CONSEJERÍA DE HACIENDA Y ADMINISTRACIÓN PÚBLICA

JUNTA DE ANDALUCIA
CONSEJERÍA DE HACIENDA Y ADMINISTRACIÓN PÚBLICA

Formación a usuarios y personal técnico:
Ventanilla Electrónica de la Administración

Dirección General de Política Digital
Enero de 2018

Ventanilla Electrónica de la Administración (VEA)

Ventanilla Electrónica de la Administración (VEA)

Introducción

El objetivo de esta sesión es detallar las especificaciones a seguir para la integración de VEA con los servicios externos que amplían su comportamiento base.

Para ello se definirán los puntos en los que VEA realiza conexiones con los servicios externos durante el ciclo de vida de un expediente en un trámite administrativo, desde la creación del borrador hasta la realización de tareas de tramitación sobre el expediente.

Estas pautas son de obligado cumplimiento para lograr la conexión de VEA con los distintos módulos.

Ventanilla Electrónica de la Administración (VEA)

I Introducción

II Conexión con servicios externos

III Componente numerador

IV Procesamiento específico

V Validación específica

VI Visibilidad

VII Requisitos a subsanar

Conexión con servicios externos

Puntos de conexión

Durante el ciclo normal de realización de una entrega, VEA realiza una serie de llamadas a distintos servicios externos, que se pueden personalizar.

De esta forma se permite extender la funcionalidad base que ofrece, mediante invocaciones a servicios externos.

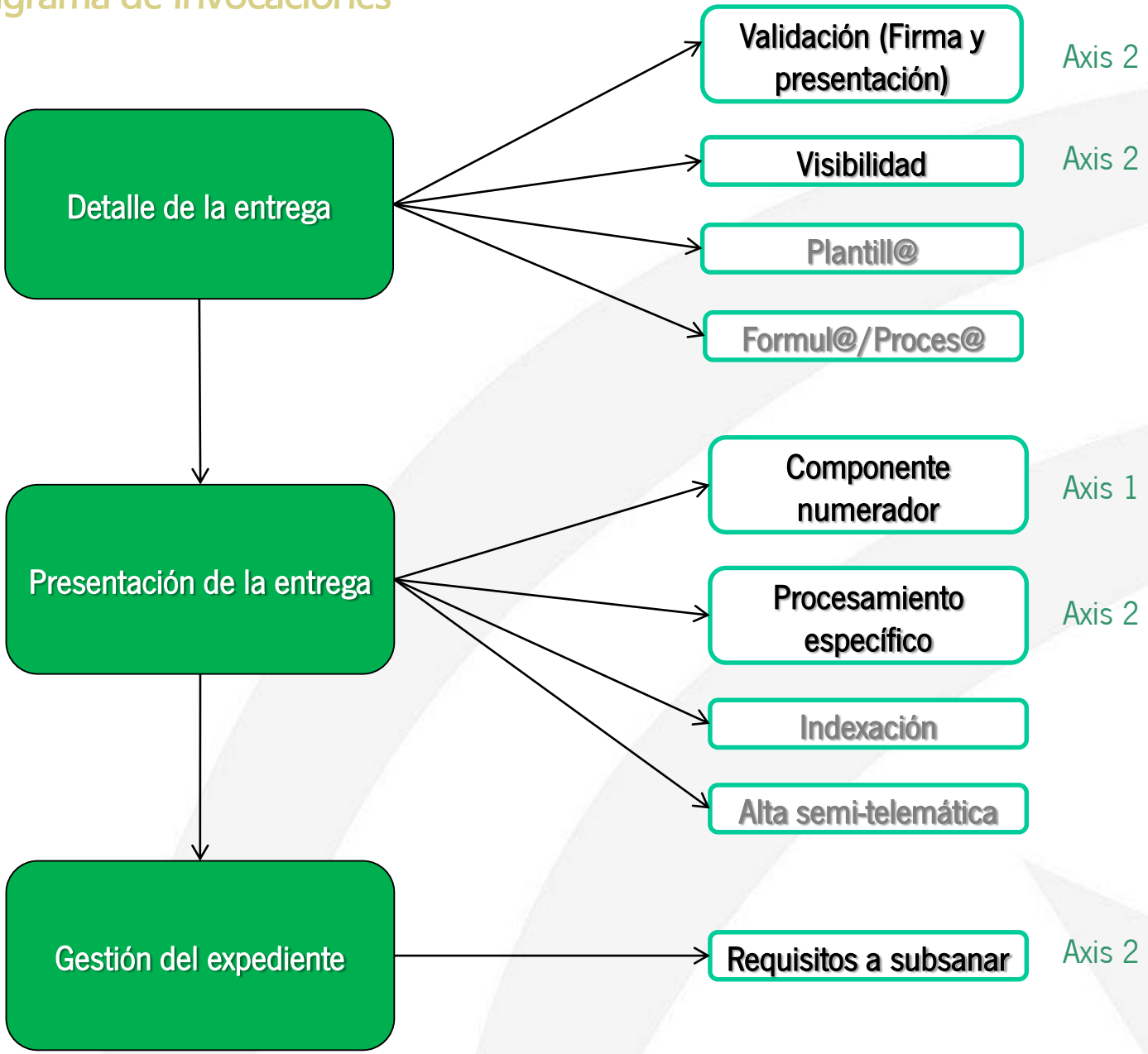
Esta invocación se realiza a través de clientes de servicios web predefinidos e invariables (exceptuando Plantill@, Formul@ y Proces@), de forma que los servicios deben acoplarse con estos clientes.

