     |
| 3.6 | Generar código para EJB cliente                 | 38     |
| 3.7 | Generar código para EJB servidor                | 42     |
| 4   | Acceso a una aplicación UDA                     | 45     |
| 4.1 | Generar el datasource en la consola de Weblogic | 45     |
| 4.2 | Desplegar la aplicación                         | 47     |
| 5   | Barra de herramientas de patrones UDA           | 50     |
| 5.1 | Tipos de patrones UDA                           | 50     |
| 5.2 | Utilizar un snippet                             | 50     |

## 1 Introducción

UDA (Utilidades de Desarrollo de Aplicaciones) es el conjunto de utilidades, herramientas, librerías, plugins, guías, y recomendaciones funcionales y técnicas que permiten acelerar el proceso de desarrollo de sistemas software con tecnología Java impulsado por EJIE/EJGV. Sus principales señas de identidad frente a sus predecesores son la **simplicidad y productividad** en el desarrollo y sus capacidades de interfaz gráfico para la construcción de aplicaciones ricas (RIA, Rich Internet Applications).

Entre las utilidades que proporciona UDA hay un conjunto de asistentes que se instalan como plugin en el entorno de desarrollo agilizando la implementación de los proyectos mediante la generación de código por capas siguiendo el paradigma MVC (Modelo – Vista – Controlador).

El objetivo de este documento es explicar de manera práctica la configuración y la utilización de los asistentes del plugin UDA en el Eclipse que guiarán al usuario en la generación de aplicaciones, EJB y nuevos mantenimientos, entre otros.

Los asistentes de UDA son:

- Crear nueva aplicación
- Generar código para una aplicación
- Crear nuevo mantenimiento
- Añadir un proyecto WAR a una aplicación
- Añadir un proyecto EJB a una aplicación
- Generar código para EJB Cliente
- Generar código para EJB Servidor

Inicialmente, indicaremos como configurar el plugin UDA en el entorno Eclipse y posteriormente veremos cómo utilizarlo paso a paso y qué genera cada asistente.

También se indicará cómo utilizar la barra de herramientas de snippets de los patrones de presentación de UDA para facilitar la generación del código de estos patrones.

## 2 Configuración de preferencias de UDA

Primero obtendremos un Eclipse Helios plataformado con las diversas herramientas (oepe, subversive svn, pmd, checkstyle, findbugs, hibernate tools, plugin UDA, plantillas de generación de código) necesarias para utilizar UDA.

- Arrancamos el Eclipse y nos dirigimos a la configuración de sus preferencias generales en el menú Windows > Preferences.

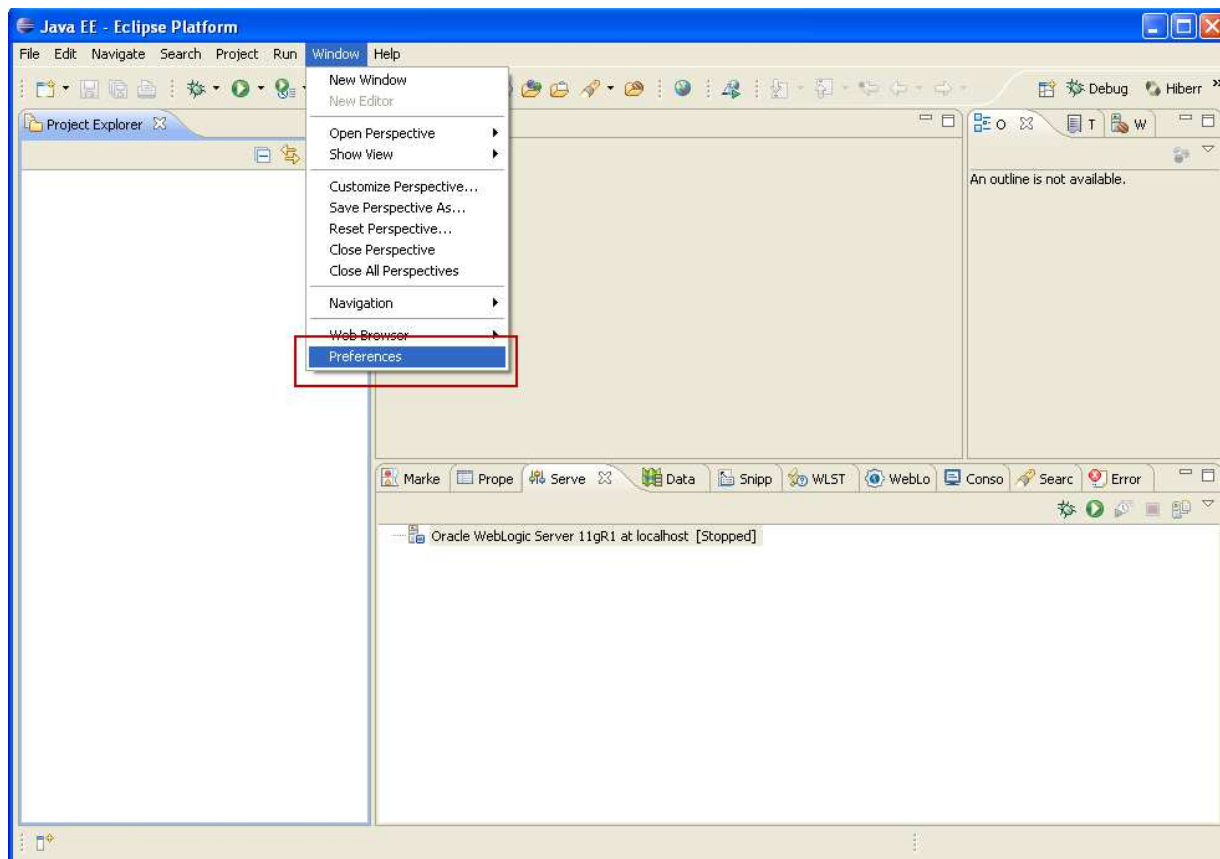


Ilustración 1. Preferencias del Eclipse.

- Luego seleccionamos la categoría UDA de las preferencias y configuramos los siguientes campos.

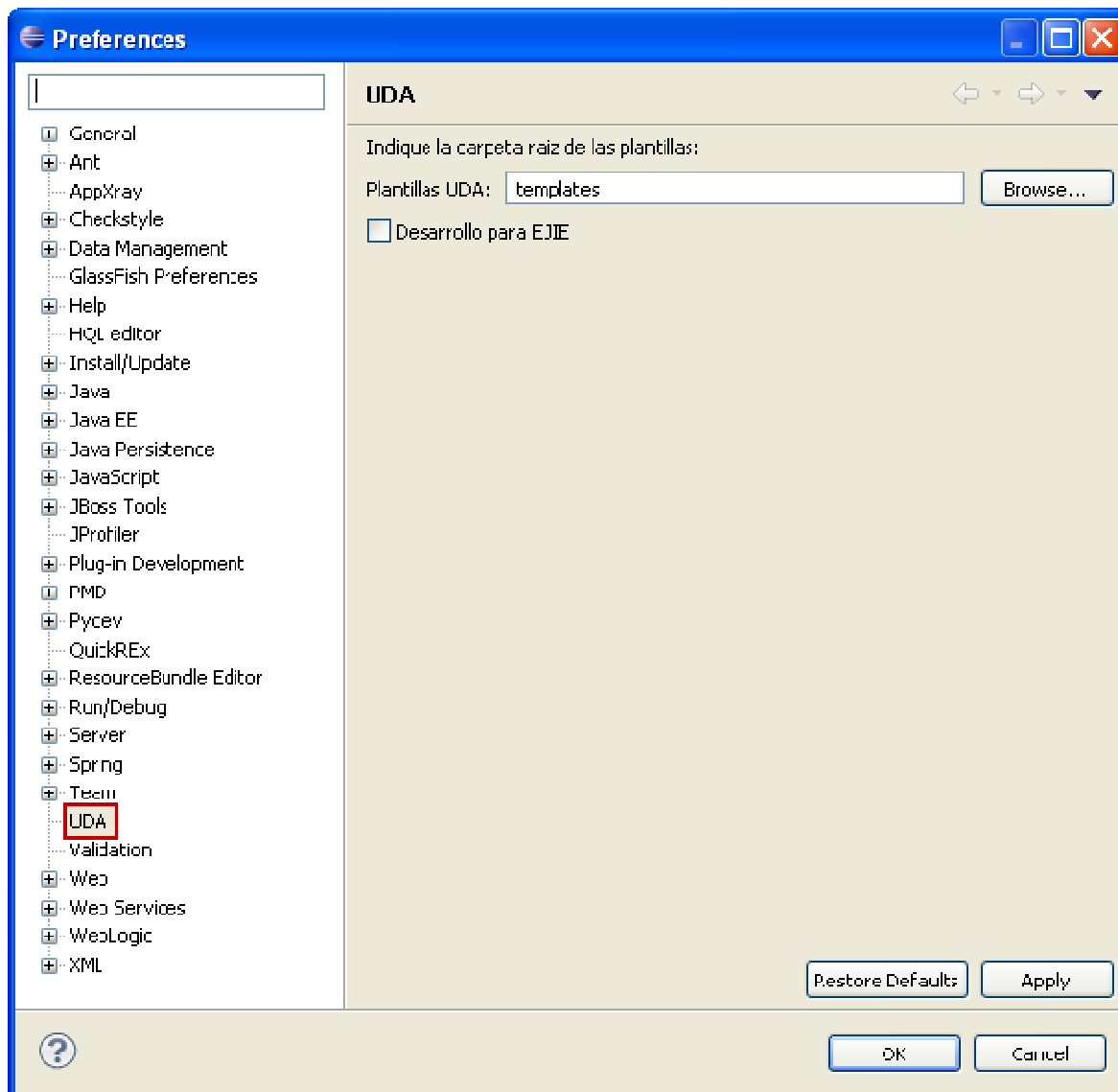


Ilustración 2. Configuración de UDA.

En el campo de Plantillas UDA indicamos la ruta dónde están situadas las plantillas proporcionadas en la instalación del plugin UDA. A través de la plantillas los asistentes serán capaces de generar los ficheros necesarios en sus ejecuciones.

También hay un checkbox Desarrollo para EJIE que activaremos en el caso que se vaya a desarrollar una aplicación para EJIE y lo generado vaya vinculado a las configuraciones específicas del entorno EJIE. Si está seleccionado implicará los siguientes cambios en los asistentes, por ejemplo:

- En los asistentes de Crear nueva aplicación y Añadir un proyecto WAR a una aplicación cambiarán la parte relativa a los layouts de disposición de pantalla. Cuando esté habilitado hará que aparezca las opciones para elegir el layout de la aplicación según el Tipo de Aplicación y la Categoría.
- En los asistentes de *Crear nueva aplicación* y *Añadir un proyecto WAR a una aplicación* cambiarán la parte relativa a los idiomas. Cuando esté habilitado se deshabilita la parte de los idiomas base indicando que los idiomas de castellano y euskera serán obligatorios.

- En el asistente de *Generación de código para una aplicación* habrá una pantalla más donde se configurará la seguridad XLNets para el entorno EJIE.
- El fichero *pom.xml* de Maven, donde se configura la gestión de dependencias, apuntará al repositorio de EJIE y no a los repositorios públicos de Maven.

En caso de no tener acceso al repositorio Maven de EJIE, será necesario registrar la siguiente librería (x38ShLibClasses.jar) que se proporciona en un repositorio Maven al que se tenga acceso. En caso contrario, el propio repositorio proporcionado por EJIE dispondrá dicha librería registrada.

Para realizar el registro de la librería X38ShLibClasses.jar en el repositorio maven local es necesario ejecutar el siguiente comando:

```
mvn install:install-file -Dfile=x38ShLibClasses.jar -DgroupId=com.ejie.x38  
-DartifactId=x38ShLibClasses -Dversion=1.0 -Dpackaging=jar
```

Con estos pasos tendremos configurado el plugin UDA para la utilización de sus asistentes.

## 3 Asistentes UDA

### 3.1 Crear nueva aplicación

Este asistente se encarga de generar una nueva aplicación que consiste en

- generar varios proyectos en el workspace de Eclipse,
- relacionarlos entre sí a nivel de configuración,
- vincularlos al runtime del Weblogic instalado en Eclipse y
- obtener las librerías necesarias de manera automática a través de Maven.

Se generan cinco proyectos en el workspace de Eclipse:

- o Proyecto de Configuración (*[nombre de la aplicación]Config*)
- o Proyecto de Estáticos (*[nombre de la aplicación]Statics*)
- o Proyecto de Clases (*[nombre de la aplicación]EARClasses*)
- o Proyecto WAR (*[nombre de la aplicación]WAR*)
- o Proyecto EAR (*[nombre de la aplicación]EAR*)

Y, de manera opcional, se podrá generar proyecto EJB:

- o Proyecto EJB (*[nombre de la aplicación]EJB*) → Opcional.

También crea una carpeta en C:\datos\xxx\log, donde se sitúan los logs de la aplicación, siendo xxx el código de aplicación.



- Para crear una nueva aplicación seleccionamos en el menú File > New > Other... o Ctrl+N.

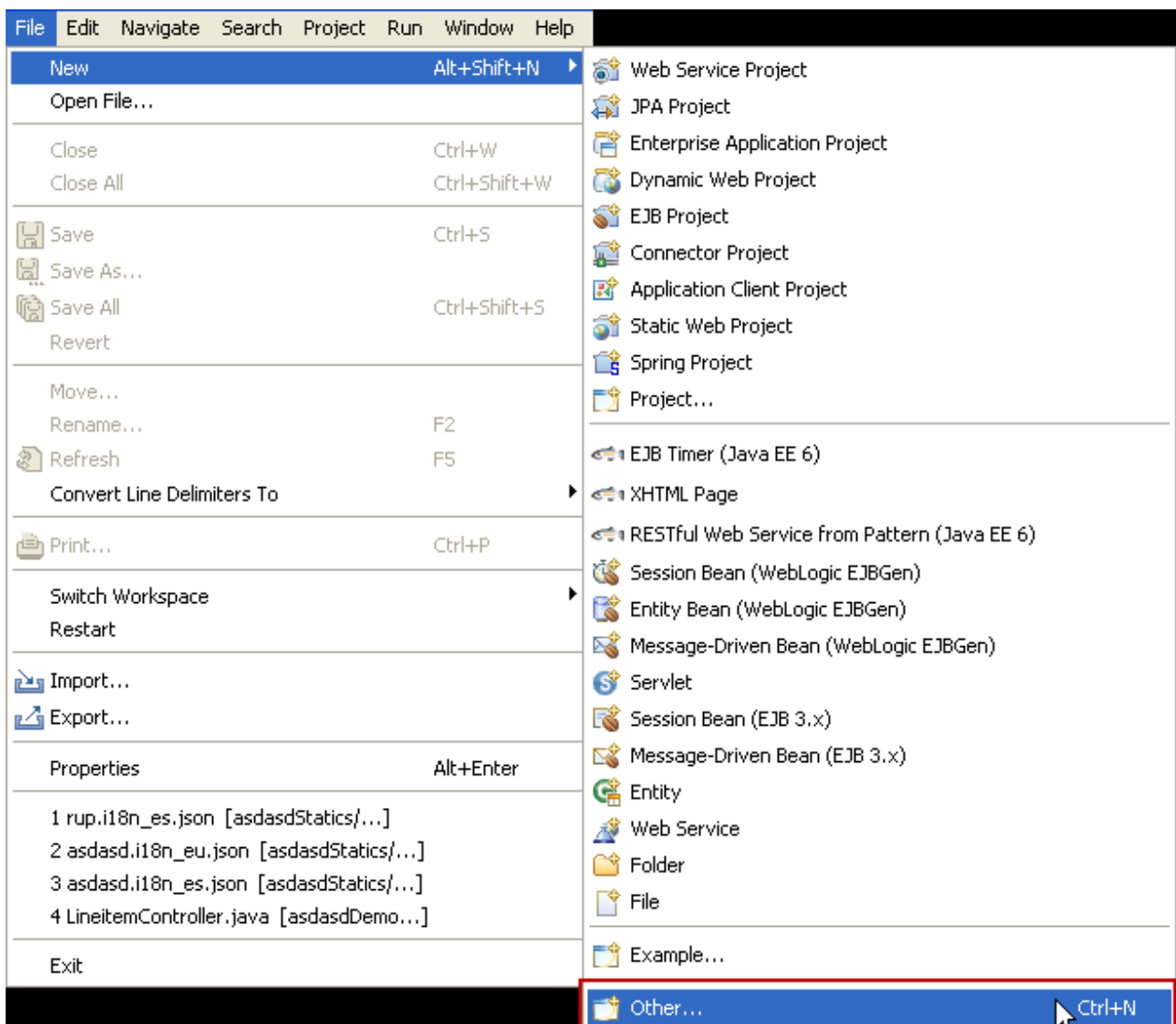


Ilustración 3. Selección de nuevo asistente

- En la categoría UDA seleccionamos la opción *1. Crear nueva aplicación*

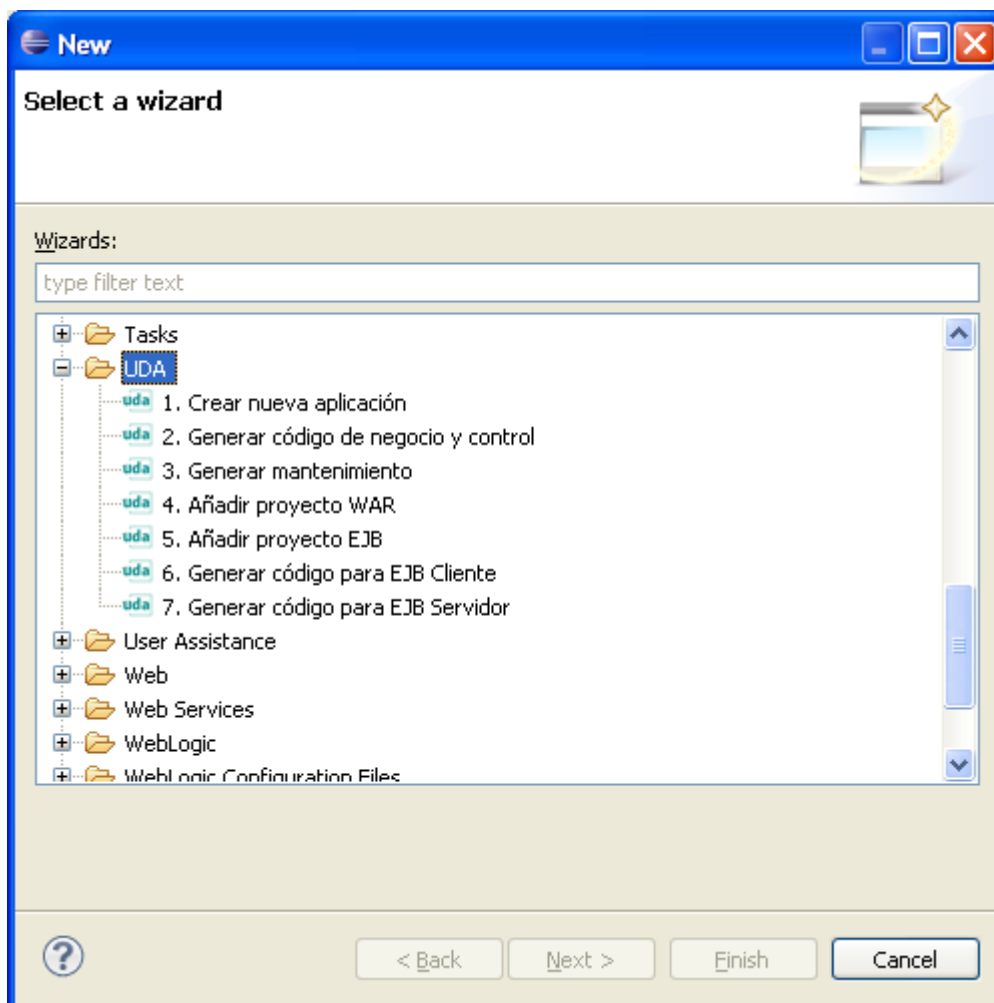


Ilustración 4. Listado de asistentes UDA – Crear nueva aplicación

- En la pantalla *Crear nueva aplicación* rellenamos los campos para generar una nueva aplicación.



Ilustración 5. Pantalla de asistente de Crear nueva aplicación

- **Código de aplicación:** Campo obligatorio del nombre de la aplicación (siempre estará en minúsculas y se validará si se ha informado).
- **Usar localización por defecto:** Este check cuando está habilitado indica que los proyectos se van a crear en la carpeta del workspace definido en el Eclipse, en caso contrario se habilita el campo de Localización para que se informe otra ruta.
- **Localización:** Indica la ruta donde se quiere generar los proyectos. Para el proyecto de configuración siempre se va a crear en la ruta de C:\config\dominio\_desa\[nombre de la aplicación].

*NOTA: Si se configura otro dominio para el servidor distinto a dominio\_desa, habría que mover el proyecto generado desde esta carpeta a la ubicación correspondiente del dominio de trabajo.*

- **Nombre del WAR:** Campo obligatorio del nombre del nuevo proyecto WAR. La primera letra siempre será en mayúscula y en el resto de la cadena se respetarán las mayúsculas y minúsculas puestas por el usuario (*CamelCase*).
- **Nombre Completo del WAR:** Es un campo de sólo lectura que muestra el nombre que tendrá el proyecto WAR. El nombre completo es la concatenación de [*código de aplicación*] + [*nombre del WAR*] + War.  
  
Por ejemplo: *x21aDemoWar*, donde el nombre de la aplicación es *x21a* y *Demo* es el nombre del WAR.
- **Generar ejemplos de código:** Si está seleccionado se generarán ejemplos de los patrones de presentación en la aplicación.  
  