Para la implementación de estos servicios se distribuyen sus ficheros .wsdl así como un pequeño proyecto Java mavenizado para facilitar el desarrollo del servicio web.

Cada servicio web debe implementarse siguiendo los descriptores proporcionados. De otra forma, el VEA no podrá invocarlos.

Conexión con servicios externos

Diagrama de invocaciones



Ventanilla Electrónica de la Administración (VEA)

- I Introducción
- II Conexión con servicios externos
- III Componente numerador
- IV Procesamiento específico
- V Validación específica
- VI Visibilidad
- VII Requisitos a subsanar

Componente numerador

Introducción

VEA genera por defecto los números de expediente de forma fija, siguiendo el patrón **CODIGO-ORGANISMO/[año-en-curso]/[contador-ascendente]**, completando un número total de caracteres de 25 de forma que se cumpla con la especificación de numeración de expediente del Esquema Nacional de Interoperabilidad.
Ej: CJI/2018/0000000000000025

Sin embargo en determinadas ocasiones y para tramites concretos, se observa la necesidad de generar números de expediente en base a determinada información de la propia solicitud. En este caso, VEA permite la integración con un componente numerador configurado en el motor de tramitación Trew@ que genere un número de expediente distinto.

A la hora de crear el expediente, VEA invoca a una URL determinada que debe contener la implementación del numerador. Esta URL se configura como componente numerador en Trew@.

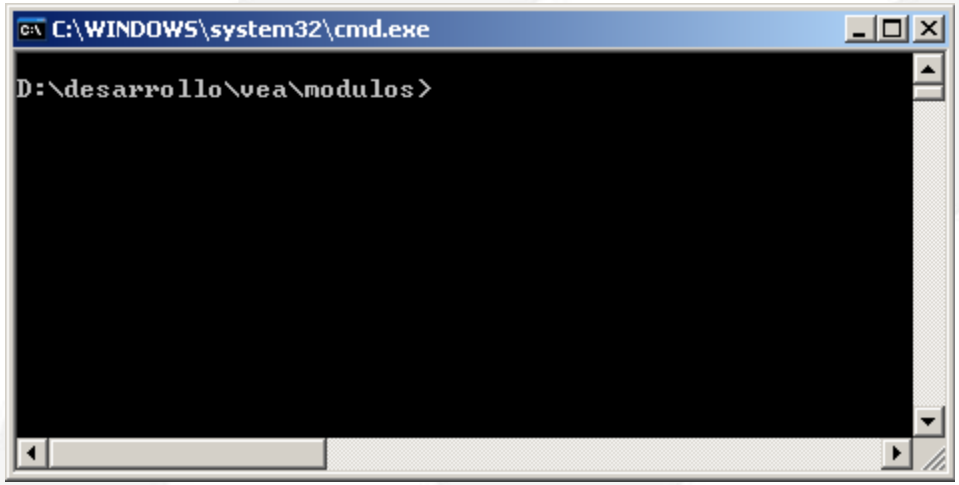
Como el cliente del servicio web se incluye de serie en VEA y no puede ser modificado, la implementación **debe regirse por el fichero wsdl distribuido**.

Componente numerador

Generación de código fuente a partir del descriptor del servicio web

Previa a la implementación del código fuente que obtendrá el número de expediente asociado a un determinado trámite, es necesario disponer de las clases necesarias para desplegar el servicio web asociado a un componente numerador.

Para ello, VEA provee un descriptor de servicio web (en adelante WSDL) en la ruta [CD_INSTALACION]/Recursos_desarrollo/Numerador/Software_base/GeneradorNumeradorImpl.wsdl, a partir del cual se puede generar el código fuente necesario para el despliegue de un componente numerador.



Componente numerador

Generación de código fuente a partir del descriptor del servicio web

Una vez situado en el espacio de trabajo se debe ejecutar el siguiente comando MAVEN para la creación del proyecto web asociado al componente numerador.

mvn archetype:generate -DgroupId=es.juntadeandalucia.vea -DartifactId=numerador -DarchetypeArtifactId=maven-archetype-webapp -Dversion=2.4.1 -Darchetype.encoding=UTF-8 -DpackageName=es.juntadeandalucia.vea

El comando anterior generará en el espacio de trabajo el proyecto "numerador".

A continuación se debe acceder al proyecto "numerador" y **editar el fichero pom.xml**, en la sección <build> se incluirá el plugin **axistools-maven-plugin**, previa configuración del elemento <url></url>, el cual contendrá la ruta hacia el fichero GeneradorNumeradorImpl.wsdl, con el descriptor del servicio web del componente numerador.

(Se muestra a continuación la sección del fichero pom.xml descrita)

Componente numerador

Generación de código fuente a partir del descriptor del servicio web

```
<plugins>
  <plugin>
    <groupId>org.codehaus.mojo</groupId>
    <artifactId>axistools-maven-plugin</artifactId>
    <version>1.4</version>
    <configuration>
      <urls>
        <url>file:///
[CD_INSTALACION]/Modulos/Numerador/Software_base/GeneradorNumeradorImpl.wsdl</
url>
      </urls>
      <packageSpace>es.juntadeandalucia.vea.${project.artifactId}</packageSpace>
      <serverSide>true</serverSide>
    </configuration>
    <executions>
      <execution>
        <goals>
          <goal>wsdl2java</goal>
        </goals>
      </execution>
    </executions>
  </plugin>
</plugins>
```

Componente numerador

Generación de código fuente a partir del descriptor del servicio web

Así mismo, en la sección <dependencies> se deben incluir las siguientes dependencias:

```
<dependency>
        <groupId>axis</groupId>
        <artifactId>axis</artifactId>
        <version>1.4</version>
</dependency>
<dependency>
        <groupId>org.apache.axis</groupId>
        <artifactId>axis-jaxrpc</artifactId>
        <version>1.4</version>
</dependency>
<dependency>
        <groupId>org.apache.axis</groupId>
        <artifactId>axis-saaj</artifactId>
        <version>1.4</version>
</dependency>
```

Una vez configurado el pom.xml del proyecto “numerador” se debe ejecutar el siguiente comando Maven, que generará el código fuente necesario para la implementación del servicio web de un componente numerador:

```
mvn clean compile
```