Para que todos los ejemplos funcionen correctamente, se aconseja ejecutar el script proporcionado de generación de la base de datos de ejemplo de UDA.
- **Persistencia:** Se elige el tipo de tecnología de persistencia con el que se va a desarrollar la aplicación Spring JDBC o JPA 2.0.
- **Layout:** En este apartado se define la disposición de pantalla que tendrá la aplicación relacionada con los logotipos en la cabecera y los menús atendiendo a ciertos criterios de organización de pantalla. Esta disposición afectará al fichero de configuración *tiles.xml*.
  - **Disposición:** Definimos si la disposición de la organización del menú será en horizontal, vertical o mixto.
  - **Tipo de aplicación:** Si el check de *Desarrollo para Ejie* está habilitado aparecerá esta opción donde informaremos si la aplicación es de tipo Intranet/Extranet/JASO o Internet generando una cabecera de aplicación con logotipos o una cabecera ficticia a eliminar cuando se integre en la herramienta de Gestión de Portales.
  - **Categoría:** Si el check de *Desarrollo para Ejie* está habilitado aparecerá esta opción donde informaremos si se trata de una aplicación Horizontal o Departamental en función del responsable del aplicativo apareciendo diferentes logotipos en cada caso.  
  
Este campo está relacionado con el campo **Tipo Aplicación**. Si la aplicación es de tipo Internet nunca podrá ser de categoría Departamental. En este caso se deshabilita el combo de Categoría y se pone su valor a Horizontal.
- **Idiomas:** Configuración de idiomas que puede tener la aplicación.
  - **Base:** Define los idiomas base, las opciones son checks de castellano y euskera. Cuando el check de *Desarrollado para Ejie* esté habilitado en la configuración del plugin, se dejarán estos idiomas seleccionados y se deshabilitarán los check para indicar que son obligatorios.
  - **Otros:** Define otros posibles idiomas para la aplicación. Los idiomas son inglés y francés.
  - **Idioma por defecto:** Se podrá seleccionar el idioma por defecto de la aplicación. Las opciones a seleccionar variarán según los idiomas elegidos entre los *Base* y *Otros*.
- **Módulo EJB Remoting:** Check para habilitar la creación del módulo EJB.
  - **Nombre del proyecto de EJBs:** Campo obligatorio si se ha activado la opción de Módulo EJB Remoting. Indica el nombre que se desea dar al módulo EJB.
  - **Nombre completo del proyecto de EJBs:** Campo de sólo lectura para mostrar el nombre del proyecto EJB que se va a generar en el workspace.
- Una vez rellenado los campos y pinchado en el botón *Finish* se creará la aplicación con sus proyectos relacionados y vinculados al runtime de WebLogic. Además se obtendrán las librerías necesarias para la aplicación del repositorio Maven. En la consola se mostrará una traza informativa del proceso de descarga automática de librerías para la aplicación.

Y se presenta un resumen que refleja las acciones que se han realizado.

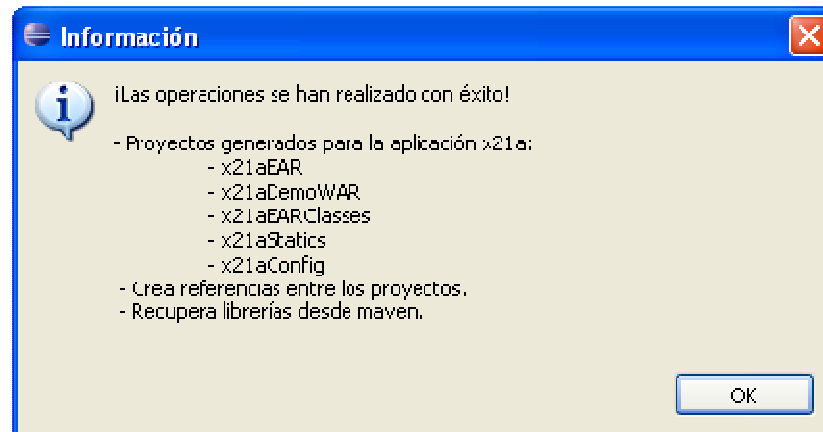


Ilustración 6. Resumen de resultados de las operaciones realizadas

- El resultado final en el workspace de eclipse es:

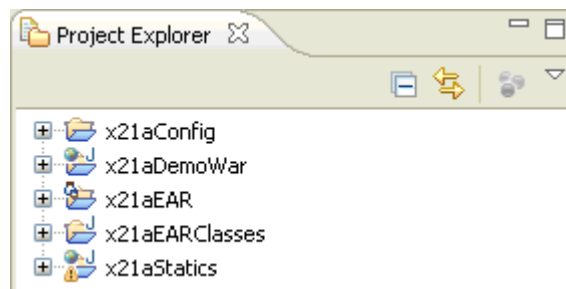


Ilustración 7. Aplicación generada en el workspace de Eclipse

En caso que falle el proceso de obtención de librerías por falta conexión u otros motivos, se podrá relanzar desde la línea de comandos.

Para ello, desde línea de comandos, en la ruta que contiene el fichero *pom.xml* del proyecto *[nombre de la aplicación]EAR* se ejecutará el siguiente comando:

```
mvn package
```

### 3.2 Añadir un proyecto WAR a una aplicación

Este asistente es el encargado de generar un nuevo proyecto WAR y vincularlo a la aplicación a través del proyecto EAR y del proyecto EARClasses generados anteriormente.

- En la categoría UDA seleccionamos la opción *4. Añadir proyecto WAR*.

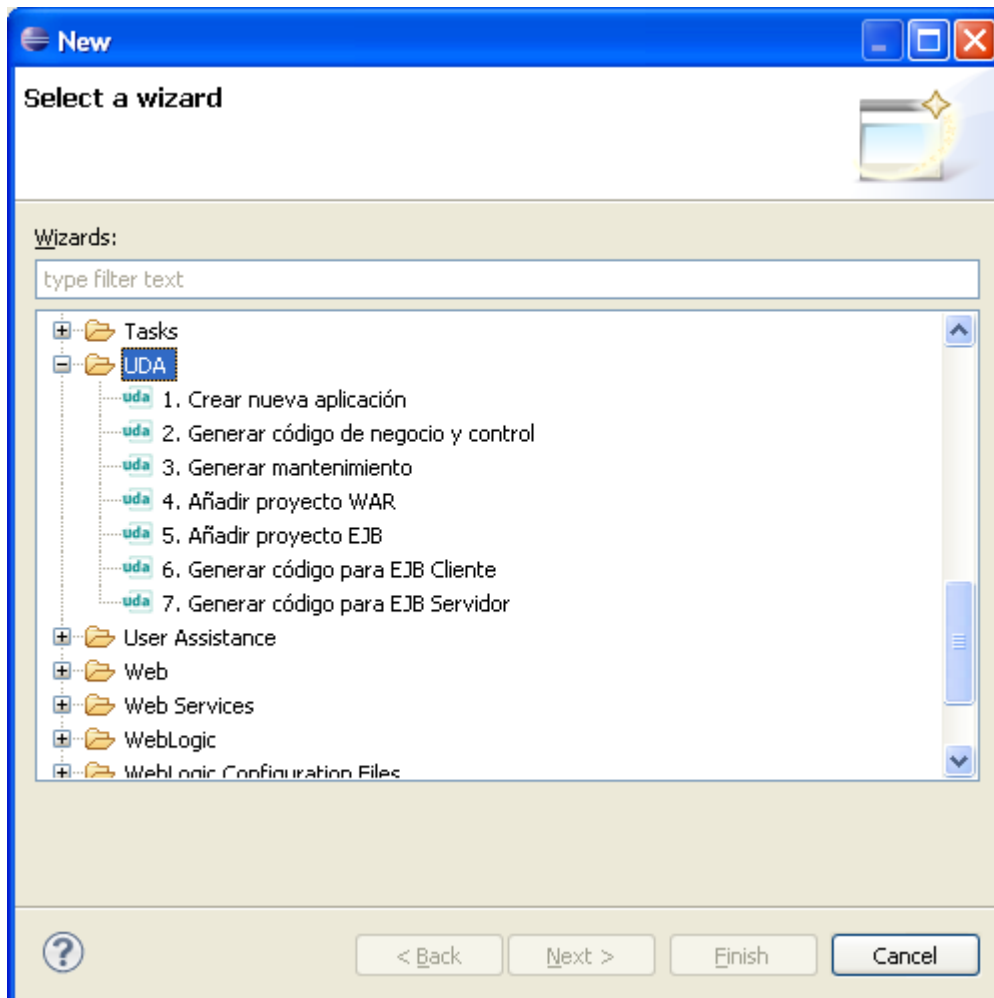


Ilustración 8. Listado de asistentes UDA – Añadir un proyecto WAR a una aplicación

- En la pantalla *Añadir un WAR a la aplicación*, rellenamos los siguientes campos de la pantalla.

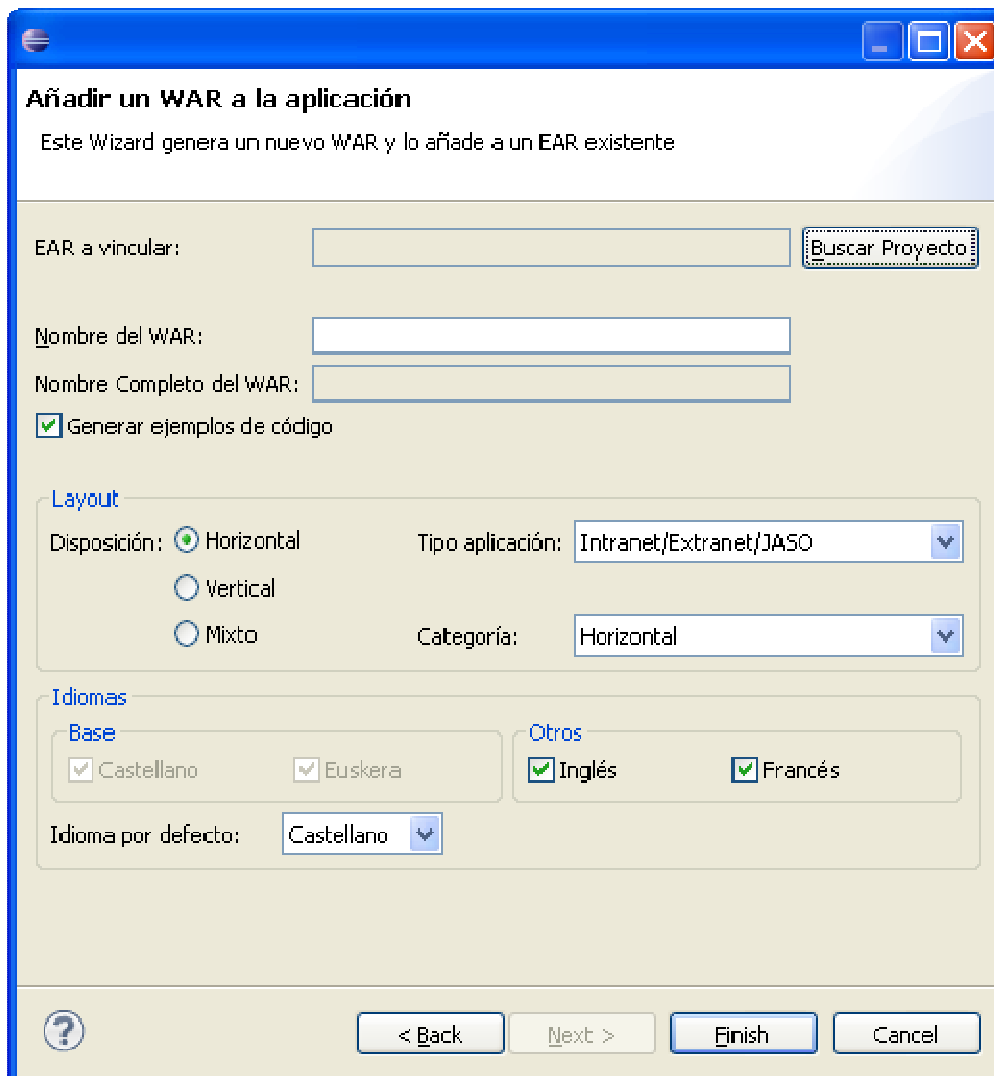


Ilustración 9. Pantalla de Añadir un proyecto WAR a una aplicación

- **EAR a vincular:** Campo obligatorio para seleccionar el proyecto EAR contenido en el workspace al que se va vincular el nuevo WAR.
- **Nombre del WAR:** Campo obligatorio del nombre del nuevo proyecto WAR. La primera letra siempre será en mayúscula y en el resto de la cadena se respetarán las mayúsculas y minúsculas puestas por el usuario (*CamelCase*).
- **Nombre Completo del WAR:** Es un campo de sólo lectura que muestra el nombre que tendrá el proyecto WAR. El nombre completo es la concatenación de [*nombre de aplicación*] + [*nombre del WAR*] + War.

Por ejemplo: *x21aDemoAddWar*, donde el nombre de la aplicación es *x21a* y *DemoAdd* es el nombre del WAR.

- **Generar ejemplos de código:** Si está seleccionado se generarán ejemplos de los patrones de presentación en la aplicación.

Para que todos los ejemplos funcionen correctamente, se aconseja ejecutar el script proporcionado de generación de la base de datos de ejemplo de UDA.

- **Layout:** En este apartado se define la disposición de pantalla que tendrá la aplicación relacionada con los logotipos en la cabecera y los menús atendiendo a ciertos criterios de organización de pantalla. Esta disposición afectará al fichero de configuración tiles.xml.
  - **Disposición:** Definimos si la disposición de la organización del menú será en horizontal, vertical o mixto.
  - **Tipo de aplicación:** Si el check de *Desarrollo para Ejje* está habilitado aparecerá esta opción donde informaremos si la aplicación es de tipo Intranet/Extranet/JASO o Internet generando una cabecera de aplicación con logotipos o una cabecera ficticia a eliminar cuando se integre en la herramienta de Gestión de Portales.
  - **Categoría:** Si el check de *Desarrollo para Ejje* está habilitado aparecerá esta opción donde informaremos si se trata de una aplicación Horizontal o Departamental en función del responsable del aplicativo apareciendo diferentes logotipos en cada caso.  
Este campo está relacionado con el campo **Tipo Aplicación**. Si la aplicación es de tipo Internet nunca podrá ser de categoría Departamental. En este caso se deshabilita el combo de Categoría y se pone su valor a Horizontal.
- **Idiomas:** Configuración de idiomas que puede tener la aplicación.
  - **Base:** Define los idiomas base, las opciones son checks de castellano y euskera. Cuando el check de *Desarrollado para Ejje* esté habilitado en la configuración del plugin, se dejarán estos idiomas seleccionados y se deshabilitarán los check para indicar que son obligatorios.
  - **Otros:** Define otros posibles idiomas para la aplicación. Los idiomas son inglés y francés.
  - **Idioma por defecto:** Se podrá seleccionar el idioma por defecto de la aplicación. Las opciones a seleccionar variarán según los idiomas elegidos entre los *Base* y *Otros*.
- Una vez rellenados los campos y pinchado en el botón *Finish* se añade el proyecto WAR al EAR indicado en la ventana. Posteriormente, se muestra un resumen donde se refleja las acciones realizadas.

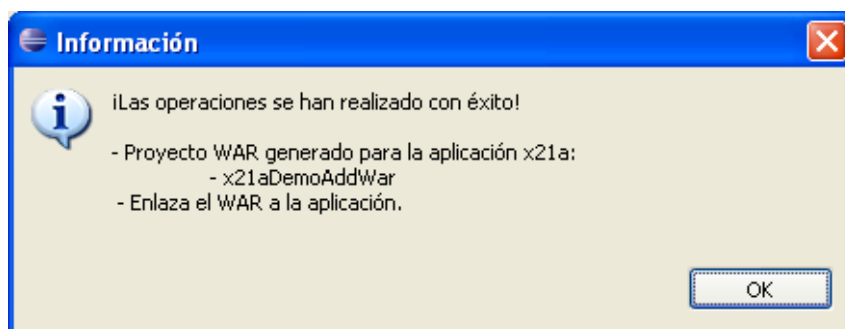


Ilustración 10. Resumen de tareas realizadas al añadir un nuevo WAR.

- El resultado en el workspace del eclipse es el siguiente:



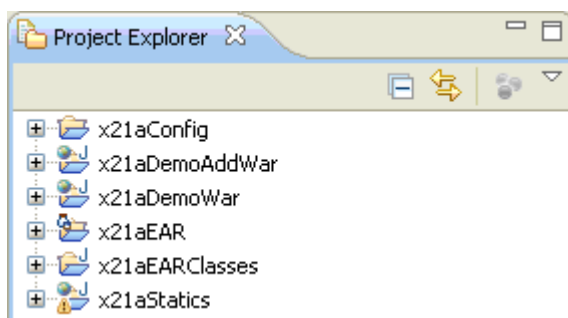


Ilustración 11. Proyecto WAR añadido a la aplicación UDA

### 3.3 Generar código para una aplicación

Este asistente es el responsable de generar el código de acceso a datos y los componentes de negocio y presentación para la aplicación desde un esquema de base de datos. Del esquema de base de datos se seleccionan las tablas y columnas para las que se va a generar el modelo, los DAOs, los servicios y los controladores. El código generado está relacionado con la tecnología de persistencia seleccionada en la creación de la aplicación UDA (asistente *Crear nueva aplicación*).

- En la categoría UDA seleccionamos la opción *Generar código de negocio y control*.

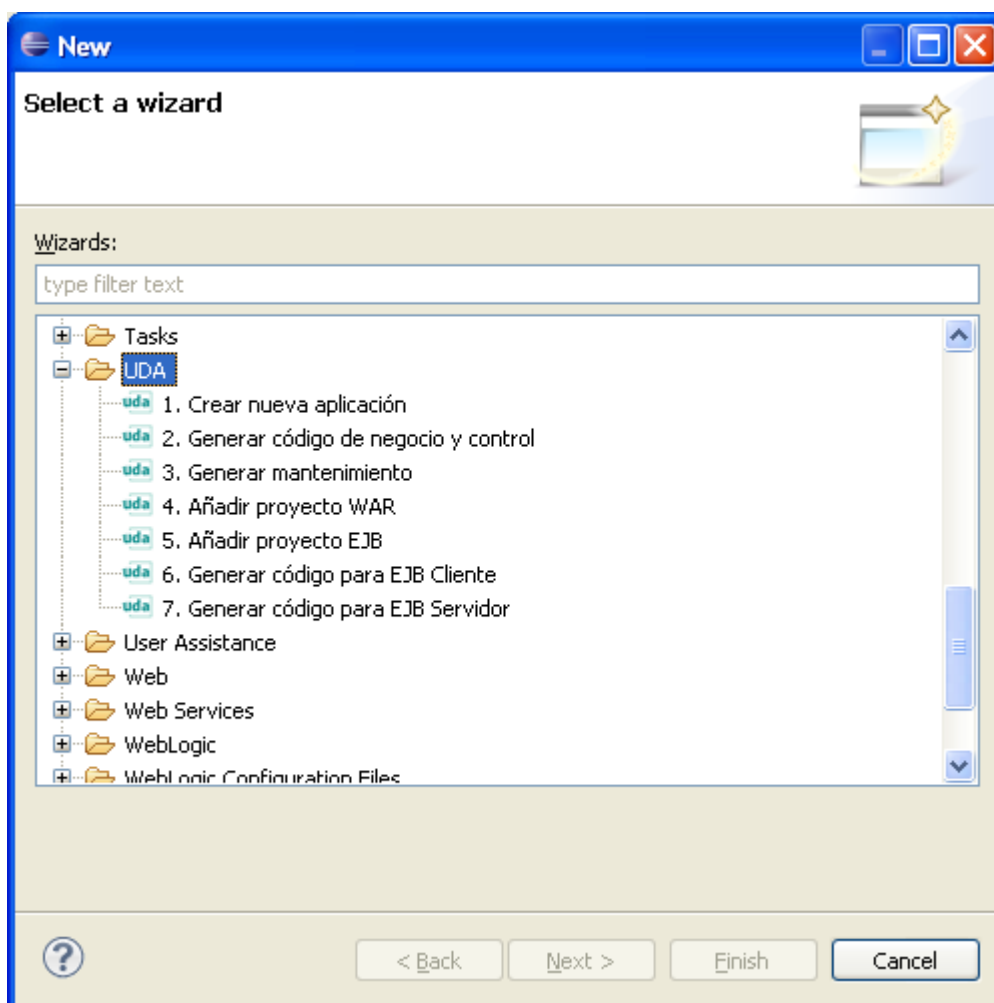


Ilustración 12. Listado de asistentes UDA – Generar código de negocio y control

- La primera pantalla nos pide los datos necesarios para conectarse a la base de datos. Una vez rellenados todos los campos podemos probar la conexión con el botón *Probar Conexión* y pasar a la siguiente pantalla.

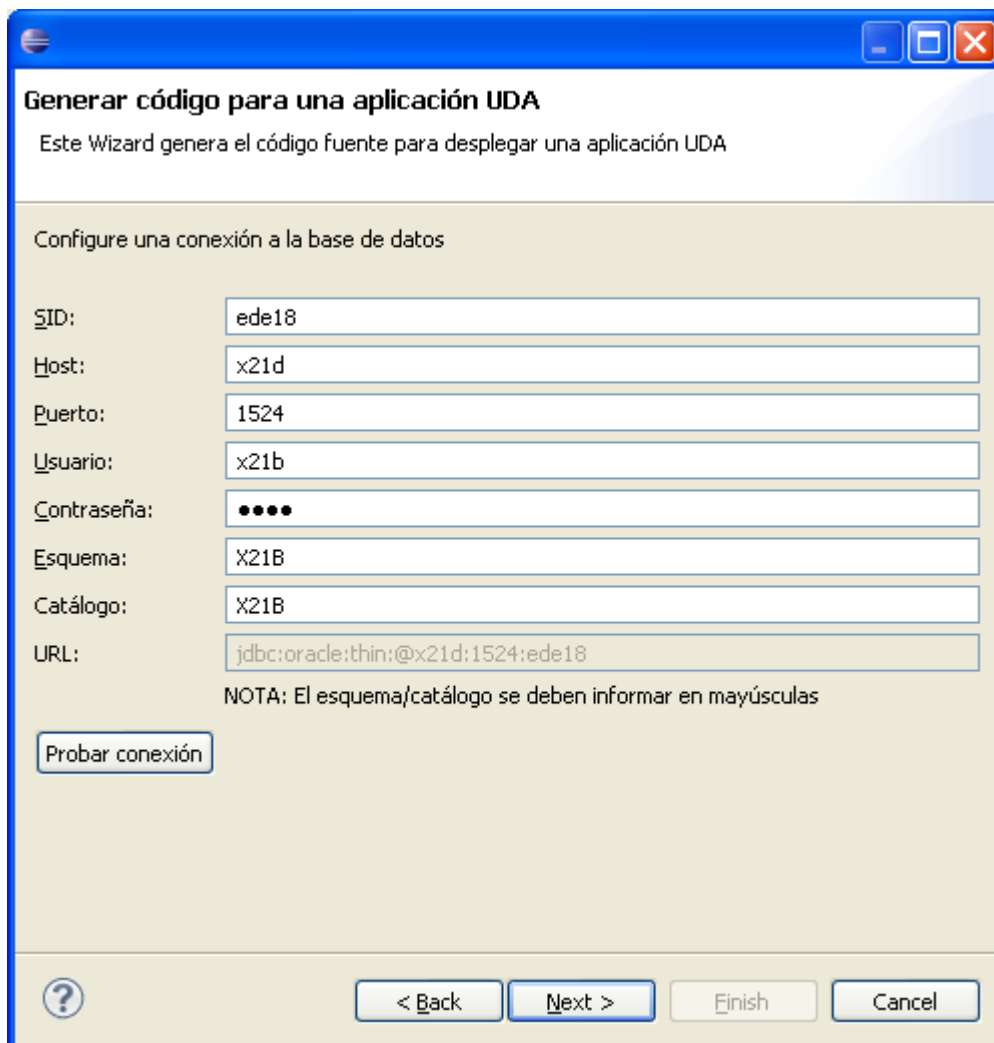


Ilustración 13. Datos de conexión a la base de datos

- **SID:** Campo obligatorio del identificador único de una base de datos.
- **Host:** Campo obligatorio del nombre o IP del ordenador de la base de datos.
- **Puerto:** Campo obligatorio del puerto de conexión a la base de datos.
- **Usuario:** Campo obligatorio del nombre de usuario de conexión a la base de datos.
- **Contraseña:** Campo obligatorio de la contraseña del usuario que se va a conectar a la base de datos.
- **Esquema:** Campo obligatorio referente al conjunto de estructuras lógicas relativas al usuario que se utilizan para obtener, almacenar o referenciar datos.
- **Catálogo:** Campo obligatorio encargado de almacenar los metadatos de cada objeto del esquema.
- **URL:** Campo de sólo lectura que muestra la cadena de conexión de la base de datos formado por el host, el puerto y el SID informados anteriormente.

- En esta ventana seleccionamos las tablas y sus respectivas columnas para las que queremos que se genere en código. En la ventana se mostrará la siguiente información:

Las tablas aparecen informadas con el nombre de la entidad a generar (si tienen sinónimo declarado se utiliza el sinónimo y sino se recogerá el nombre de la tabla).

En las columnas se visualiza si es clave primaria o compuesta y además su tipo en la base de datos y si permite valores nulos.

No está permitido seleccionar una tabla y dejar deseleccionada sus claves, vendrán seleccionadas por defecto.

Es obligatorio tener algún elemento seleccionado para pasar a la siguiente pantalla.

*NOTA: Si en la pantalla anterior, la comprobación de la conexión ha sido correcta pero en esta pantalla no aparecen las entidades puede deberse al uso de minúsculas en los datos Catálogo y Esquema de la conexión.*

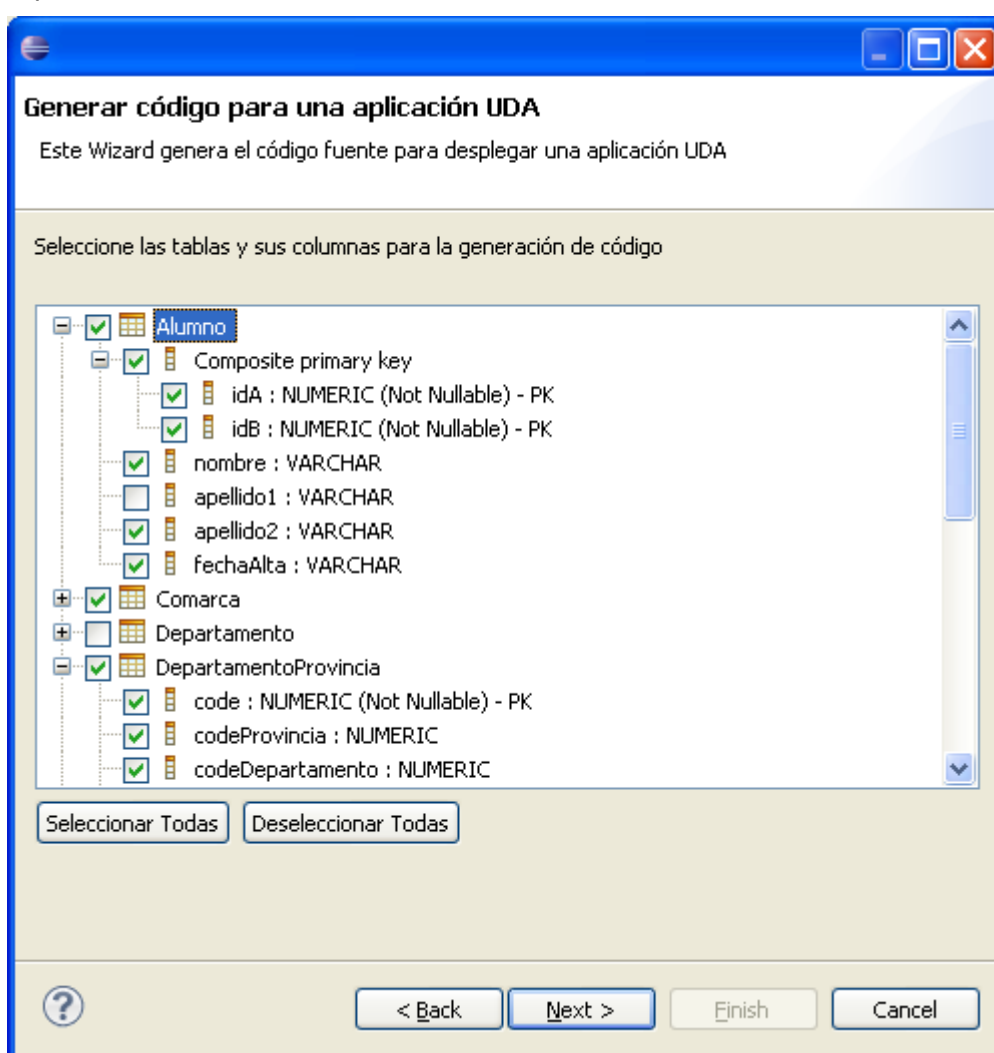


Ilustración 14. Selección de tablas y columnas

- Seleccionar Todas:** Botón encargado de seleccionar todos los checks del árbol del esquema. Por defecto, la primera vez que se enseña la ventana, todos los checks estarán seleccionados.

- **Deseleccionar Todas:** Botón encargado de deseleccionar todas las tablas y columnas del árbol de esquema.
- A continuación se seleccionarán los componentes a generar. Se recupera la información de las tablas y columnas filtradas en la ventana anterior y se genera el código correspondiente a los componentes seleccionados. Posteriormente se volcará lo generado en el proyecto seleccionado de la aplicación UDA.

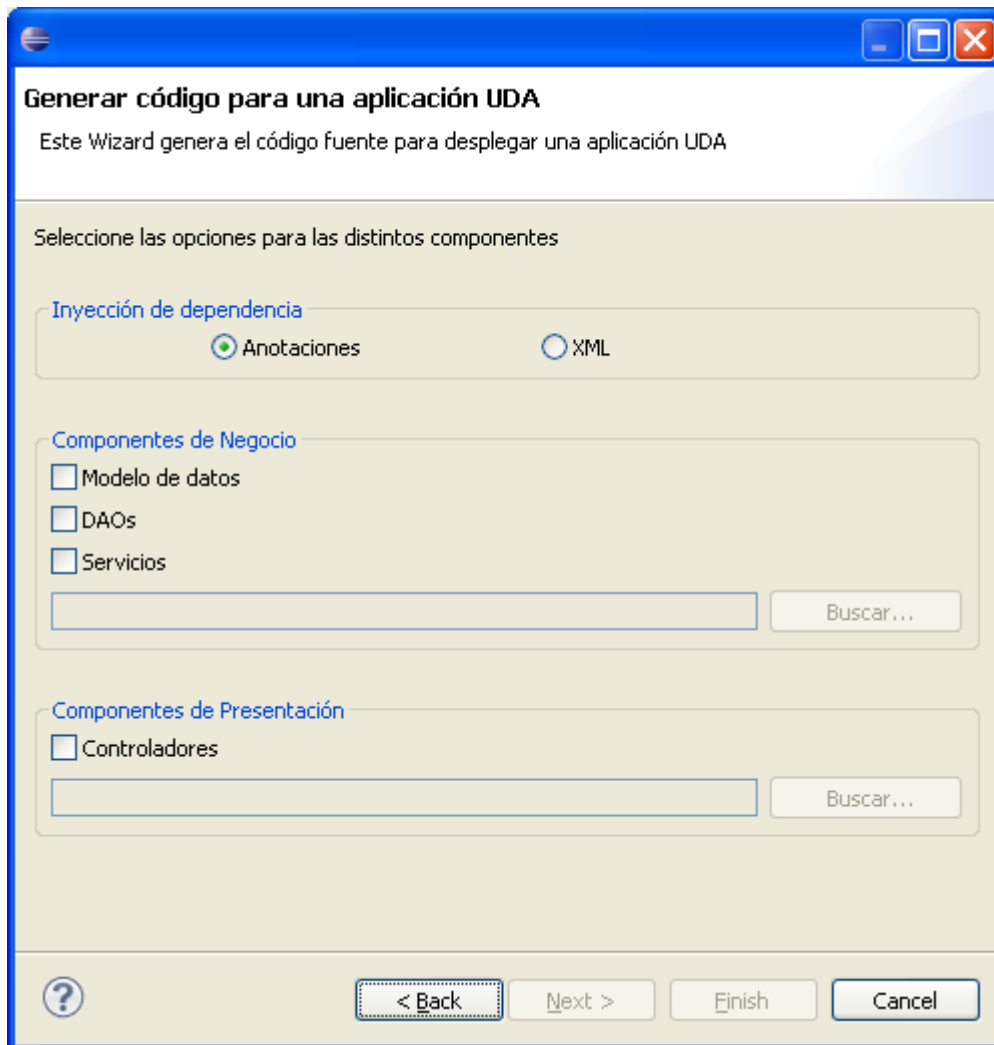


Ilustración 15. Selección de componentes a generar

- **Inyección de dependencias:** Concepto que indica como se van a resolver las dependencias entre capas.
  - **Anotaciones:** La inyección de dependencias se realiza mediante anotaciones.
  - **XML:** La inyección de dependencias se hace mediante un fichero xml.
- **Componentes de Negocio:** Concepto que agrupa tres capas de la arquitectura de las aplicaciones UDA (model, dao y service).
  - **Modelos de datos:** Check encargado de generar el Modelo de Datos correspondiente a las tablas y columnas seleccionadas anteriormente. El código se creará en el proyecto EARClasses de la aplicación según la tecnología de persistencia (Spring

JDBC o JPA 2.0), bajo la paquetería con la nomenclatura *com.ejie.[nombre de la aplicación].model*.

- **DAOs:** Check encargado de crear las clases de Acceso a Datos correspondientes a las tablas y columnas seleccionadas anteriormente. El código se genera en el proyecto EARClasses de la aplicación según la tecnología de persistencia (Spring JDBC o JPA 2.0), bajo la paquetería con la nomenclatura *com.ejie.[nombre de la aplicación].dao*.
- **Servicios:** Check encargado de generar los servicios de negocio. El código se creará en el proyecto EARClasses de la aplicación, bajo la paquetería con nomenclatura *com.ejie.[nombre de la aplicación].service*.

Una vez chequeado cualquiera de los componentes de negocio se habilita en botón *Buscar* para que se seleccione el proyecto EARClasses de la aplicación UDA.

- **Componentes de Presentación:** Concepto que hace referencia a la capa de presentación de la aplicación.

- **Controladores:** Check encargado de generar los controladores correspondientes a las tablas y columnas seleccionadas anteriormente. El código se creará en el proyecto de WAR de la aplicación según la tecnología de persistencia (Spring JDBC o JPA 2.0), bajo la paquetería con la nomenclatura *com.ejie.[nombre de la aplicación].control*.

Si el check de Controladores está marcado, se habilitará el botón *Buscar* para que se seleccione el proyecto de WAR de la aplicación UDA.

- La siguiente ventana aparecerá dependiendo si está seleccionado el check de *Desarrollo para EJIE* en la configuración de preferencias del plugin UDA, en caso contrario no se mostrará. Ver punto 3.1. Ilustración 2.

Esta ventana facilita la configuración de una parte de la seguridad de la aplicación.

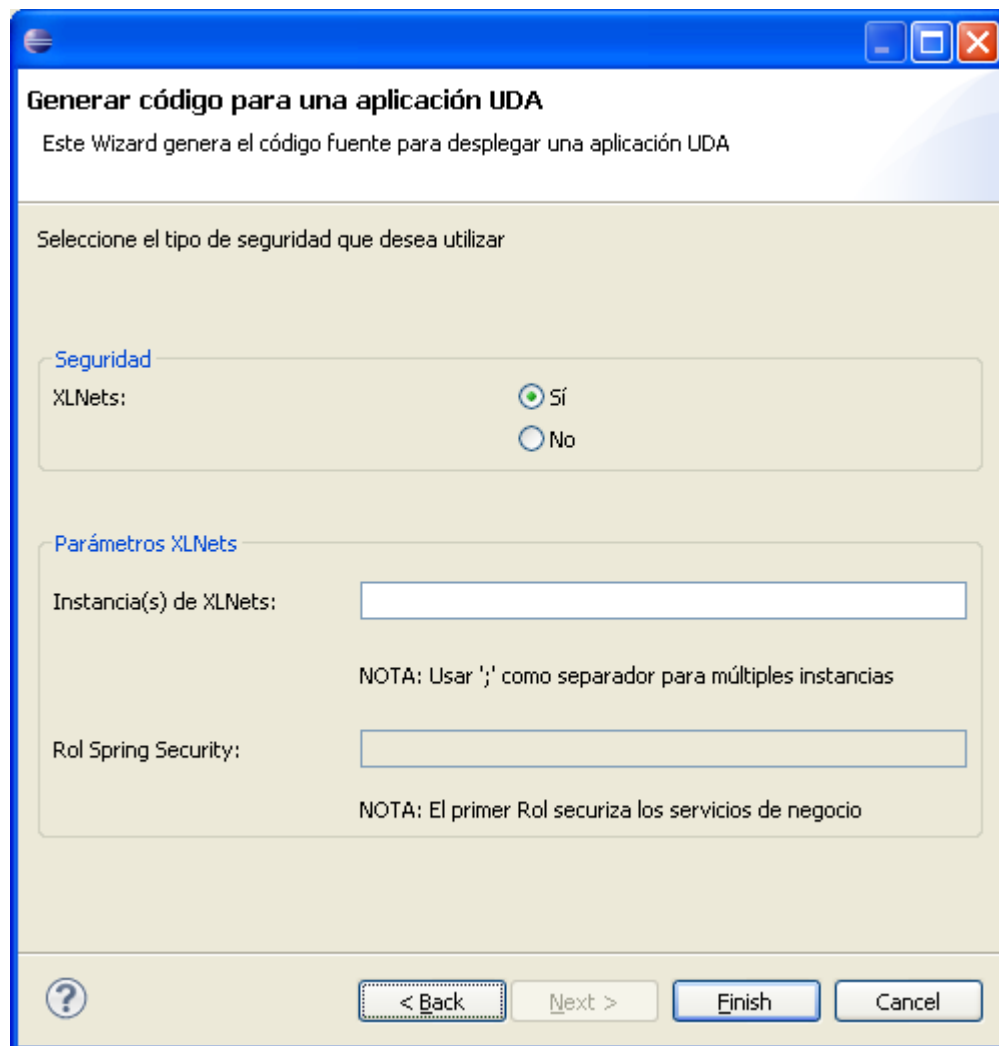


Ilustración 16. Configuración de la seguridad

- **Seguridad:** Si se desea aplicar XLNets, sistema de seguridad de EJIE/GV.
  - **XLNets:** Marcando *Sí*, se mostrarán los campos bajo *Parámetros XLNets*.
- **Parámetros XLNets:** Parámetros a especificar para vincular el sistema XLNets con la seguridad de las aplicaciones UDA (basada en Spring Security).
  - **Instancia XLNets:** Nombre de la instancia o instancias que la aplicación tendrá configuradas en XLNets. En caso de que haya varias, las instancias se separarán por el carácter ';'. Estas instancias se aplicarán al service-config.xml del proyecto WAR. En lo referente a la seguridad de la capa de servicios (service-config.xml existente en el proyecto EARClasses), únicamente se aplicará la seguridad en los servicios con la primera instancia informada.
  - **Rol Spring Security:** Nombre del rol de Spring Security que utilizará la aplicación como correspondencia con la instancia XLNets.
- Al presionar el botón *Finish*, antes de empezar la generación del código, se advierte al usuario que se sobrescribirá el código que se hubiera generado anteriormente con el mismo nombre y en la misma paquetería.

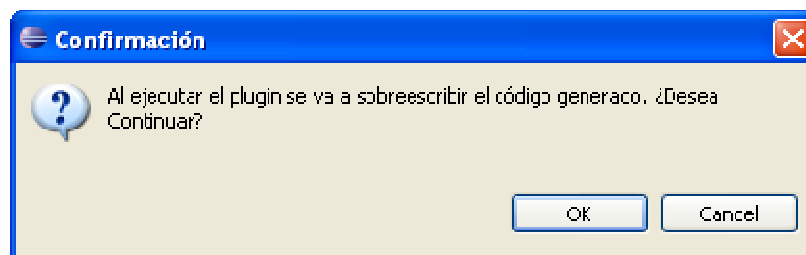


Ilustración 17. Confirmación de sobreescritura de código

- Una vez aceptada la confirmación, UDA genera el código con las opciones seleccionadas en los correspondientes proyectos de la aplicación. Al finalizar, se visualiza la pantalla de resumen con las tareas realizadas.

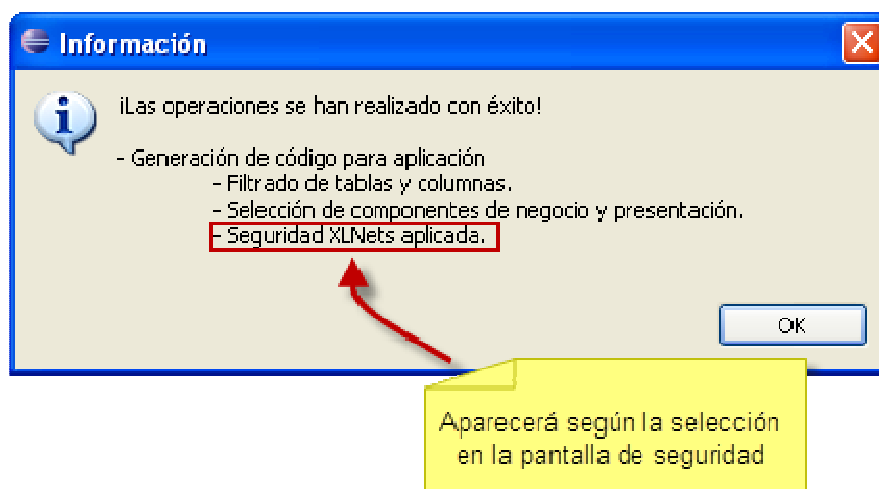


Ilustración 18. Resumen de Generación de código de negocio y control.

### 3.4 Crear nuevo mantenimiento

Este asistente permite generar un nuevo mantenimiento para la aplicación. El patrón mantenimiento se crea para facilitar la lógica necesaria en las operaciones básicas sobre una tabla, denominadas CRUD (create, read, update y delete) y el comportamiento de la página.

- En la categoría UDA seleccionamos la opción *Generar mantenimiento*.