Componente numerador

Adecuación del código fuente

Una vez realizados los pasos descritos en el punto anterior, el código fuente generado se localizará en la siguiente carpeta
del proyecto:
target\generated-sources\axistools\wsdl2java
Se ejecutaran los siguientes comandos desde la consola del sistema, para reorganizar el código fuente generado en las rutas adecuadas.

```
cd numerador
mkdir src\main\java
mkdir src\main\webapp\WEB-INF\GeneradorNumeradorImplService\es\juntadeandalucia\vea\numerador\impl
move target\generated-sources\axistools\wsdl2java\es\. src\main\java\.
copy "[CD_INSTALACION]\Modulos\Numerador\Software_base\server-config.wsdd" src\main\webapp\WEB-INF\.
copy /Y "[CD_INSTALACION]\Modulos\Numerador\Software_base\web.xml" src\main\webapp\WEB-INF\.
move src\main\java\es\juntadeandalucia\vea\numerador\deploy.wsdd src\main\webapp\WEB-INF\
GeneradorNumeradorImplService\es\juntadeandalucia\vea\numerador\impl\.
move src\main\java\es\juntadeandalucia\vea\numerador\undeploy.wsdd src\main\webapp\WEB-INF\
GeneradorNumeradorImplService\es\juntadeandalucia\vea\numerador\impl\.
rd /S /Q target
```

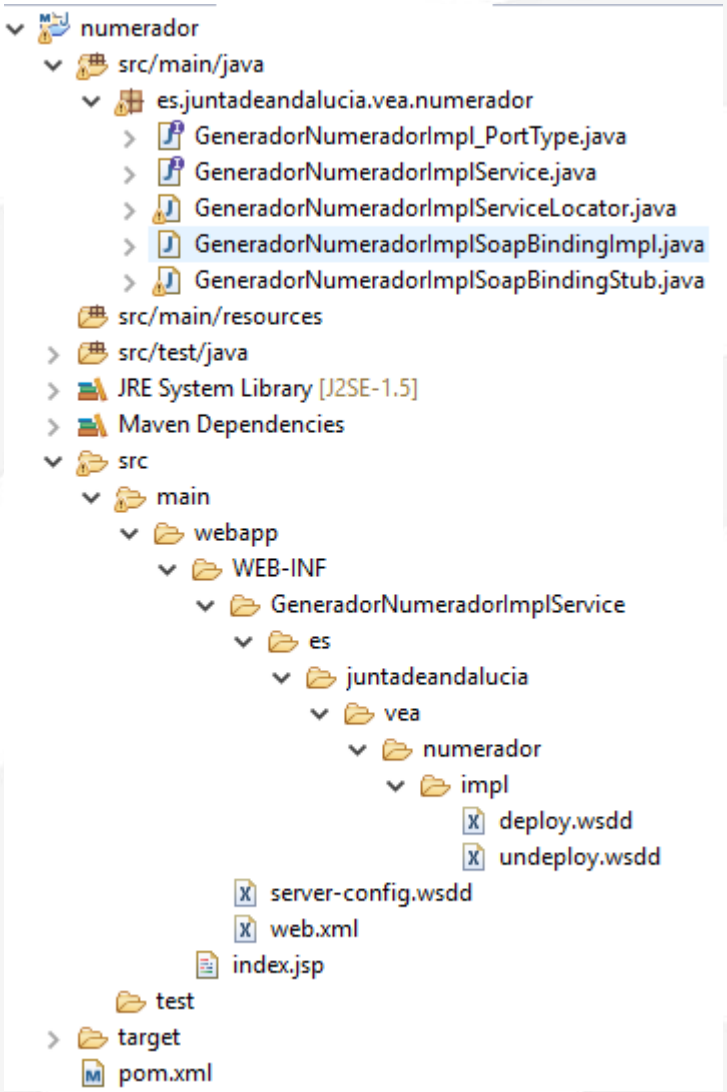
NOTA: Los comandos mkdir, move, copy y rd son propios de un entorno WINDOWS, en otros sistemas operativos se
tendrán que emplear los comandos equivalentes.

Componente numerador

Adecuación del código fuente

A continuación, se editará el fichero pom.xml y **se eliminará el plugin axistools-maven-plugin** de la sección <build>, a partir de este momento no es necesario dicho plugin puesto que el código fuente ya se ha generado.

Por último se importará en eclipse el proyecto como proyecto maven ya existente y automáticamente se creará el proyecto web.



Componente numerador

Implementación del servicio web

En la estructura de paquetes del proyecto "numerador", se puede observar la existencia del paquete es.juntadeandalucia.vea.numerador, en él se encuentra la clase GeneradorNumeradorImplSoapBindingImpl, la cual contiene el método **public java.lang.String generar(java.lang.String xmlInformacionEntrega) throws java.rmi.RemoteException**, este método será el que un usuario desarrollador deberá implementar con la lógica necesaria para generar y devolver a VEA el número de expediente.

El método recibe de VEA un objeto String con la información asociada a la entrega en estructura XML y **debe devolver un objeto de tipo String con el número de expediente generado**. En caso de error, el servicio web devolverá a VEA un objeto null, lo que implica que VEA no cree el expediente en Trew@ y se desencadene la gestión de errores que permitirá en un momento posterior la creación del expediente desde la consola de administración.

El objeto XML que recibe como entrada, contiene los datos de la entrega presentada por el usuario según el siguiente formato.

Componente numerador

Formato XML datos de la entrega

```xml
<?xml version="1.0" encoding="UTF-8"?>
<datos-vea>
    <entrega id-entrega="8887" id-fase-trewa="FASE INICIO EXPEDIENTE" id-tarea-trewa="NOMBRE TAREA SUBSANACION" transicion="TRANSICIÓN A EJECUTAR">
        <datos-tramitador>
            <JNDI>TREWA_JNDI</JNDI>
            <sistema>SISTEMA_TREWA</sistema>
            <procedimiento>COD_PROC_TREWA</procedimiento>
            <unidad-organica>COD_UNIDAD_ORGANICA</unidad-organica>
            <tipo-expediente>TIPO_EXP_PROCEDIMIENTO</tipo-expediente>
        </datos-tramitador>
        <asiento-aries>NUM REGISTRO ARIES</asiento-aries>
        <fecha>16/10/2017</fecha>
        <modoEntrega>TELEMÁTICA</modoEntrega>
        <nif-presentador>00000000T</nif-presentador>
        <formularios>
            <formulario id-formulario="COD_PROCESA" correcto="true" firmado="false" orden-formulario="2" tipo-documento="TIPO_DOC_TREWA_FORM1">
                <campo>
                    <nombre>SOLI1_APELLIDO1</nombre>
                    <valor>ESPAÑOL</valor>
                </campo>
                <campo>
                    <nombre>SOLI1_APELLIDO2</nombre>
                    <valor>ESP</valor>
                </campo>
                <campo>
                    <nombre>SOLI1_NOMBRE</nombre>
                    <valor>JUAN</valor>
                </campo>
            </formulario>
        </formularios>
        <documentos-adjuntos>
            <docAdjunto cod-doc="TIPO_DOC_TREWA_ADJ" incorporado="true" firmado="false" autorizacion="false" autorizacionSCSP="false">
            </docAdjunto>
        </documentos-adjuntos>
    </entrega>
</datos-vea>
```

Componente numerador

Implementación del servicio web

Ejemplo de implementación

```
/**
 * {@inheritDoc}
 */
public java.lang.String generar(java.lang.String xmlInformacionEntrega) throws java.rmi.RemoteException {
    LOGGER.info("\n\n1) EJECUTANDO NUMERADOR DE EXPEDIENTES EXTERNO");
    String numeroExp = null;
    try {
        if (StringUtils.isNotEmpty(xmlInformacionEntrega)) {
            numeroExp = generarNumeroEntrega(xmlInformacionEntrega);
        } else {
            LOGGER.warn("Se recibe objeto de VEA nulo. Se genera número de expediente con la fecha.");
            final SimpleDateFormat sdf = new SimpleDateFormat("yyyyMMddHHmmss");
            numeroExp = sdf.format(new Date());
        }
        // Se realiza el tratamiento del número de expediente
        final int longitudNum = numeroExp.length();
        int longNumeracionExp = LONG_NUM_EXP_ENI - longitudNum;

        // Se completa con 0 a la izquierda el tamaño minimo del expediente
        numeroExp = numeroExp + StringUtils.leftPad(String.valueOf(1L), longNumeracionExp, '0');
    } catch (final Exception e) {
        LOGGER.error("Ha ocurrido un error al generar el número de expediente externo.", e);
    }
    return numeroExp;
}
```