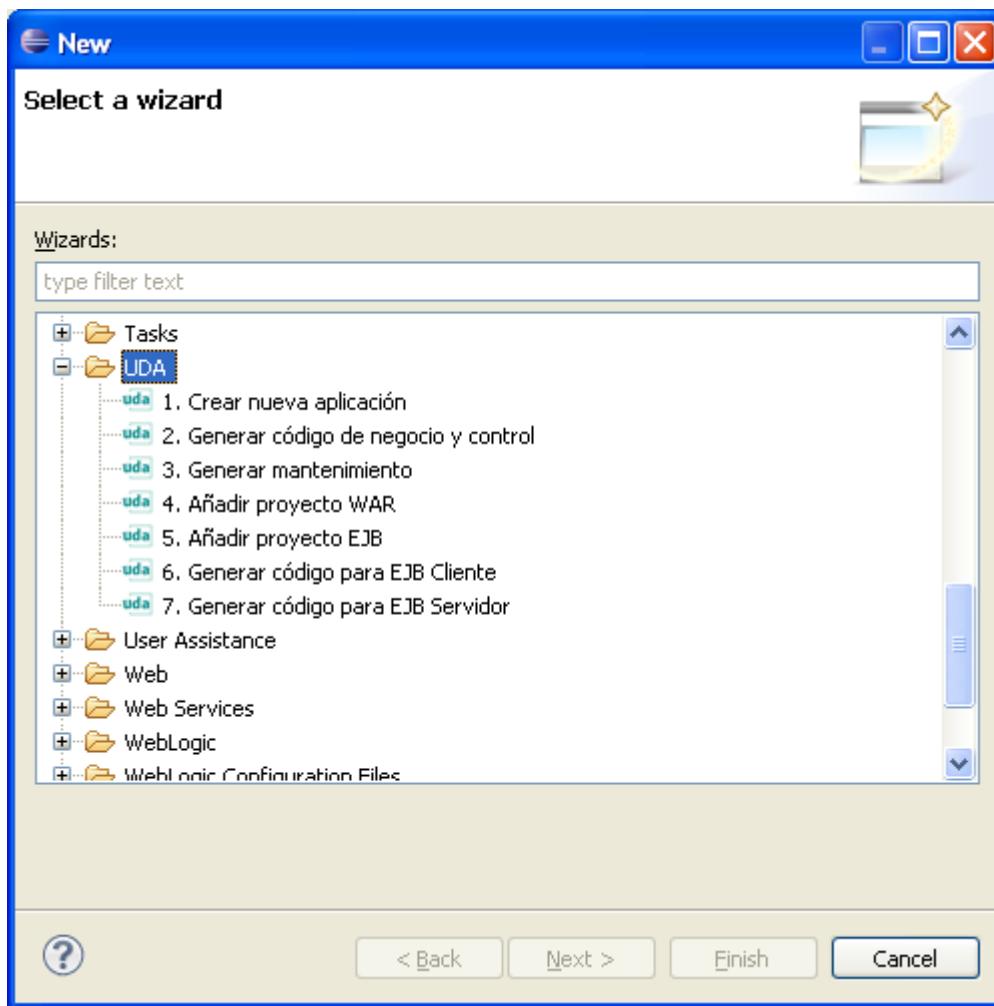
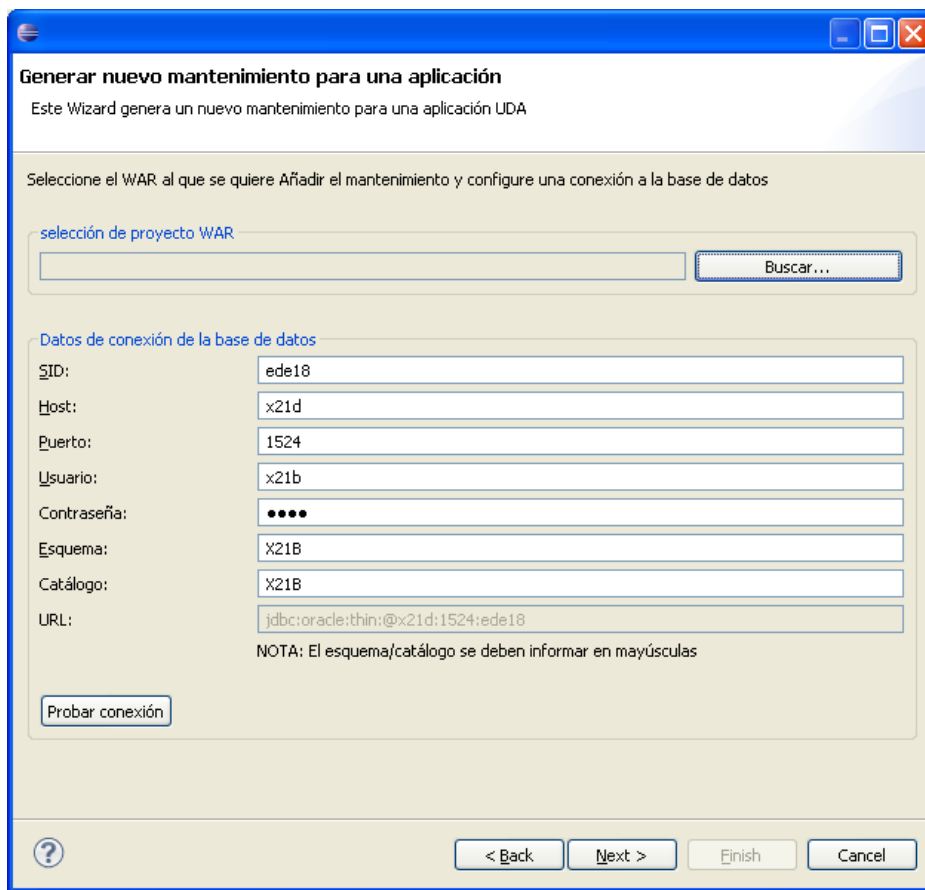


Ilustración 19. Listado de asistentes UDA – Generar mantenimiento

- La primera pantalla pide el proyecto WAR donde se va a crear el mantenimiento. Además se solicitan los datos necesarios para la conexión a la base de datos con la que se podrá recuperar la entidad (tabla del esquema) sobre la que se realizará el mantenimiento.

Una vez rellenados los campos relativos a la base de datos podemos probar la conexión con el botón *Probar Conexión* y pasar a la siguiente pantalla.



**Generar nuevo mantenimiento para una aplicación**  
Este Wizard genera un nuevo mantenimiento para una aplicación UDA

Seleccione el WAR al que se quiere Añadir el mantenimiento y configure una conexión a la base de datos

selección de proyecto WAR

Buscar...

Datos de conexión de la base de datos

SID: ede18

Host: x21d

Puerto: 1524

Usuario: x21b

Contraseña: ••••

Esquema: X21B

Catálogo: X21B

URL: jdbc:oracle:thin:@x21d:1524:ede18

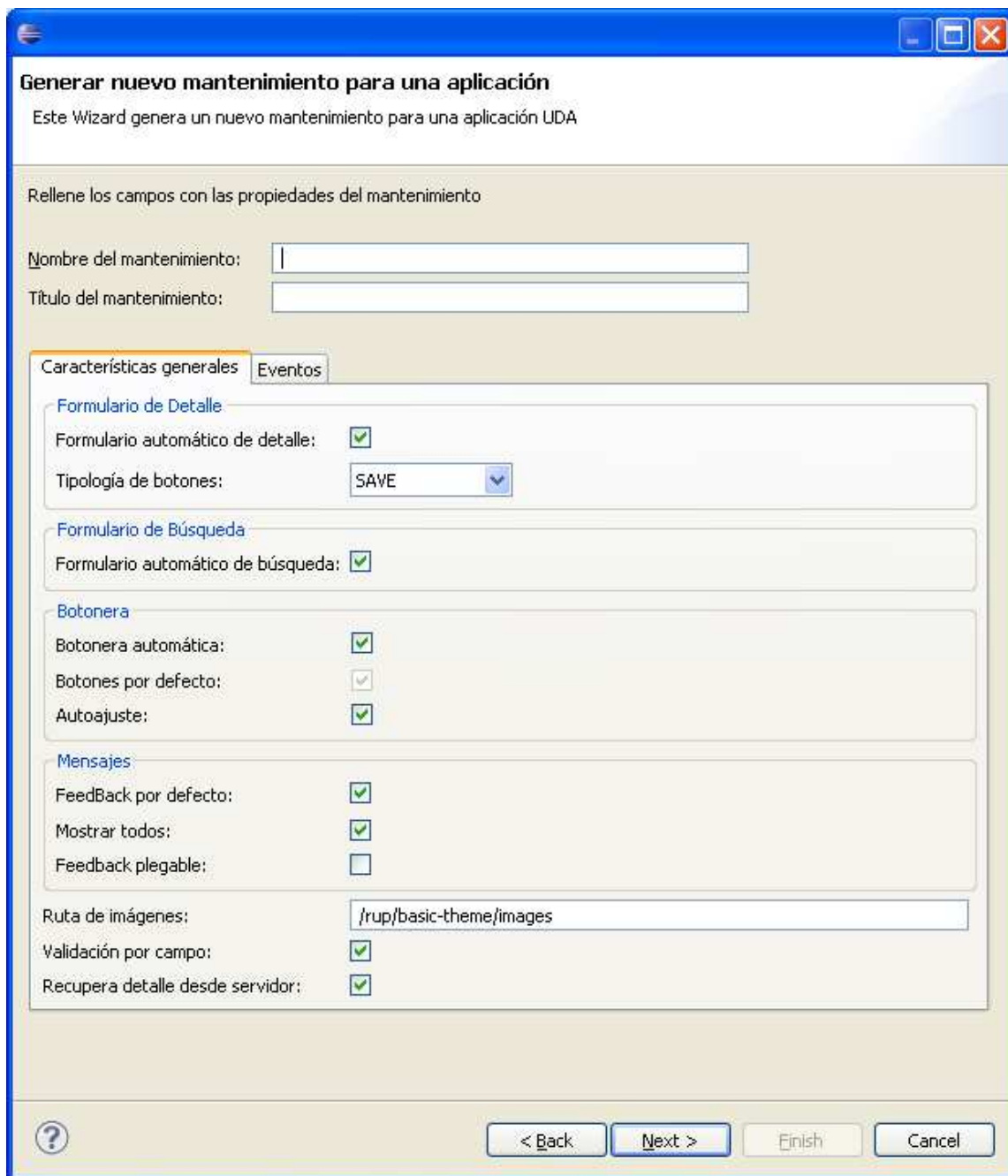
NOTA: El esquema/catálogo se deben informar en mayúsculas

Probar conexión

< Back Next > Finish Cancel

Ilustración 20. Datos de conexión a la base de datos

- **Selección de proyecto WAR:** campo obligatorio, se selecciona un WAR de tipo UDA contenido en el workspace.
  - **SID:** Campo obligatorio del identificador único de una base de datos.
  - **Host:** Campo obligatorio del nombre o IP del ordenador de la base de datos.
  - **Puerto:** Campo obligatorio del puerto de conexión a la base de datos.
  - **Usuario:** Campo obligatorio del nombre de usuario de conexión a la base de datos.
  - **Contraseña:** Campo obligatorio de la contraseña del usuario que se va a conectar a la base de datos.
  - **Esquema:** Campo obligatorio referente al conjunto de estructuras lógicas relativas al usuario que se utilizan para obtener, almacenar o referenciar datos.
  - **Catálogo:** Campo obligatorio encargado de almacenar los metadatos de cada objeto del esquema.
  - **URL:** Campo de sólo lectura que muestra la cadena de conexión de la base de datos formado por el host, el puerto y el SID informados anteriormente.
- En esta pantalla se configuran las características generales (propiedades) del mantenimiento y sus eventos principales. Se dividen en dos pestañas, *Características generales* y *Eventos*.



**Generar nuevo mantenimiento para una aplicación**  
Este Wizard genera un nuevo mantenimiento para una aplicación UDA

Rellene los campos con las propiedades del mantenimiento

Nombre del mantenimiento:

Título del mantenimiento:

Características generales **Eventos**

**Formulario de Detalle**

Formulario automático de detalle: ☒

Tipología de botones:

**Formulario de Búsqueda**

Formulario automático de búsqueda: ☒

**Botonera**

Botonera automática: ☒

Botones por defecto: ☒

Autoajuste: ☒

**Mensajes**

FeedBack por defecto: ☒

Mostrar todos: ☒

Feedback plegable: ☐

Ruta de imágenes:

Validación por campo: ☒

Recupera detalle desde servidor: ☒

Ilustración 21. Propiedades del mantenimiento – Pestaña de Características generales

- **Nombre del mantenimiento:** Campo obligatorio del nombre al mantenimiento.
- **Título del mantenimiento:** Campo obligatorio con el título que se mostrará en la pantalla del mantenimiento.

Pestaña *Características generales*: En esta pestaña están las propiedades del mantenimiento.

- **Formulario de Detalle:** Propiedades para la definición del formulario de detalle.
  - **Formulario automático de detalle:** Por defecto el check de este campo está activado indicando que se va a generar el formulario de detalle de forma automática sin que el

desarrollador tenga que dibujar los controles en la JSP haciendo uso de las propiedades establecidas en la configuración del grid.

En el caso que el desarrollador quiera definir el formulario, se desmarcará el check y aparecerá un campo para informar el nombre del id del formulario que se quiera invocar en el mantenimiento.

- **Tipología de botones:** Propiedad para seleccionar los botones del formulario de detalle.
  - **SAVE:** Se pintarán dos botones, el de *Guardar* (realiza la acción de guardar y cierra la ventana de diálogo del detalle) y el de *Cancelar* (que cierra el diálogo de detalle siempre y cuando no se hayan realizado cambios en el formulario).
  - **SAVE\_REPEAT:** Se crearán tres botones, el de *Guardar* (realiza la acción de guardar y cierra la ventana de diálogo del detalle), el de *Guardar y Repetir* (que realiza la acción de guardar y vuelve a mostrar el diálogo en blanco para una nueva acción de guardar) y el de *Cancelar* (que cierra el diálogo de detalle siempre y cuando no se hayan realizado cambios en el formulario).

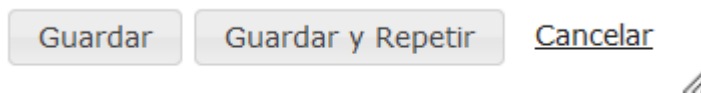


Ilustración 22. Tipología de botones

- **Formulario Búsqueda:** Propiedades sobre el formulario de búsqueda del mantenimiento.
  - **Formulario de búsqueda:** Si este check está activo, se generará el formulario de búsqueda de forma automática sin que el desarrollador tenga que dibujar los controles en la JSP haciendo uso de las propiedades establecidas en la configuración del grid. Además se crearán los botones de acción (*Buscar y Limpiar*) así como la búsqueda enviando los valores del formulario al servidor para realizar la carga del grid con los resultados obtenidos.

En el caso que el desarrollador quiera definir el formulario, se desmarcará el check y aparecerá un campo para informar el nombre del id del formulario que se quiera invocar en el mantenimiento.

Criterios de búsqueda:

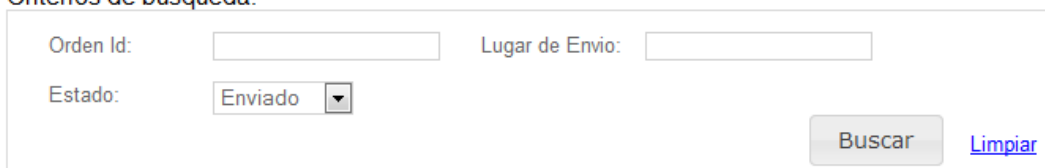


Ilustración 23. Ejemplo de formulario de búsqueda

- **Botonera:** Botonera del mantenimiento.
  - **Botonera automática:** Por defecto el check de este campo está activado, de manera que se genera una botonera de manera automática.  
En caso contrario, el usuario deberá deshabilitar el check e informar en el campo que se activa el identificador de la botonera desarrollada.

- **Botones por defecto:** Está vinculado al campo *Botonera automática*. Indica si se van a crear los botones por defecto en la botonera (*Nuevo, Editar, Borrar y Filtrar*), bien se trate de la botonera creada manualmente por el desarrollador o bien la generada automáticamente por el mantenimiento.
- **Autoajuste:** Indica si se va a ajustar el tamaño de la botonera al tamaño del grid.



Ilustración 24. Botones por defecto de la botonera

- **Mensajes:** Formulario de mensajes.
  - **Feedback por defecto:** Este campo está activado por defecto, de manera que se generará el feedback (área de información al usuario de la aplicación) de forma automática.
  - **Mostrar todos:** Por defecto estará activado indicando que se mostrarán todas las advertencias. En caso contrario, no se mostrarán los mensajes correspondientes a las acciones de añadir, modificar o borrar que se hayan realizado con éxito.
  - **Feedback plegable:** Por defecto está desactivado. Sirve para que la zona de feedback se pueda reaprovechar mientras no se muestra la información al usuario moviendo dinámicamente el contenido de la página.
- **Ruta de imágenes:** Ruta donde están las imágenes del grid. Por defecto las imágenes estarán en la ruta `"/rup/basic-theme/images"` del proyecto de *[nombre aplicación]Statics*.
- **Validación por campo:** Está activado por defecto indicando que la validación de los campos del formulario se realizará de forma individual (a la hora de perder el foco). En caso contrario se realizará por formulario (se validarán todos los campos antes de realizar el envío de datos).
- **Recupera detalle desde servidor:** Por defecto estará activado indicando que el mantenimiento accederá al servidor para obtener los datos a mostrar en el detalle.

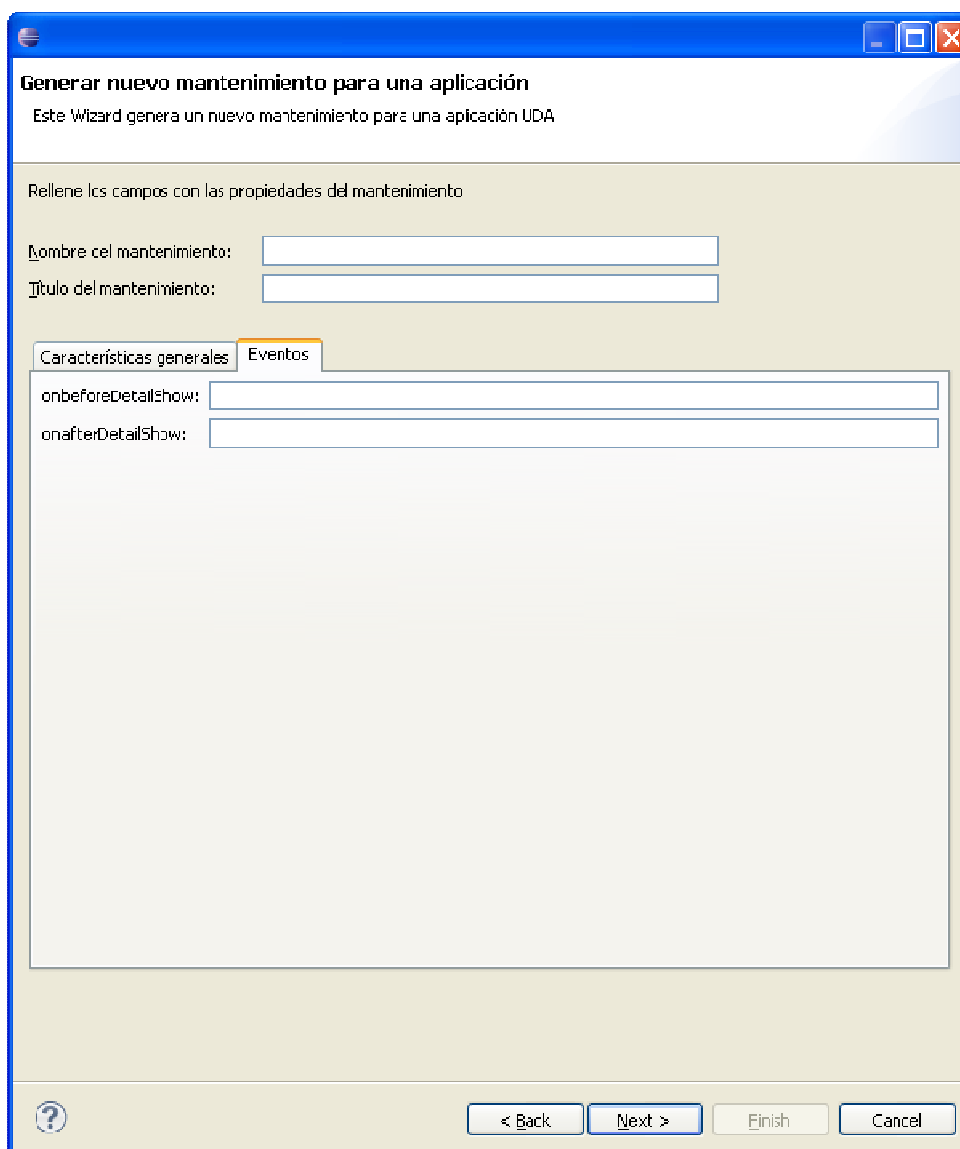


Ilustración 25. Eventos del mantenimiento

**Pestaña *Eventos*:** en esta pestaña se detallan los eventos del mantenimiento, mediante los que se podrán extender sus funcionalidades para adecuarlo a las necesidades particulares de cada aplicación. Se rellenarán los campos con el nombre de la función que se ejecutará a la hora de ejecutar el evento.

- **onbeforeDetailShow:** Nombre de la función que se ejecuta antes de enseñar el formulario de edición.
  - **onafterDetailShow:** Nombre de la función que se ejecuta después de mostrar el formulario de edición y que el patrón haya realizado todas sus acciones.
- En la siguiente ventana se seleccionará la entidad a mantener y las características (propiedades y eventos) de la tabla en la que se mostrarán los datos.