Componente numerador

Implementación del servicio web

```
DocumentBuilderFactory domFactory = DocumentBuilderFactory.newInstance();
domFactory.setNamespaceAware(true);
DocumentBuilder builder = domFactory.newDocumentBuilder();
Document doc = builder.parse(new InputSource(new StringReader(xmlInformacionEntrega)));

NodeList nodeList;
Node node;
Element elemento;

// Se obtiene el nombre del procedimiento
String procedimiento = "";
nodeList = doc.getElementsByTagName("procedimiento");
node = nodeList.item(0);
if (node.getNodeType() == Node.ELEMENT_NODE) {
    elemento = (Element) node;
    procedimiento = elemento.getFirstChild().getNodeValue();
}

// Se obtiene el registro @ries
String registro = "";
nodeList = doc.getElementsByTagName("asiento-aries");
node = nodeList.item(0);
if (node.getNodeType() == Node.ELEMENT_NODE) {
    elemento = (Element) node;
    registro = elemento.getFirstChild().getNodeValue();
}

StringBuilder num = new StringBuilder(procedimiento);
num.append("/");
num.append(registro);

if (num.length() > LONG_NUM_EXP_ENI) {
    num.setLength(LONG_NUM_EXP_ENI);
}

numero = num.toString();
LOGGER.info("\n\n\nGenerado el número de expediente: " + numero + "\n\n\n");
```

Componente numerador

Implementación del servicio web

Junto con el cd de la aplicación se distribuye un proyecto de ejemplo, que puede ser usado como plantilla. El proyecto contiene el código básico (junto con su fichero pom.xml) para comenzar el servicio web.

Es la forma más rápida y sencilla de implementar el servicio, ya que contiene las clases generadas y el fichero pom.xml preparado.

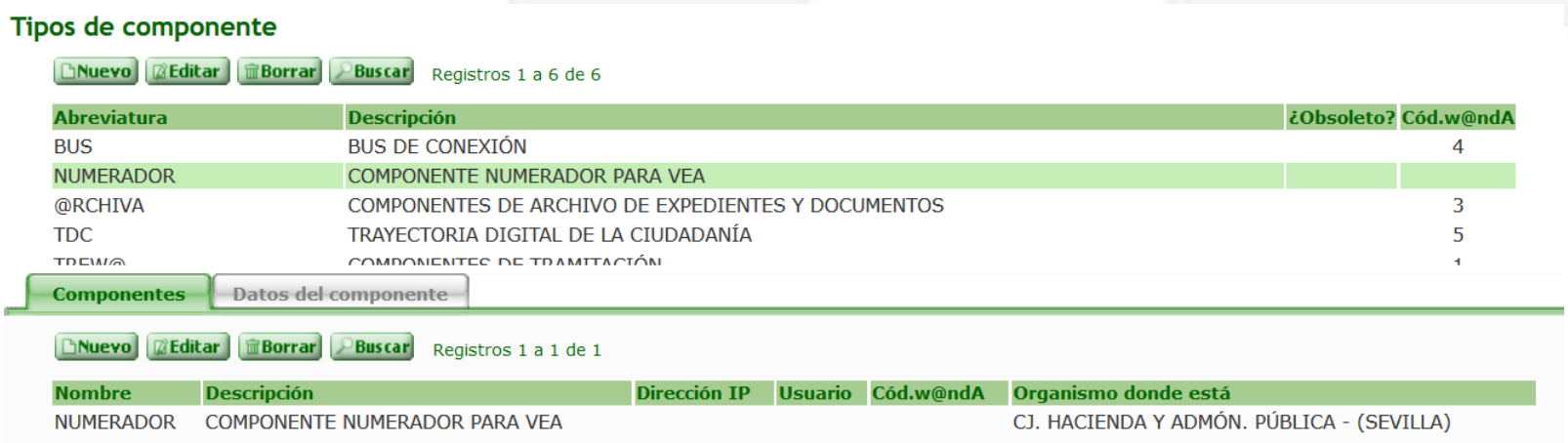
Para usar este proyecto, solo es necesario modificar el fichero pom.xml para incluir las dependencias necesarias, modificar el artifactId y groupId e implementar el método indicado previamente con la lógica correspondiente.

Componente numerador

Configuración del componente numerador en Trew@

Para la generación de números de expediente, VEA se basa en los componentes numerador configurados en el motor de tramitación Trew@, por lo que dicho componente debe ser configurado desde la administración de Trew@.

En primer lugar, desde el panel de administración de Trew@ se debe acceder a la Configuración Componentes y crear en el caso de que no exista, un nuevo tipo de componente llamado "NUMERADOR".



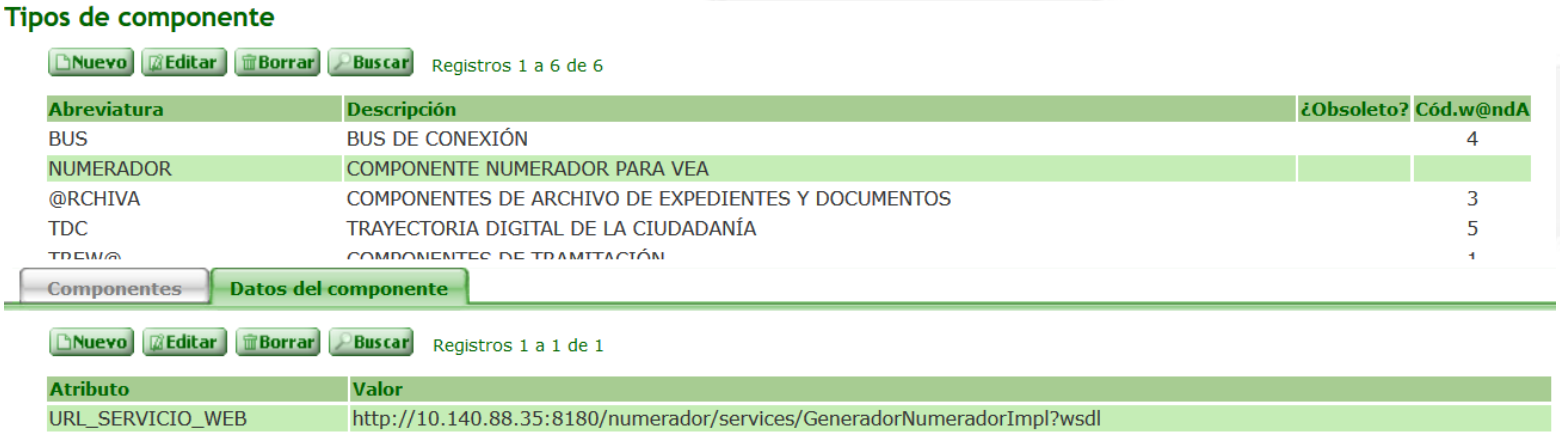
Abreviatura	Descripción	¿Obsoleto?	Cód.w@ndA
BUS	BUS DE CONEXIÓN		4
NUMERADOR	COMPONENTE NUMERADOR PARA VEA		
@RCHIVA	COMPONENTES DE ARCHIVO DE EXPEDIENTES Y DOCUMENTOS		3
TDC	TRAYECTORIA DIGITAL DE LA CIUDADANÍA		5

Nombre	Descripción	Dirección IP	Usuario	Cód.w@ndA	Organismo donde está
NUMERADOR	COMPONENTE NUMERADOR PARA VEA				CJ. HACIENDA Y ADMÓN. PÚBLICA - (SEVILLA)

Componente numerador

Configuración del componente numerador en Trew@

A continuación, se debe crear un nuevo Componente con un Atributo llamado URL_SERVICIO_WEB cuyo valor se corresponda con la ruta al servicio web del componente numerador correspondiente.