*NOTA: Si en la pantalla inicial, la comprobación de la conexión ha sido correcta pero en esta pantalla no aparecen las entidades puede deberse al uso de minúsculas en los datos Catálogo y Esquema de la conexión.*

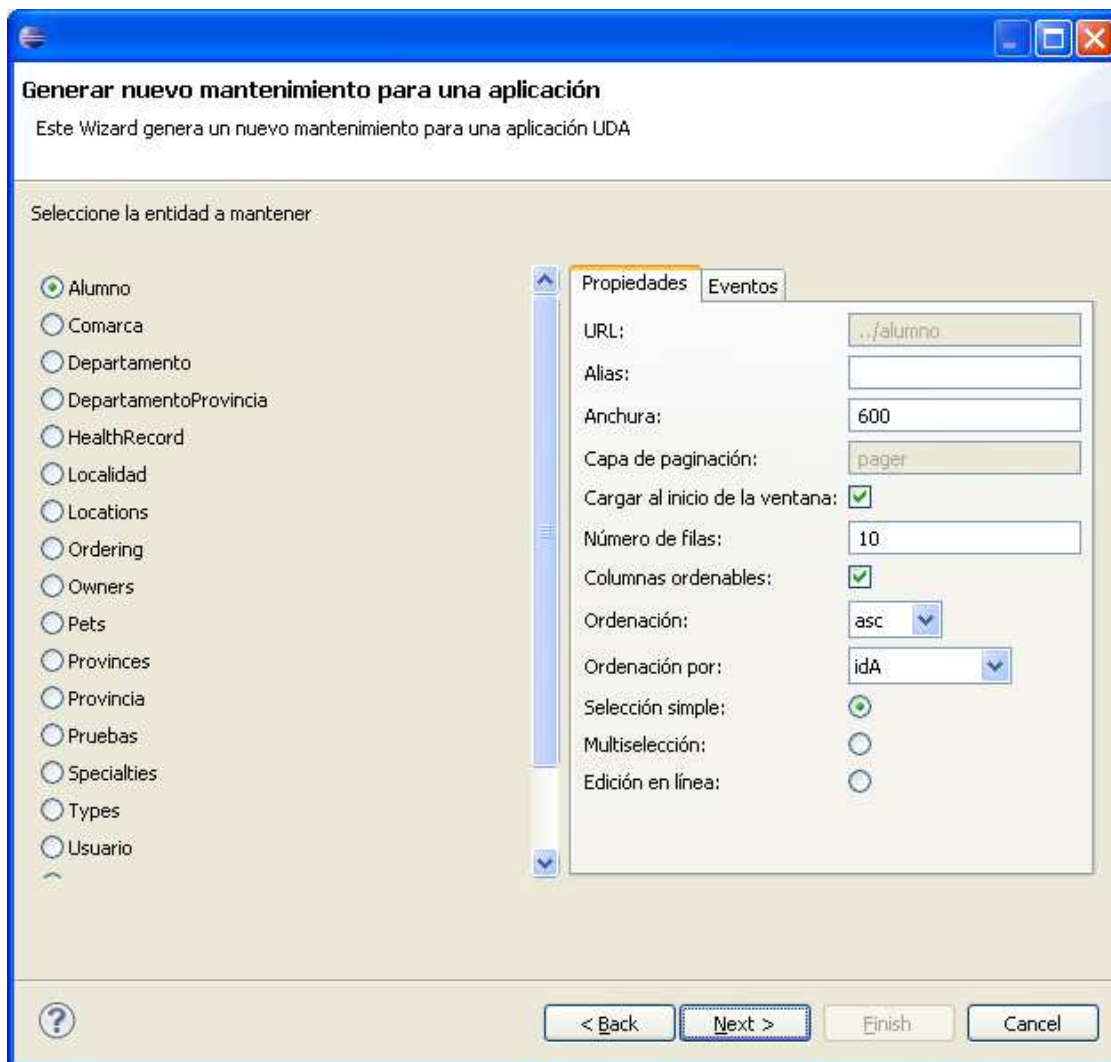
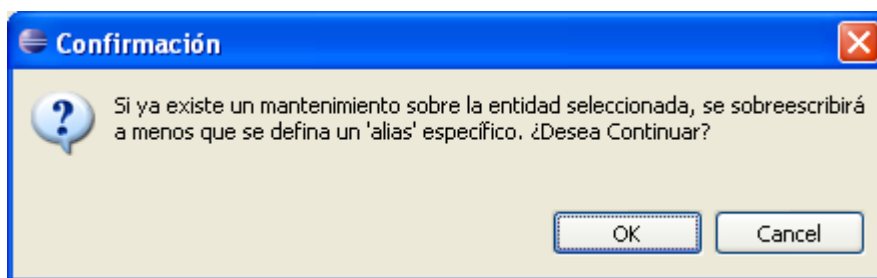


Ilustración 26. Listado de entidades y propiedades del grid.

- **Seleccione la entidad a mantener:** Campo obligatorio para elegir la entidad a mantener correspondiéndose ésta con las tablas existentes en base de datos.

**Pestaña *Propiedades*:** En esta pestaña se detallan las propiedades de la tabla o grid que mostrará los datos de la entidad.

- **URL:** Campo de sólo lectura que define la URL a través de la cual se cargará el grid. Se cambiará automáticamente según la entidad seleccionada en el listado.
- **Alias:** Campo que permite generar un nombre alternativo para el mantenimiento de la entidad seleccionada. Sirve para realizar dos mantenimientos de la misma entidad sin perder los cambios de código realizados en el primer mantenimiento de la entidad. En caso de no indicarse aparecerá el siguiente mensaje para avisar de la sobrescritura de código:



- **Anchura:** Define la anchura del grid.
- **Capa de paginación:** Indica el elemento HTML que contiene la paginación de los resultados del grid. Por defecto este campo tendrá el valor *pager*.
- **Cargar al inicio de la ventana:** Este campo aparecerá activo indicando que el grid se cargará con los datos en la petición de la página; en caso contrario, se deberá invocar al método *reloadGrid* para realizar la carga.
- **Número de filas:** Define el número de elementos en cada página del grid.
- **Columnas ordenables:** Se mostrará activado indicando que las columnas del grid se pueden reordenar entre sí, en caso contrario las columnas no se podrán mover.
- **Ordenación:** Indica el criterio de ordenación de la columnas (ascendente o descendente) en al cargar el grid.
- **Ordenación por:** Permite seleccionar el nombre de la columna de la entidad por el que ordenar la primera vez que se carga el grid.
- **Selección simple:** Indica que se trabajará de forma individual con las filas de la tabla.
- **Multiselección:** Permitirá que el grid pueda gestionar múltiples filas de forma simultánea.
- **Editable:** Indica si las filas se pueden editar en la propia tabla.

Pestaña *Eventos*: En esta pestaña se detallan los eventos del grid. Se informarán los campos con el nombre de la función que se ejecuta en cada evento que gestiona el mantenimiento.

- **beforeRequest:** Nombre de la función que se ejecuta antes de solicitar una petición.
- **loadBeforeSend:** Hace un pre-callback para modificar el objeto XMLHttpRequest antes de ser enviado.
- **gridComplete:** Nombre de la función que se ejecuta después de que todos los datos están cargados en el grid y todos los demás procesos se hayan completado.
- **loadComplete:** Nombre de la función que se ejecuta inmediatamente después de cada petición al servidor.
- **ondblclickRow:** Nombre de la función que se ejecuta inmediatamente después de un doble click sobre una fila del grid.
- **onSelectRow:** Nombre de la función que se ejecuta inmediatamente después de un click sobre una fila del grid.
- **onSelectAll:** Nombre de la función que se ejecuta cuando el grid es de múltiple selección y se selecciona el check de selección múltiple del grid.



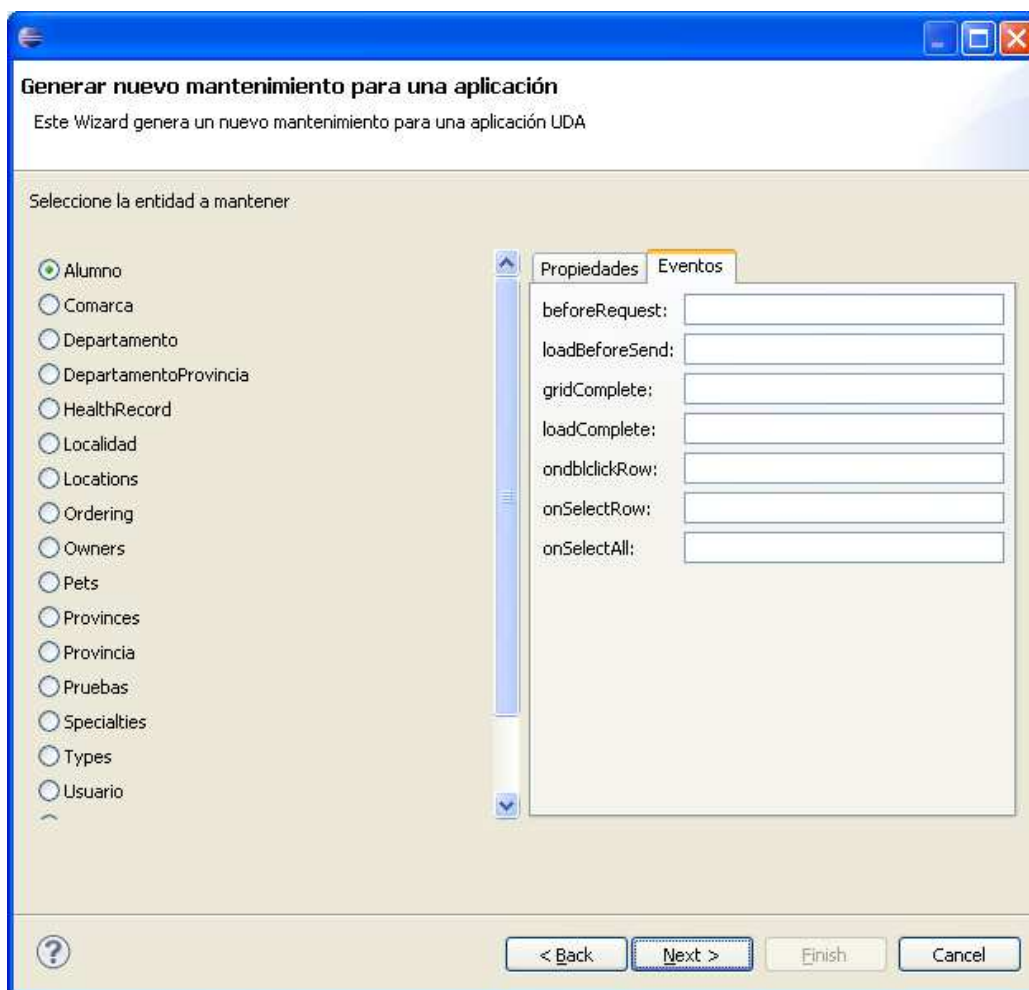


Ilustración 27. Listado de entidades y eventos del grid

- En la siguiente ventana se podrá personalizar las columnas a mostrar en el grid. Al inicio de la ventana aparecerán las columnas de la entidad seleccionada y todas estarán chequeadas.

Las columnas *chequeadas* serán las que se exporten a la configuración de la tabla para que puedan ser gestionadas en la aplicación. Para configurar cada columna debemos seleccionarla y automáticamente se mostrará la capa de propiedades cargada con los valores por defecto para permitir la parametrización.

Las claves de la entidad son obligatorias para trabajar con el grid por lo que siempre se exportarán a la configuración del mantenimiento. Pero en el caso que se quiera ocultar en la tabla, se activará su propiedad *Hidden* que se encuentra dentro de las *Propiedades avanzadas*.

Por otro lado, si se quiere que una columna aparezca en el formulario de detalle para permitir la modificación de sus valores, entonces se activará su propiedad *Editable*.

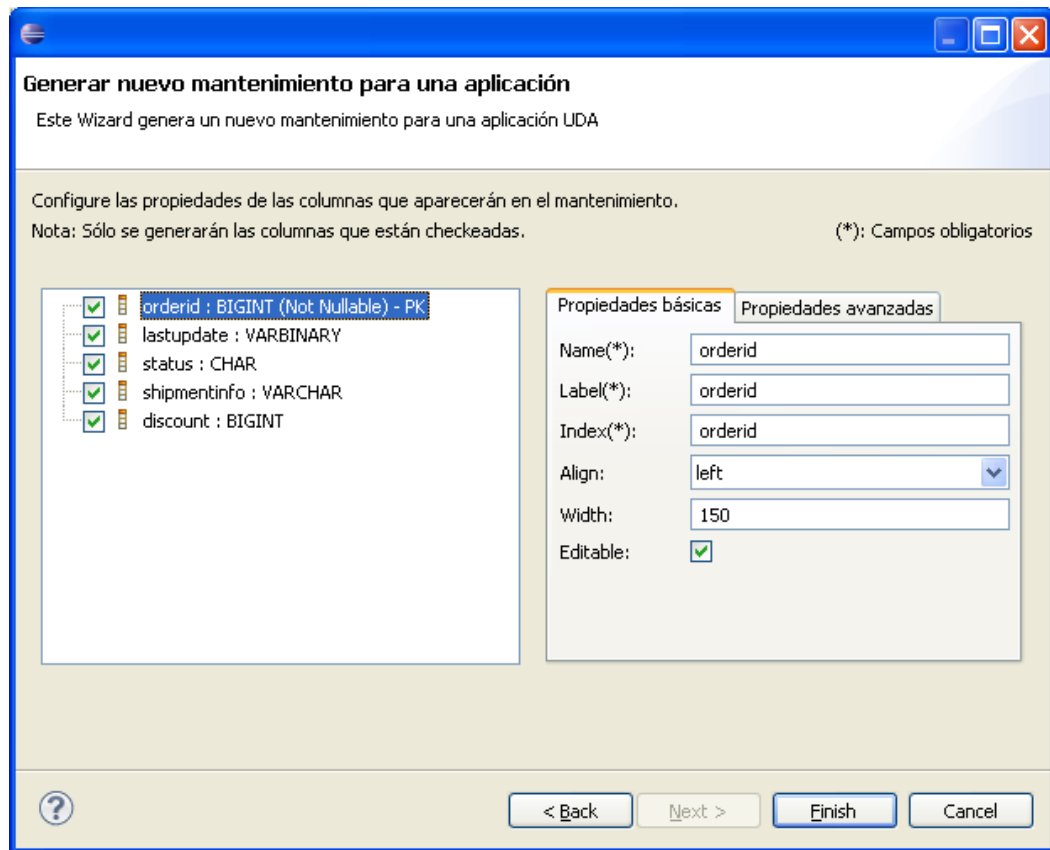
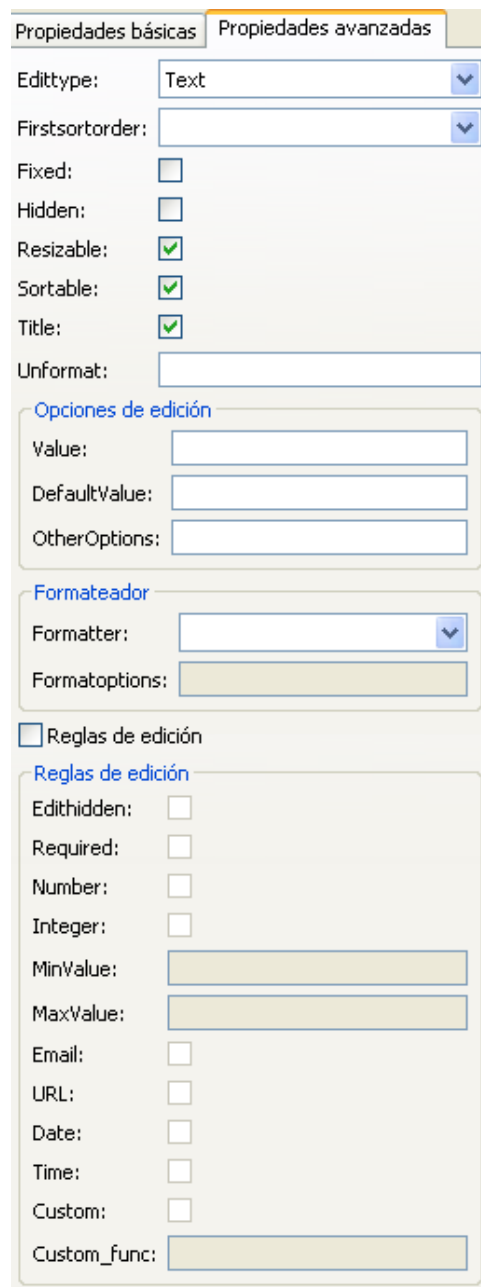


Ilustración 28. Checks de columnas y sus propiedades.

Las propiedades de *Name*, *Label* e *Index* serán obligatorias para las columnas chequeadas.

- **Name:** Campo obligatorio que indica nombre interno de la columna.
  - **Label:** Campo obligatorio para informar el texto que figurará en la cabecera de la columna.
  - **Index:** Campo obligatorio para indicar el índice de la columna.
  - **Align:** Establece la alineación de los elementos en la columna.
  - **Width:** Define el ancho de la columna.
  - **Editable:** Por defecto estará activado indicando que la columna es editable.
- *Propiedades avanzadas* de las columnas:



The screenshot shows the 'Propiedades avanzadas' tab of a configuration window. It contains several sections:

- Basic Properties:**
  - Edittype: Text (dropdown)
  - Firstsortorder: (dropdown)
  - Fixed: ☐
  - Hidden: ☐
  - Resizable: ☒
  - Sortable: ☒
  - Title: ☒
  - Unformat: (text input)
- Opciones de edición (Editing Options):**
  - Value: (text input)
  - DefaultValue: (text input)
  - OtherOptions: (text input)
- Formateador (Formatter):**
  - Formatter: (dropdown)
  - Formatoptions: (text input)
- Reglas de edición (Editing Rules):**
  - ☐ Reglas de edición
  - Reglas de edición (sub-section):**
    - Edithidden: ☐
    - Required: ☐
    - Number: ☐
    - Integer: ☐
    - MinValue: (text input)
    - MaxValue: (text input)
    - Email: ☐
    - URL: ☐
    - Date: ☐
    - Time: ☐
    - Custom: ☐
    - Custom\_func: (text input)

Ilustración 29. Propiedades avanzadas de la columna.

- **Edittype:** Indica el tipo de elemento que se forma al poner la columna en modo edición. Los posibles valores son:
  - **Text:** Construye un campo de texto.
  - **Textarea:** Construye un textarea.
  - **Select:** Construye una combo.
  - **Checkbox:** Construye un check.
  - **Password:** Construye un campo para introducir claves.
  - **Button:** Introduce un botón.
  - **Image:** Introduce una imagen.
  - **File:** Introduce un campo de tipo fichero.

- **Firstsortorder:** Indica la dirección de ordenación de la columna.
  - **asc:** Se ordenará la columna de forma ascendente.
  - **desc:** Se ordenará la columna de forma descendente.
- **Fixed:** Establece que la columna será de dimensiones fijas y no se redimensionará si se cambia el tamaño del grid.
- **Hidden:** Indica si se mostrará o no la columna en el grid, pero también se usa a la hora de la creación del formulario de detalle para indicar si es un campo oculto o no. Aunque la primera propiedad que se mira para crear un campo hidden o no es *edithidden* del conjunto de propiedades de *Reglas de edición (editrules)*.
- **Resizable:** Indica si la columna es redimensionable o no.
- **Sortable:** Indica si la columna es ordenable o no.
- **Title:** Activa o no el tooltip en la columna.
- **Unformat:** Define el nuevo formato cuando una columna deja de ser editable.
- **Opciones de edición:** Conjunto de propiedades relativas a la propiedad *edittype* que se establezca en esta columna.
  - **Value:** Está directamente relacionado a la propiedad *edittype*, su valor será una cadena con dos posibles valores separados por (:). Por ejemplo: {value:"1:One;2:Two"}.
  - **DefaultValue:** Este valor puede ser una cadena o una función. Coge el valor indicado cuando el elemento esté sin ningún valor.
  - **OtherOptions:** Es un array de propiedades están unidas al tipo (*edittype*) que se establezca para esa columna, es decir, contiene información acerca de la columna que se va a editar. Es importante que este array contenga propiedades válidas para el tipo (*edittype*) seleccionado. Por ejemplo si se establece una columna de tipo text los valores de esta propiedad podrían ser todos los valores posibles de un *input* (*size*, *maxlength*, etcétera).
- **Formateador:** Conjunto de propiedades relativas al formato y sus opciones.
  - **Formatter:** Indica el formato a aplicar al visualizar la columna.
  - **Formatoptions:** Se define a las columnas para sobrescribir los valores por defecto.
- **Reglas de edición:** Conjunto de propiedades usadas en la validación de los campos que se crearán en el detalle.
  - **Edithidden:** Si esta propiedad está a true creará un control de tipo hidden.
  - **Required:** Verifica si el valor es obligatorio.
  - **Number:** Verifica si el valor es de tipo numérico.
  - **Integer:** Verifica si el valor es de tipo entero.
  - **MinValue:** Establece un mínimo valor posible.
  - **MaxValue:** Establece un máximo valor posible.
  - **Email:** Verifica si el valor de email tiene un formato correcto.
  - **URL:** Verifica si el valor de URL tiene un formato correcto.
  - **Date:** Verifica si el valor de fecha tiene un formato correcto.
  - **Time:** Verifica si el valor de tiempo tiene un formato correcto.
  - **Custom:** Activa la personalización a través de *Custom\_func*.

- **Custom\_func:** Función personalizada de los roles de edición, vinculado al campo *Custom*.
- Una vez configuradas las columnas con sus correspondientes preferencias y pinchar en el botón *Finish*, se procede a generar el mantenimiento en el proyecto WAR seleccionado a través de las JSPs y la edición del fichero tiles.xml. Además se añade al proyecto de Estáticos el fichero javascript correspondiente que se encarga de implementar el mantenimiento. Esas tareas se informan en la siguiente pantalla como resumen del asistente de generación del mantenimiento.

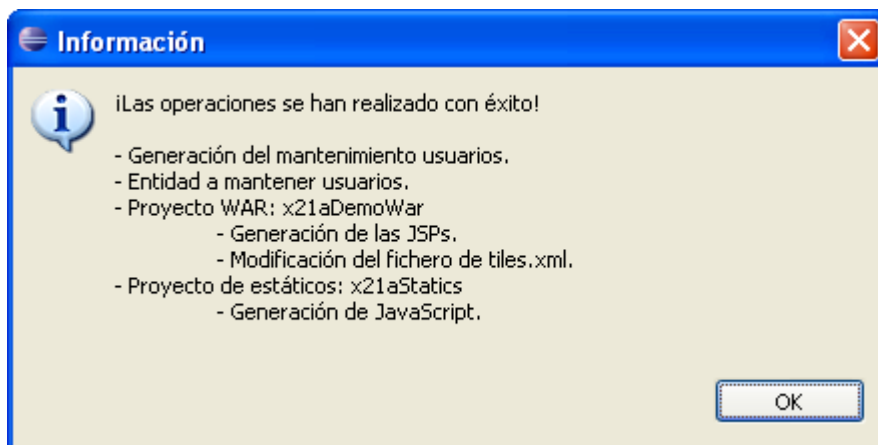






Ilustración 30. Resumen de Crear nuevo mantenimiento.

- Al desplegar la aplicación en un servidor Weblogic, el aspecto final que tendrá el mantenimiento es:

## Mantenimiento (Selección simple)

 Añadir
  Editar
  Eliminar
  Filtrar

Criterio de búsqueda:

id:  nombre:   
 apellido1:  apellido2:   
 ejje:  fechaAlta:   
 fechaBaja:

Buscar [Limpiar](#)

| id   | nombre   | apellido1 | apellido2     | ejje | fechaAlta  | fechaBaja  |
|------|----------|-----------|---------------|------|------------|------------|
| 1    | admin    | a         | informador    | 1    | 21/03/201  | 06/06/201  |
| 10   | CO00397L | Amaia     | desarrollador | 1    | 22/10/2007 | 06/06/2011 |
| 100  | CO01543A | Ivan      | desarrollador | 0    | 24/07/2008 | 06/06/2011 |
| 1000 | CO01655T | Claudia   | informador    | 0    | 25/05/2010 | 14/04/2011 |
| 1001 | CO01650E | Christine | informador    | 1    | 03/06/2010 | 14/04/2011 |
| 1002 | CO01639A | Christine | desarrollador | 0    | 10/02/2011 | 26/05/2011 |
| 1003 | CO01638K | Carolina  | manager       | 0    | 30/03/2010 | 14/04/2011 |
| 1004 | CO01635T | Carlos    | espectador    | 0    | 20/02/2007 | 03/06/2011 |
| 1005 | CO01632H | Carlos    | informador    | 1    | 25/05/2011 | 31/05/2011 |
| 1006 | CO01631T | Carlos    | informador    | 1    | 11/02/2009 | 05/03/2009 |

[Primera Página](#) [Anterior](#)
 Página  de 112
 [Siguiente](#) [Última Página](#)

Mostrando 1 - 10 de 1.114

Ilustración 31. Ejemplo de mantenimiento (grid).

**Agregar Registro**
cerrar x

 1 de 1114 Elementos
 [Primero](#) [Anterior](#) [Siguiente](#) [Último](#)

id  nombre  apellido1   
 apellido2  ejje  fechaAlta   
 fechaBaja

Guardar [Cancelar](#)

Ilustración 32. Ejemplo de mantenimiento (detalle).

### 3.5 Añadir un proyecto EJB a una aplicación

Este asistente es el encargado de generar un módulo EJB y de asociarlo a una aplicación UDA existente.

- En la categoría UDA seleccionamos la opción *Añadir un proyecto EJB a una aplicación*.

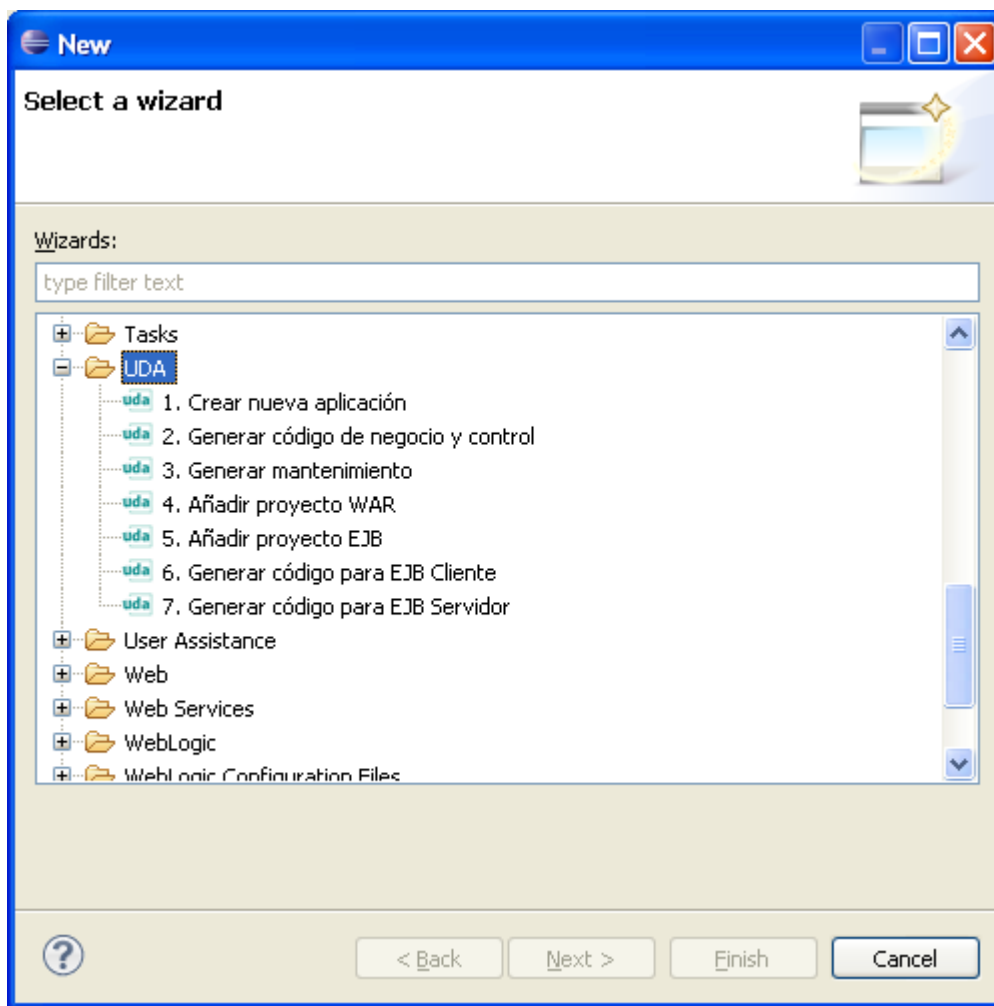


Ilustración 33. Listado de asistentes UDA – Añadir un proyecto EJB a una aplicación

- En la siguiente pantalla del asistente se solicitan los datos de la aplicación para la que se quiere generar EJB y el nombre del mismo.

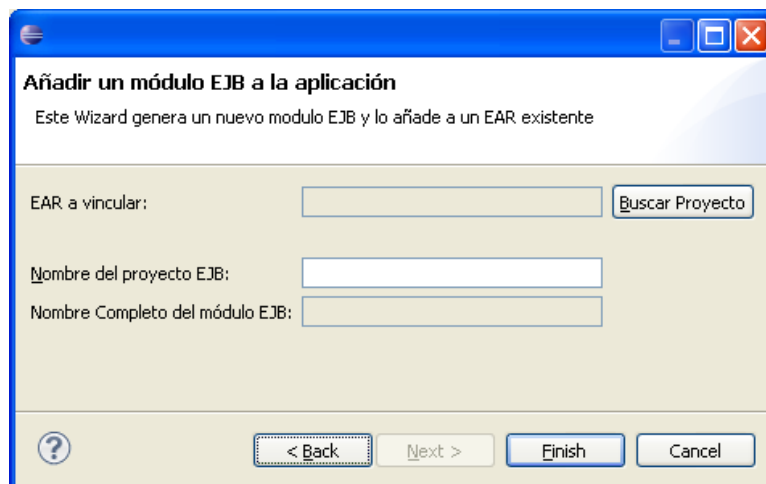


Ilustración 34. Datos generación proyecto EJB

- **EAR a vincular:** Campo obligatorio, se selecciona un EAR de tipo UDA entre los existentes en el workspace.
  - **Nombre del EJB:** Campo obligatorio para indicar el nombre que se quiere dar al módulo EJB.
  - **Nombre Completo del módulo EJB:** Campo de sólo lectura que indica el nombre del proyecto EJB que se va a generar.
- Si la generación del proyecto EJB y su enlace con la aplicación han sido satisfactorios, se muestra un resumen de las acciones que ha realizado el asistente.

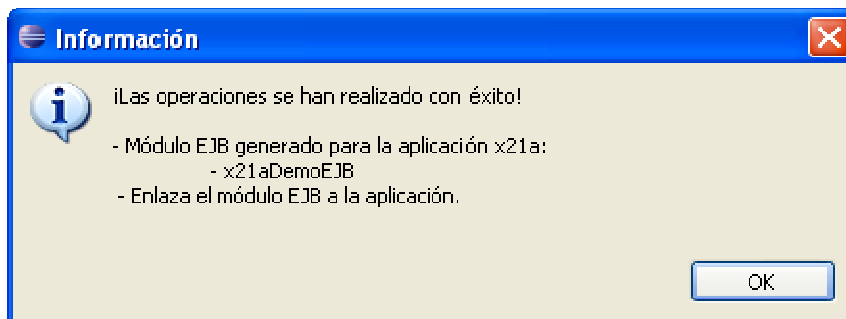


Ilustración 35. Resumen de tareas realizadas en la creación del EJB

- Se visualiza en nuevo proyecto EJB en el workspace.

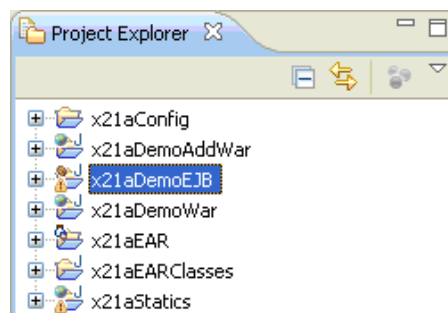


Ilustración 36. Proyecto EJB añadido a la aplicación UDA



### 3.6 Generar código para EJB cliente

Este asistente es el encargado de generar el código de un EJB Cliente para el consumo de servicios remotos expuestos por terceras aplicaciones. Para su correcto funcionamiento es necesario disponer del cliente del ejb en el que está expuesta la interfaz remota del servicio que se quiere consumir pudiendo tratarse de ejb 2.0 o un ejb 3.0.

- En la categoría UDA seleccionamos la opción *Generar código para EJB cliente*.

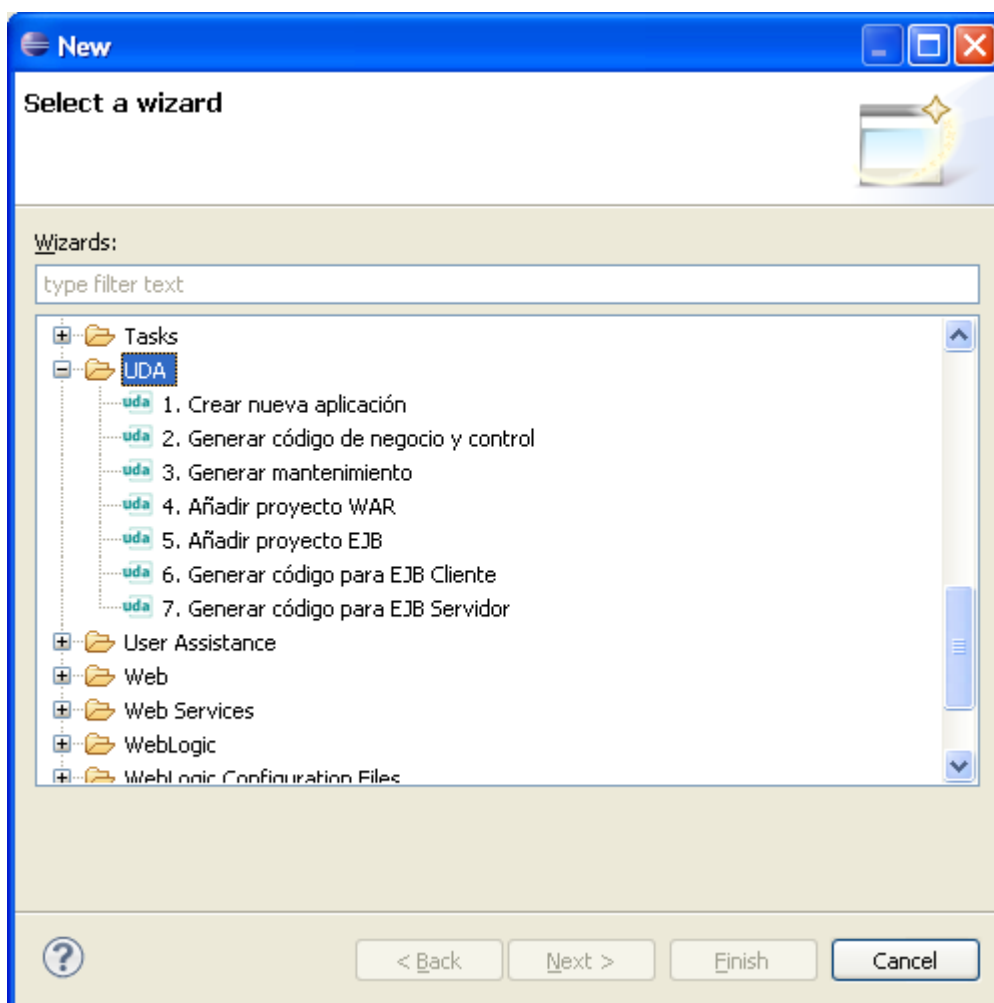
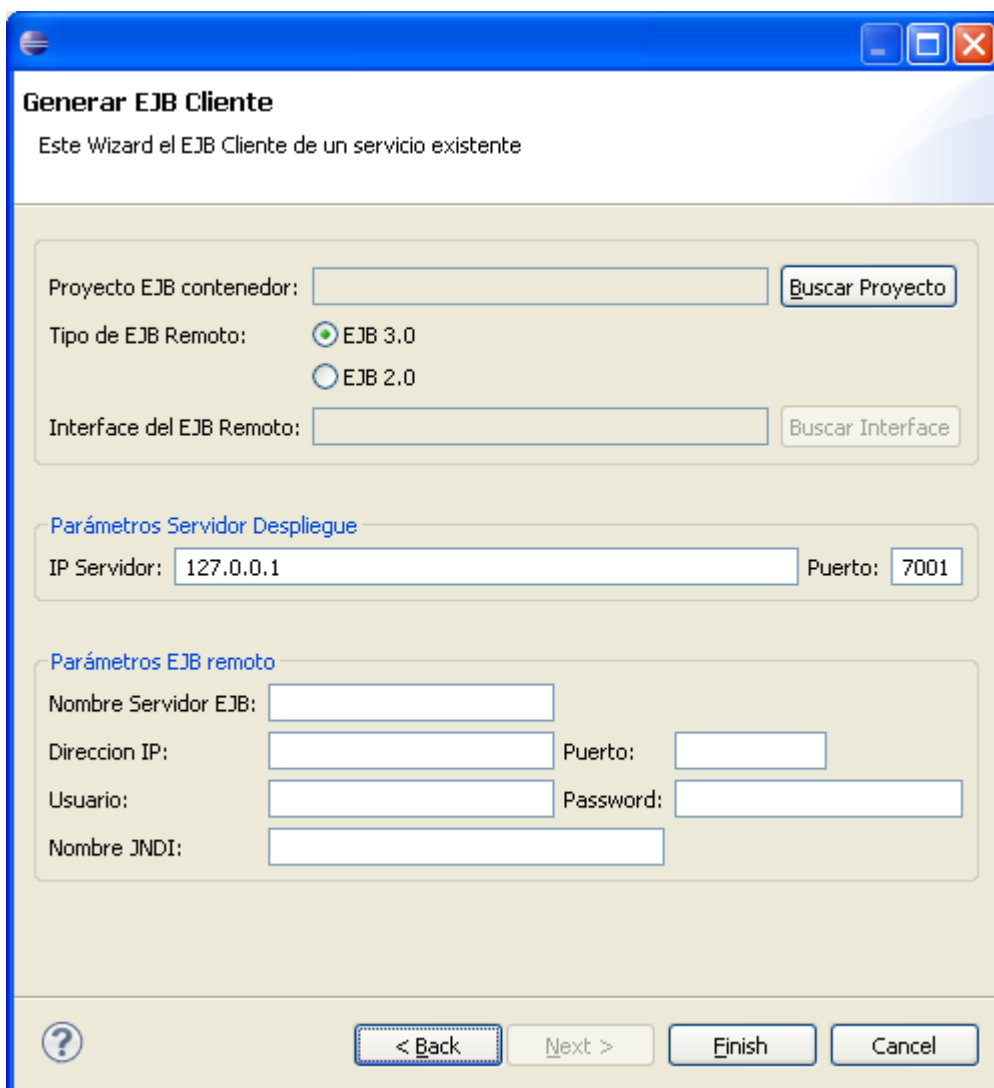


Ilustración 41. Listado de asistentes UDA – Generar código para EJB Cliente

- En la pantalla de este asistente, se piden los parámetros necesarios para la generación del código (*stubs*) del EJB partiendo la librería cliente que se encontrará entre las dependencias de la aplicación (denominada como *codAppRemoting.jar*).



**Generar EJB Cliente**

Este Wizard el EJB Cliente de un servicio existente

Proyecto EJB contenedor:

Tipo de EJB Remoto: ☒ EJB 3.0  
☐ EJB 2.0

Interface del EJB Remoto:

**Parámetros Servidor Despliegue**

IP Servidor:  Puerto:

**Parámetros EJB remoto**

Nombre Servidor EJB:

Direccion IP:  Puerto:

Usuario:  Password:

Nombre JNDI:

Ilustración 42. Pantalla parámetros de generación EJB Cliente

- **Proyecto EJB contenedor:** Campo obligatorio para seleccionar el proyecto EJB entre los existentes en el workspace.
- **Tipo de EJB remoto:** Campo obligatorio para elegir la tecnología que utiliza el EJB Servidor del que se va a generar el EJB Cliente.
  - **EJB 3.0:** Se selecciona cuando el EJB Servidor utiliza tecnología EJB 3.0.
  - **EJB 2.X:** se selecciona cuando el EJB servidor utiliza tecnología EJB 2.X.
- **Interface del EJB Remoto:** Campo obligatorio para escoger la interfaz del EJB Servidor para el que se generará el EJB Cliente (stub) correspondiente. En función de la tecnología seleccionada en el *Tipo de EJB remoto*, se filtrarán las interfaces que se podrán seleccionar en la siguiente pantalla:

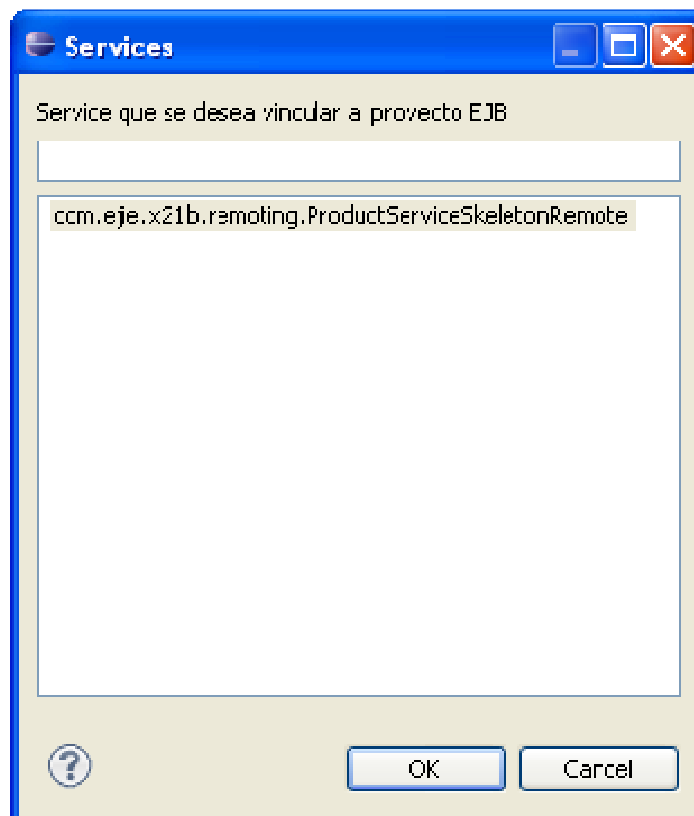


Ilustración 43. Pantalla de selección del interfaz EJB a consumir

En caso de que no existir ningún interface de EJB Remoto, se mostrará la siguiente pantalla:

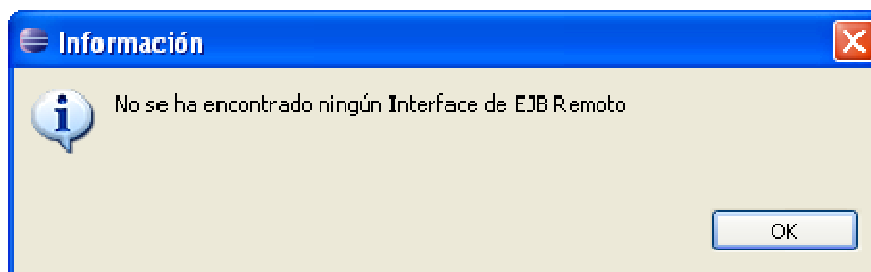


Ilustración 44. Resumen de tareas realizadas

- **Parámetros Servidor Despliegue:** Parámetros donde esta expuesto el servicio a consumir
  - **IP Servidor:** Campo obligatorio, en el que se debe introducir la dirección IP o el nombre del host en el que se encuentra desplegado el EJB del servicio a consumir.
  - **Puerto:** Campo obligatorio, en el que se debe introducir el puerto del servidor en el que se encuentra desplegado el EJB del servicio a consumir.
- **Parámetros EJB remoto:** Parámetros para la generación del código del cliente EJB
  - **Nombre Servidor EJB:** Campo obligatorio para informar el nombre del servidor en el que se va a desplegar el cliente.
  - **Dirección IP:** Campo obligatorio para informar la dirección IP o el nombre del host en el que se va a desplegar el cliente.

- **Puerto:** Campo obligatorio para indicar el puerto del servidor en el que se va a desplegar el cliente.
  - **Usuario:** Campo obligatorio para informar el usuario del servidor en el que se va a desplegar el cliente.
  - **Password:** Campo obligatorio para informar el password del servidor en el que se va a desplegar el cliente.
  - **Nombre JNDI:** Campo obligatorio para informar el nombre completo del JNDI del servidor en el que se va a desplegar el cliente.
- Tras cumplimentar los campos y pulsar el botón *Finish*, si el resultado de la generación ha sido satisfactorio, se muestra el resumen del mismo en la siguiente pantalla:

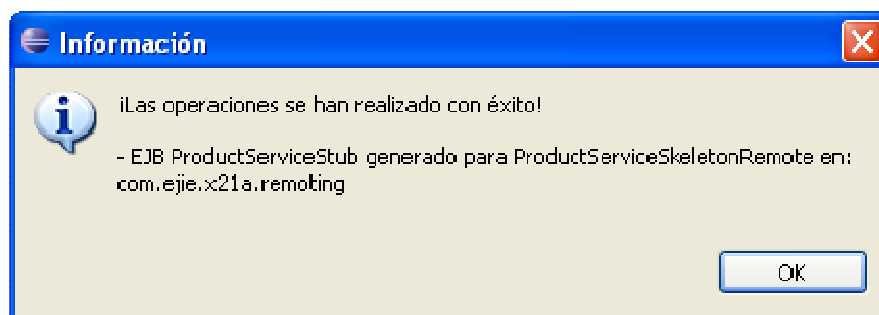


Ilustración 45. Pantalla Resumen de generación satisfactoria de un EJB Cliente

### 3.7 Generar código para EJB servidor

Este asistente es el encargado de generar las interfaces de los servicios de una aplicación UDA que se expondrán como EJB a fin de exponerlos para su consumo.

- En la categoría UDA seleccionamos la opción *Generar código para EJB servidor*.

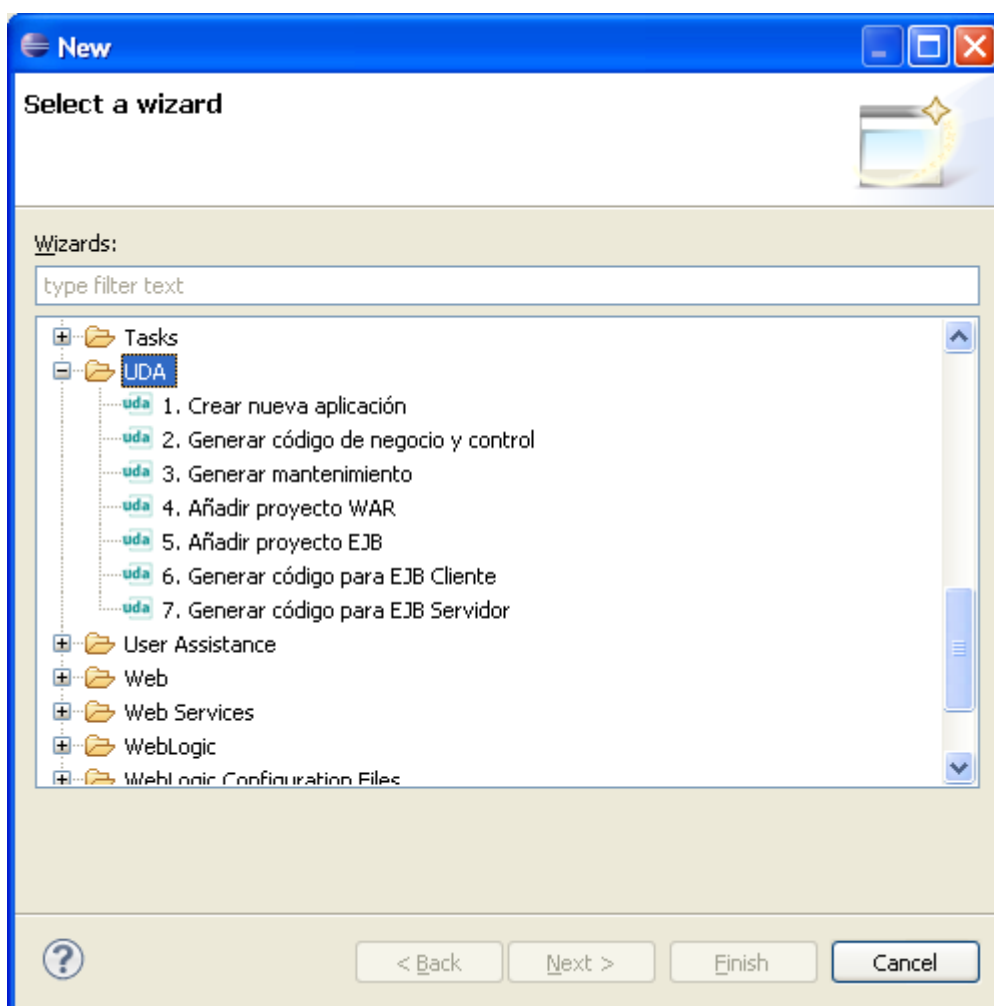


Ilustración 37. Listado de asistentes UDA – Generar código para EJB servidor

- En la pantalla de este asistente, se piden los parámetros necesarios para la generación del código (*Skeleton*) del servicio seleccionado por el usuario.

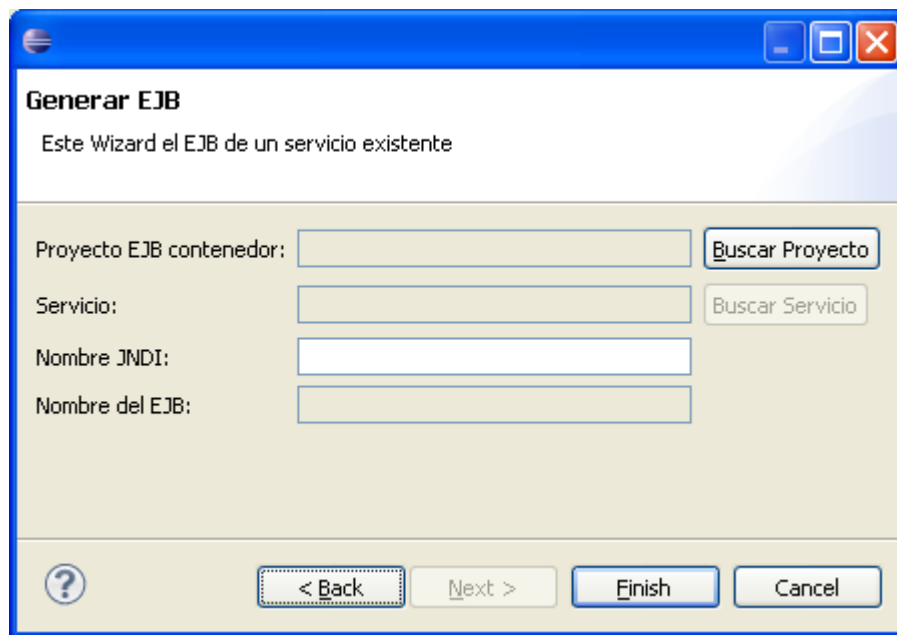


Ilustración 38. Pantalla parámetros de generación EJB servidor

- **Proyecto EJB contenedor:** Campo obligatorio para seleccionar el proyecto EJB de tipo UDA entre los existentes en el workspace.
- **Servicio:** Campo obligatorio para seleccionar el servicio que se desea exponer entre los disponibles en la aplicación (se buscan dentro del proyecto EARClasses en el paquete *com.ejie.codapp.service.\**).

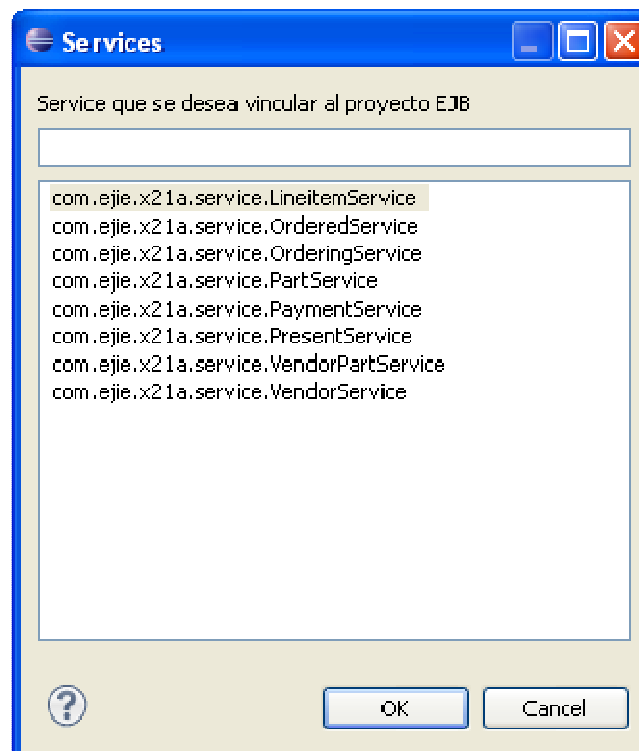


Ilustración 39. Pantalla seleccionar los servicios para generar los Skeleton

- **Nombre JNDI:** Campo obligatorio para indicar el nombre del JNDI completo por el cual va a ser accesible el servicio expuesto.
  - **Nombre del EJB:** Campo de solo lectura para mostrar la clase dentro del proyecto EJB que se va a generar (*servicioSkeleton*).
- Una vez introducidos todos los campos, si el proceso ha funcionado correctamente, se mostrará la siguiente pantalla donde aparecerá un resumen de las acciones realizadas:

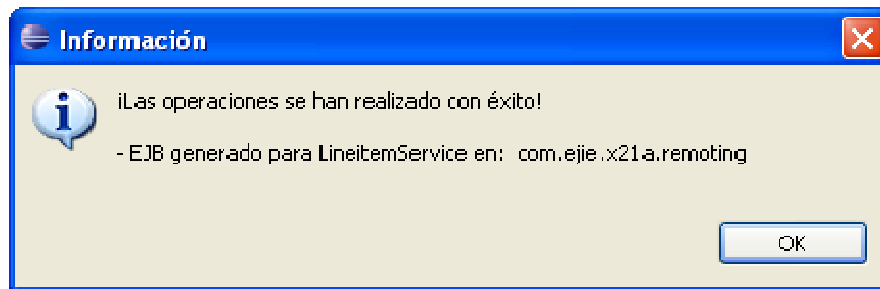


Ilustración 40. Resumen de tareas realizadas en la creación del EJB servidor

## 4 Acceso a una aplicación UDA

Una vez se ha generado el código necesario para el funcionamiento de la aplicación utilizando los asistentes para desplegar la aplicación y acceder a la misma hay que realizar las siguientes acciones:

### 4.1 Generar el datasource en la consola de Weblogic

Las conexiones a las bases de datos desde las aplicaciones, se realizan a través de DataSources. Para crear un datasource XA (o no XA) con el cliente 11g de Oracle que viene por defecto con WebLogic Server 11 (10.3.1.0), se han de seguir los siguientes pasos:

1. Arrancar el servidor. (Se puede realizar desde el propio Eclipse, en la vista de Servers)

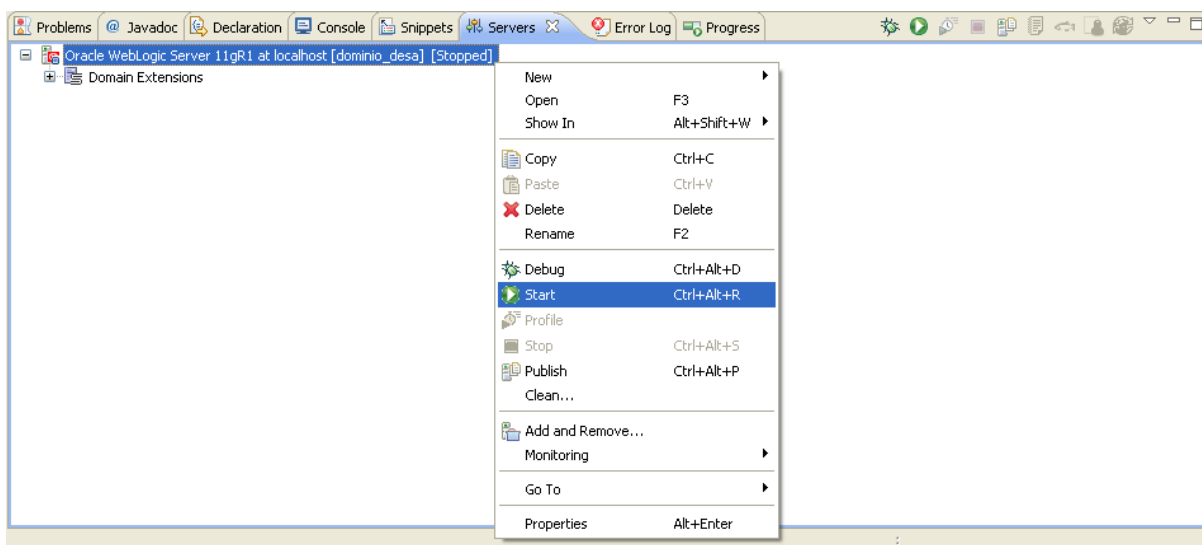


Ilustración 41. Arrancar el servidor desde Eclipse

2. Una vez arrancado, abrir la consola de WebLogic desde un navegador: <http://localhost:7001/console> o bien desde el propio Eclipse (usuario/passwor de la instalación).

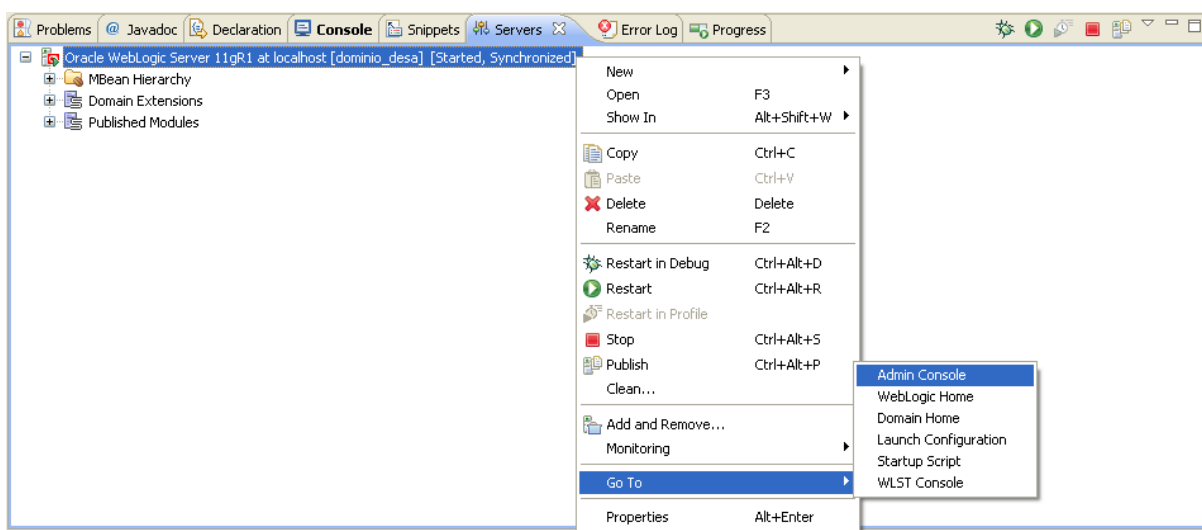


Ilustración 42. Acceso a la consola de administración desde Eclipse



- Para crear un nuevo DataSource seleccionar Services > JDBC > DataSources, pulsar en 'New'.

Name: bbbDataSource o bbbDataSourceXA, según la nomenclatura establecida en la normativa de desarrollo de WLS 11g

JNDI name: bbb.bbbDataSource o bbbDataSourceXA

Database Driver: En el caso de bases de datos Oracle, seleccionar el Driver 'Oracle's Driver (Thin) for Instance Connections; versions 9.0.1, 9.2.0, 10, 11' u 'Oracle's Driver (Thin XA) for Instance Connections; versions 9.0.1, 9.2.0, 10, 11'.

El asistente de generación de proyecto del plugin UDA crea el fichero **dao-config.xml** (*[código de aplicación]EARClasses/src/META-INF/spring/*) donde se figura el nombre del datasource que utilizará la aplicación.

```
<?xml version="1.0" encoding="utf-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:context="http://www.springframework.org/schema/context"
xmlns:jee="http://www.springframework.org/schema/jee"

xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-3.0.xsd
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-3.0.xsd
http://www.springframework.org/schema/jee http://www.springframework.org/schema/jee/spring-
jee-3.0.xsd">

<jee:jndi-lookup id="dataSource" jndi-name="x21a.x21aDataSource" resource-ref="false" />
<!-- Scans the classpath of this application for @Repository to deploy as beans -->
<context:component-scan base-package="com.ejie.x21a.dao" />

</beans>
```

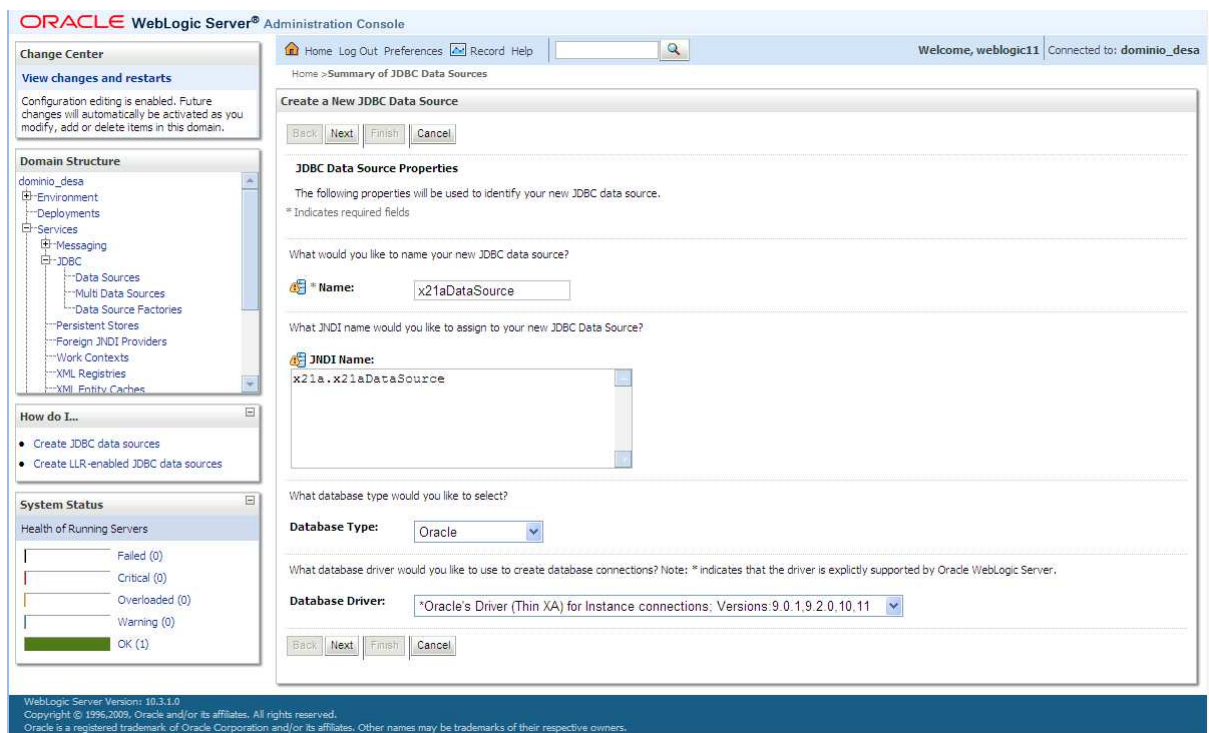


Ilustración 43. Configuración de un datasource en la consola de Weblogic

La creación de un dataSource genera automáticamente un Connection Pool, con el driver:

- “oracle.jdbc.xa.client.OracleXADataSource” para Thin XA
- “oracle.jdbc.OracleDriver” para Thin no XA

## 4.2 Desplegar la aplicación

Para desplegar la aplicación bastará publicar en el servidor el proyecto *EAR* y el proyecto *Statics*.

Desde el propio Eclipse, en la vista Servers, con el botón derecho sobre el servidor de despliegue, se selecciona *Add and Remove*:

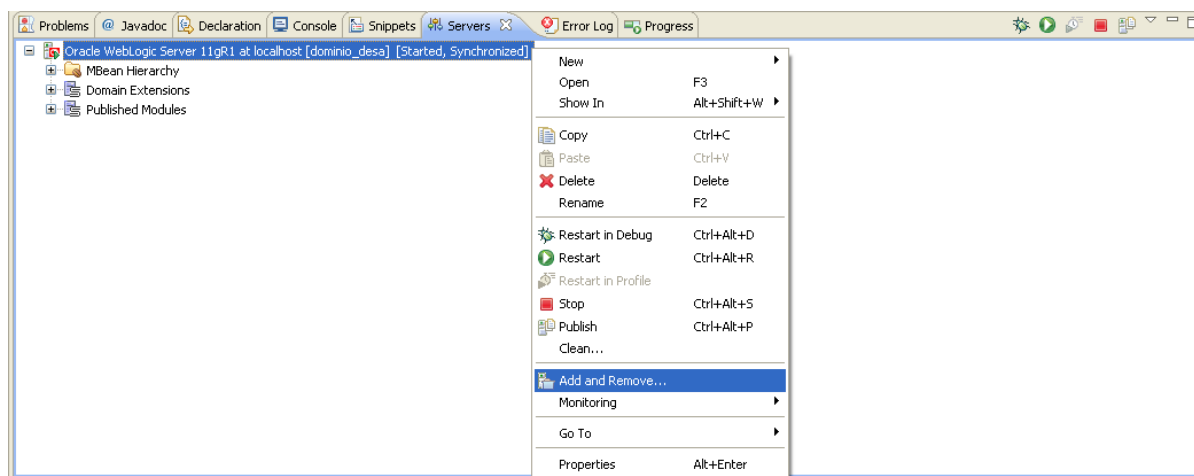


Ilustración 44. Configuración de los módulos a desplegar en el servidor Weblogic

Aparecerá la siguiente pantalla para seleccionar los proyectos a publicar:

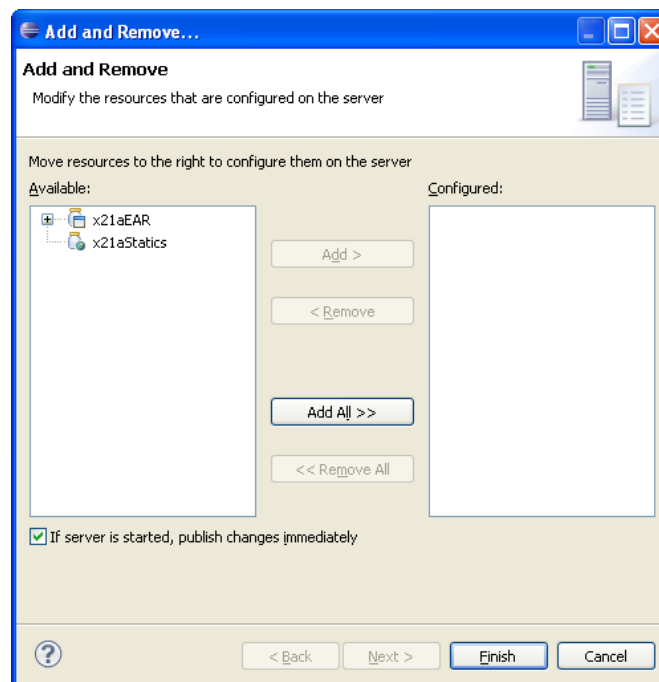


Ilustración 45. Módulos a desplegar en el servidor Weblogic

*El proyecto de estáticos ([codigoAplicación]Statics) que genera el plugin UDA no está vinculado al proyecto EAR de la aplicación y por tanto hay que desplegarlo de forma independiente. Esto se debe a la especialización del entorno de despliegue, es decir, un servidor web se puede encargar de resolver los recursos estáticos (js, html, css, imágenes) y un servidor de aplicaciones y/o contenedor de EJBs resolver las peticiones dinámicas (jsp's).*

*La relación entre ambos módulos queda establecida mediante la configuración de la url de acceso a los recursos estáticos en el fichero códigoAplicación.properties en el proyecto [códigoAplicación]Config según la configuración de la preferencia de Desarrollo para "EJIE":*

*statics.path = <http://localhost:7001/codappStatics>*

*statics.path = <http://desarrollo.jakina.ejiedes.net:7001/codappStatics>*

*NOTA: desarrollo.jakina.ejiedes.net estará configurado en el fichero hosts*

Una vez el servidor ha desplegado los proyectos, la url para el acceso a la aplicación dependerá de la configuración "Desarrollo para EJIE" en las preferencias del plugin:

Si la casilla de verificación no está activa:

[http://localhost:7001/\[código de aplicación\] + \[nombre del WAR\] + War/](http://localhost:7001/[código de aplicación] + [nombre del WAR] + War/)

En caso de tenerla activa

[http://desarrollo.jakina.ejiedes.net:7001/\[código de aplicación\] + \[nombre del WAR\] + War/](http://desarrollo.jakina.ejiedes.net:7001/[código de aplicación] + [nombre del WAR] + War/)

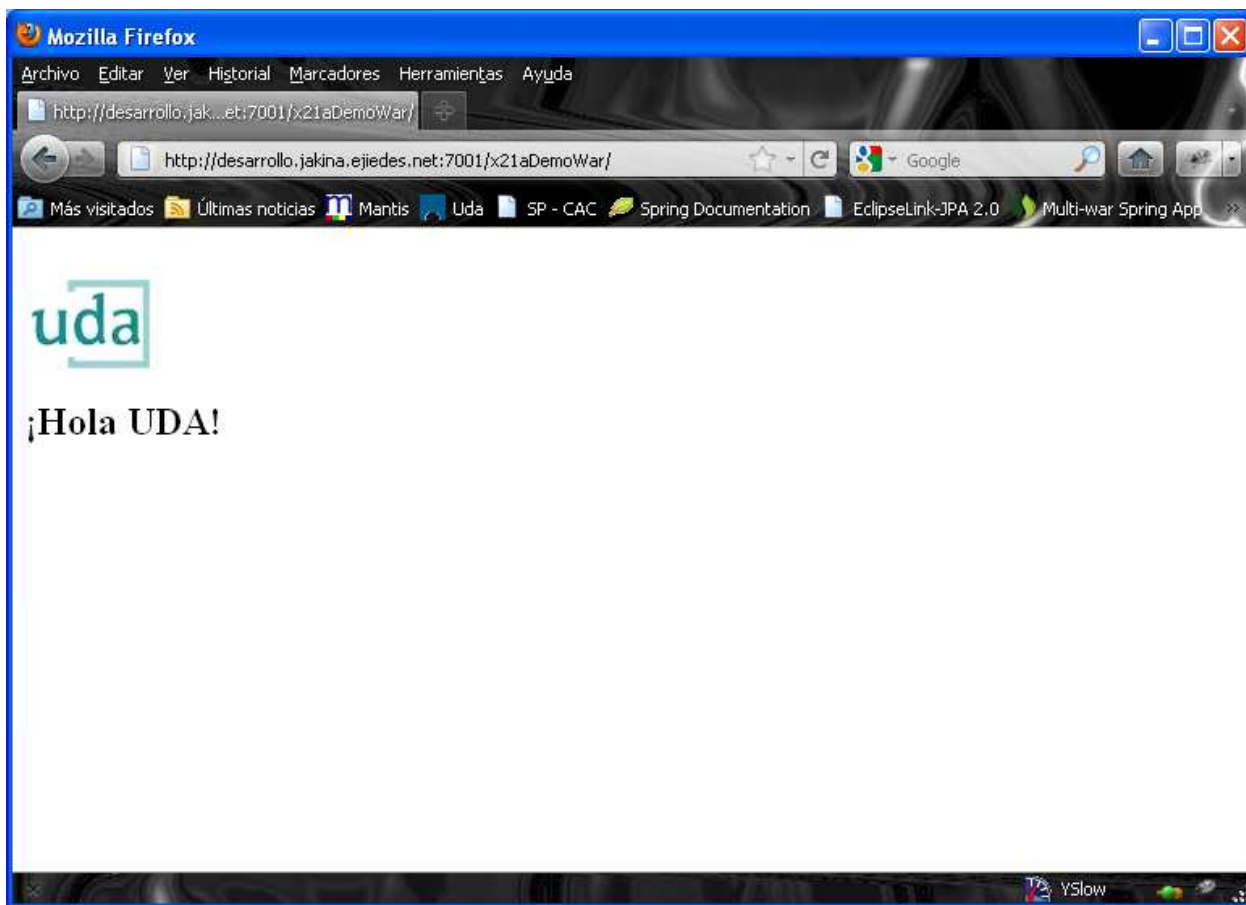
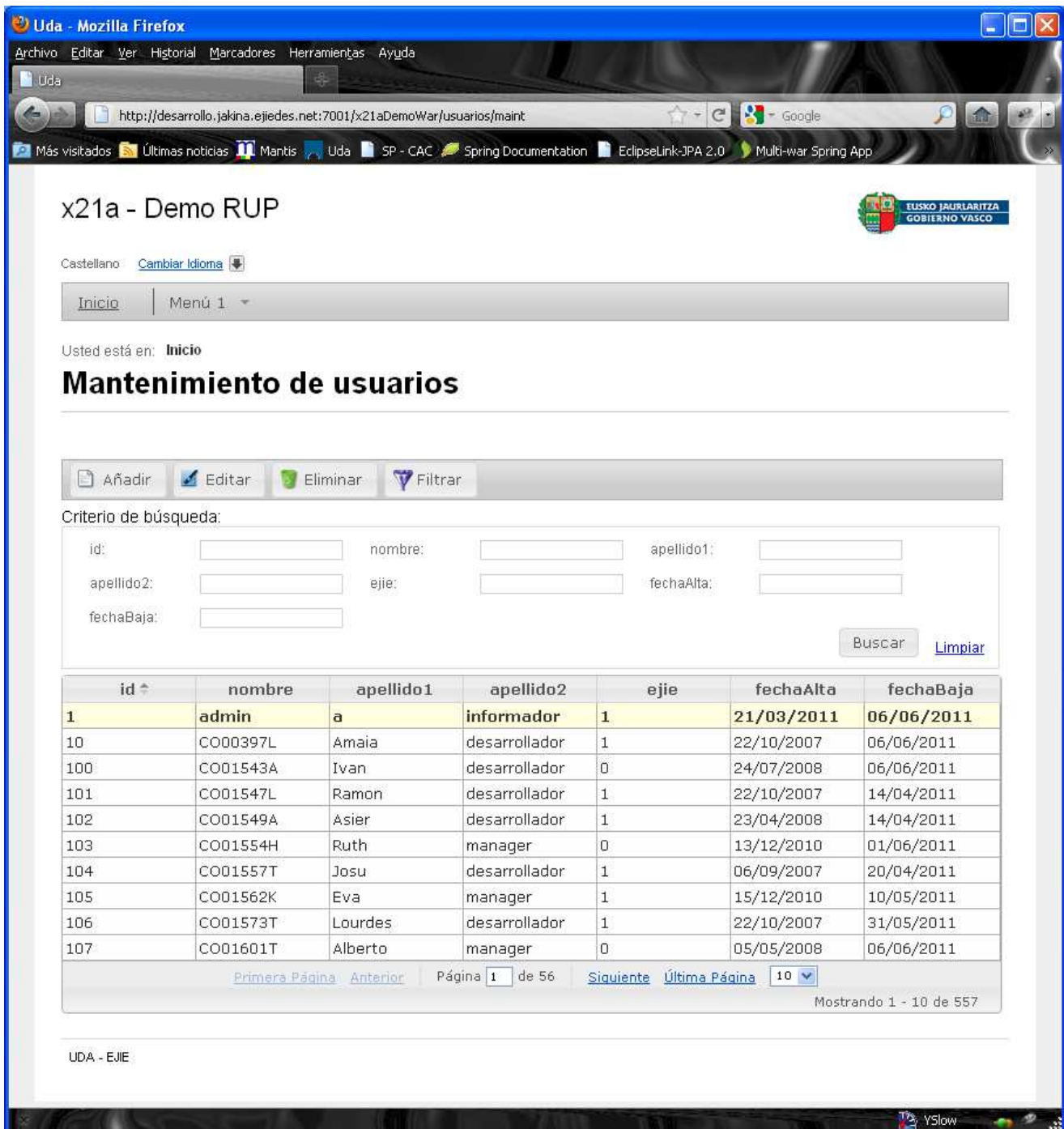


Ilustración 46. Acceso a una aplicación UDA

En caso de haber generado un mantenimiento mediante el asistente, la url de acceso a la aplicación tendrá en cuenta la entidad para la que se ha generado:

http://desarrollo.jakina.ejjes.net:7001/[código de aplicación] + [nombre del WAR] + War/[nombreEntidad]/maint



**x21a - Demo RUP**

Castellano [Cambiar Idioma](#)

[Inicio](#) | Menú 1

Usted está en: **Inicio**

## Mantenimiento de usuarios

[Añadir](#) [Editar](#) [Eliminar](#) [Filtrar](#)

Criterio de búsqueda:

id:  nombre:  apellido1:   
 apellido2:  ejje:  fechaAlta:   
 fechaBaja:

[Buscar](#) [Limpiar](#)

| id  | nombre   | apellido1 | apellido2     | ejje | fechaAlta  | fechaBaja  |
|-----|----------|-----------|---------------|------|------------|------------|
| 1   | admin    | a         | informador    | 1    | 21/03/2011 | 06/06/2011 |
| 10  | CO00397L | Amaia     | desarrollador | 1    | 22/10/2007 | 06/06/2011 |
| 100 | CO01543A | Ivan      | desarrollador | 0    | 24/07/2008 | 06/06/2011 |
| 101 | CO01547L | Ramon     | desarrollador | 1    | 22/10/2007 | 14/04/2011 |
| 102 | CO01549A | Asier     | desarrollador | 1    | 23/04/2008 | 14/04/2011 |
| 103 | CO01554H | Ruth      | manager       | 0    | 13/12/2010 | 01/06/2011 |
| 104 | CO01557T | Josu      | desarrollador | 1    | 06/09/2007 | 20/04/2011 |
| 105 | CO01562K | Eva       | manager       | 1    | 15/12/2010 | 10/05/2011 |
| 106 | CO01573T | Lourdes   | desarrollador | 1    | 22/10/2007 | 31/05/2011 |
| 107 | CO01601T | Alberto   | manager       | 0    | 05/05/2008 | 06/06/2011 |

[Primera Página](#) [Anterior](#) Página 1 de 56 [Siguiente](#) [Última Página](#) 10

Mostrando 1 - 10 de 557

UDA - EJJE

Ilustración 47. Acceso a un mantenimiento UDA

## 5 Barra de herramientas de patrones UDA

Para facilitar la labor del programador a la hora de incluir un determinado patrón en la aplicación que se está desarrollando, se ha incluido una barra de herramientas accesible desde la vista de *Snippets* de Eclipse donde se visualizarán los patrones de presentación de UDA.

Un snippet es una porción de código que se utiliza para la generación de código de manera automática facilitando la implementación a la hora de programar.

### 5.1 Tipos de patrones UDA

El entorno de desarrollo Eclipse proporciona la capacidad de crear un listado de snippets y agruparlos en una categoría. En la vista de snippets se ha creado la categoría *UDA* que contendrá los siguientes patrones:

- **Autocomplete:** patrón para sugerir resultados coincidentes con el texto de búsqueda que esta escribiendo el usuario.
- **Combo:** patrón para guiar al usuario en la introducción de valores tipificados pudiendo ser una o varias listas de valores independientes o dependientes entre sí.
- **Diálogo:** patrón que permite lanzar un subproceso dentro de un proceso principal sin salirse de éste. Su funcionalidad es similar al antiguo pop-up.
- **Feedback:** patrón que informa al usuario del resultado de una acción dentro de la aplicación.
- **Grid:** patrón que presenta la información al usuario de forma tabular facilitando la visualización y la búsqueda de datos posibilitando la ordenación de las columnas en la tabla y de los datos de cada columna.
- **Idioma:** patrón que permite al usuario elegir el idioma en el que se le presenta la aplicación.
- **Mantenimiento:** patrón definido para facilitar la lógica necesaria en las acciones básicas sobre un conjunto de datos denominadas CRUD (Create, Read, Update y Delete).
- **Mensajes:** patrón que tiene el objetivo de mostrar al usuario de forma homogénea, clara y llamativa los posibles mensajes que pueden desencadenar las acciones que realiza la aplicación. Se trata de mensajes de advertencia, aceptación, confirmación o error.
- **Menú:** patrón que muestra las entradas directas a secciones clave de la aplicación.
- **Migas:** patrón que muestra al usuario la ruta de navegación que ha seguido en la aplicación desde la página de inicio.
- **Toolbar:** patrón que contiene un conjunto de botones con distintas finalidades.
- **Tooltip:** patrón que proporciona ayuda contextual al usuario para la introducción de datos en la aplicación.

### 5.2 Utilizar un snippet

Cuando se arranca el Eclipse, dependiendo de la perspectiva que esté activada, se podrá visualizar directamente la vista de Snippets. En el caso que la vista no se visualice, podremos habilitarla de la siguiente forma.

- Nos dirigimos al menú *Windows > Show View > Snippets*, según se ve en la siguiente ilustración

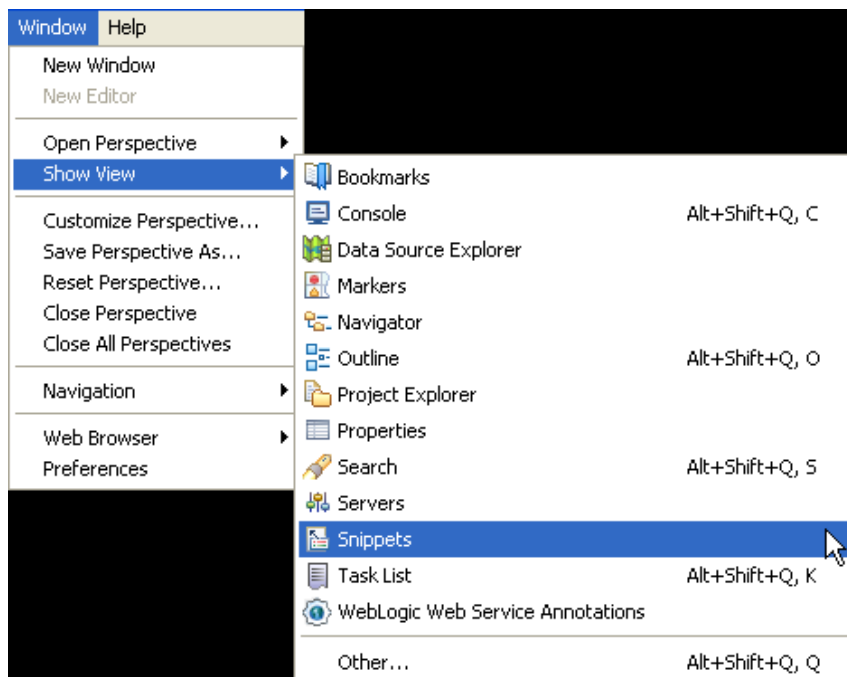


Ilustración 48. Visualizar la vista de Snippets de Eclipse

Con eso se visualizará el listado de patrones de presentación dentro de la categoría UDA de esta vista.

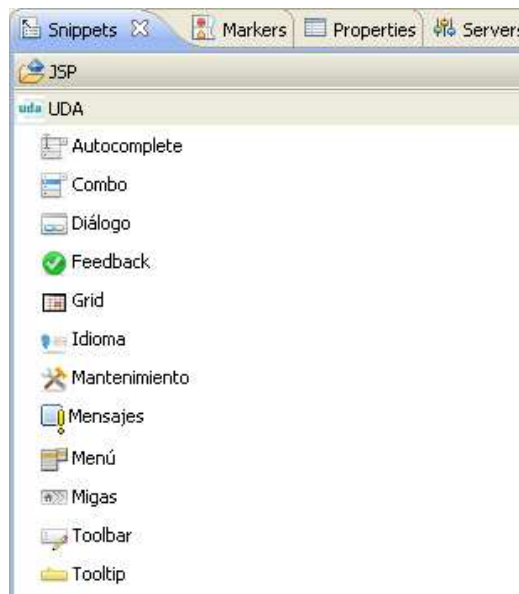


Ilustración 49. Snippets de patrones UDA.

Una vez que tengamos la vista habilitada, pasaremos a editar el fichero en el que queramos añadir el código de un patrón UDA.

- Con el ratón, haremos un doble click o arrastraremos el patrón al editor e código que se quiere añadir.

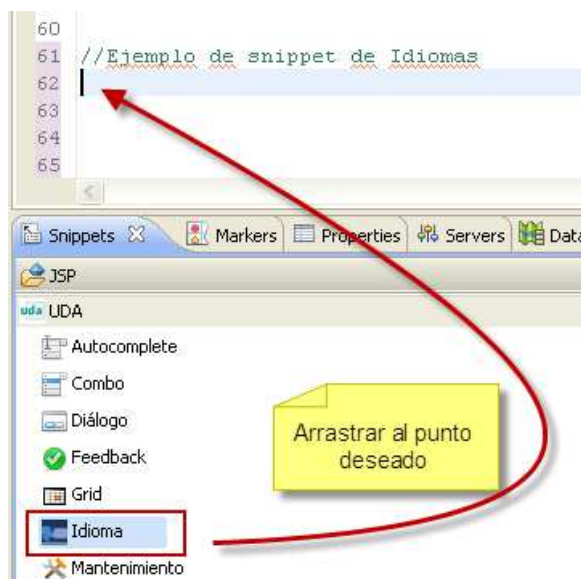


Ilustración 50. Arrastrar el patrón al código.

- Si el snippet tiene parámetros de entrada, antes de pintar el código, se mostrará la siguiente ventana, donde podremos rellenar el valor que va a tener determinada parte del código.

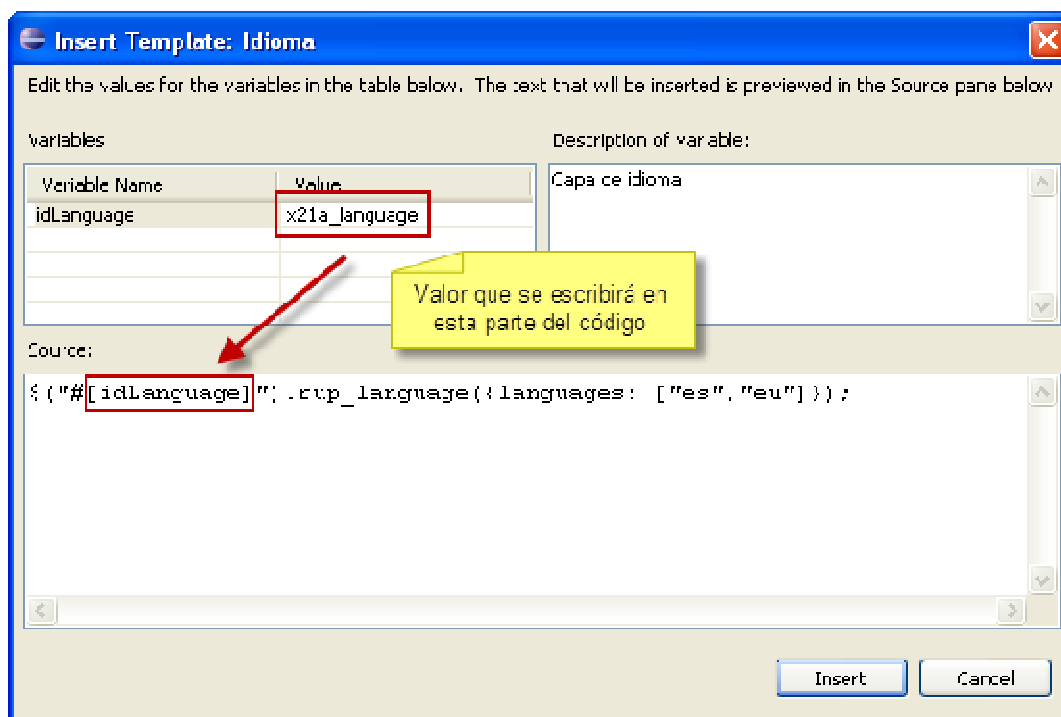


Ilustración 51. Pantalla para insertar la parte variable del código.

- El resultado final del uso del snippet es la inclusión del código en el fichero que se estaba editando.

```
57  
58     });  
59  
60  
61 //Ejemplo de snippet de Idiomas  
62 $( "#x21a_language" ).rup_language( { languages: [ "es", "eu" ] } );  
63  
64  
65  
66
```

Ilustración 52. Resultado de la utilización del snippet.