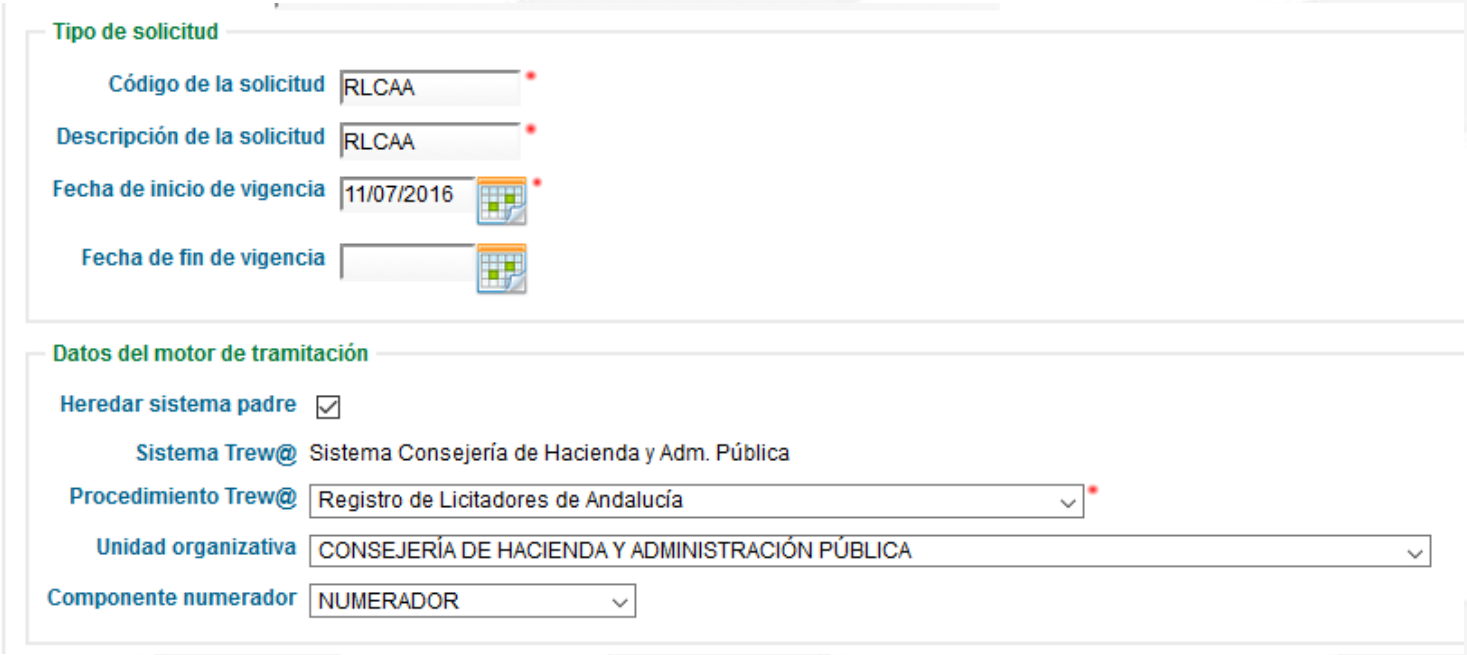


Componente numerador

Configuración del componente numerador en VEA

Una vez creado y configurado el componente numerador en Trew@, será necesario indicar en la administración de VEA, que para un procedimiento determinado, se debe usar dicho numerador en lugar del que VEA provee por defecto.

Para ello se accederá a *Gestión de trámites -> Seleccionar trámite y editar -> En el seleccionable "Componente numerador" elegir el que se haya configurado en Trew@*



Ventanilla Electrónica de la Administración (VEA)

- I Introducción
- II Conexión con servicios externos
- III Componente numerador
- **IV Procesamiento específico**
- V Validación específica
- VI Visibilidad
- VII Requisitos a subsanar

Procesamiento específico

Introducción

Durante la presentación de una entrega, VEA ejecuta una serie de operaciones fijas, pero en dos puntos de su ejecución invoca a servicios externos: la generación del componente numerador y el procesamiento específico.

A través del servicio web de procesamiento específico se puede añadir funcionalidad extra al ciclo de creación del expediente de forma transparente y desacoplada.

En el punto en el que se invoca al procesamiento específico el expediente ya existe en Trew@, por lo que es posible realizar modificaciones sobre él, recuperando dicho expediente a partir del número de expediente proporcionado en el xml de la petición.

Procesamiento específico

Generación del código fuente a partir del descriptor del servicio web

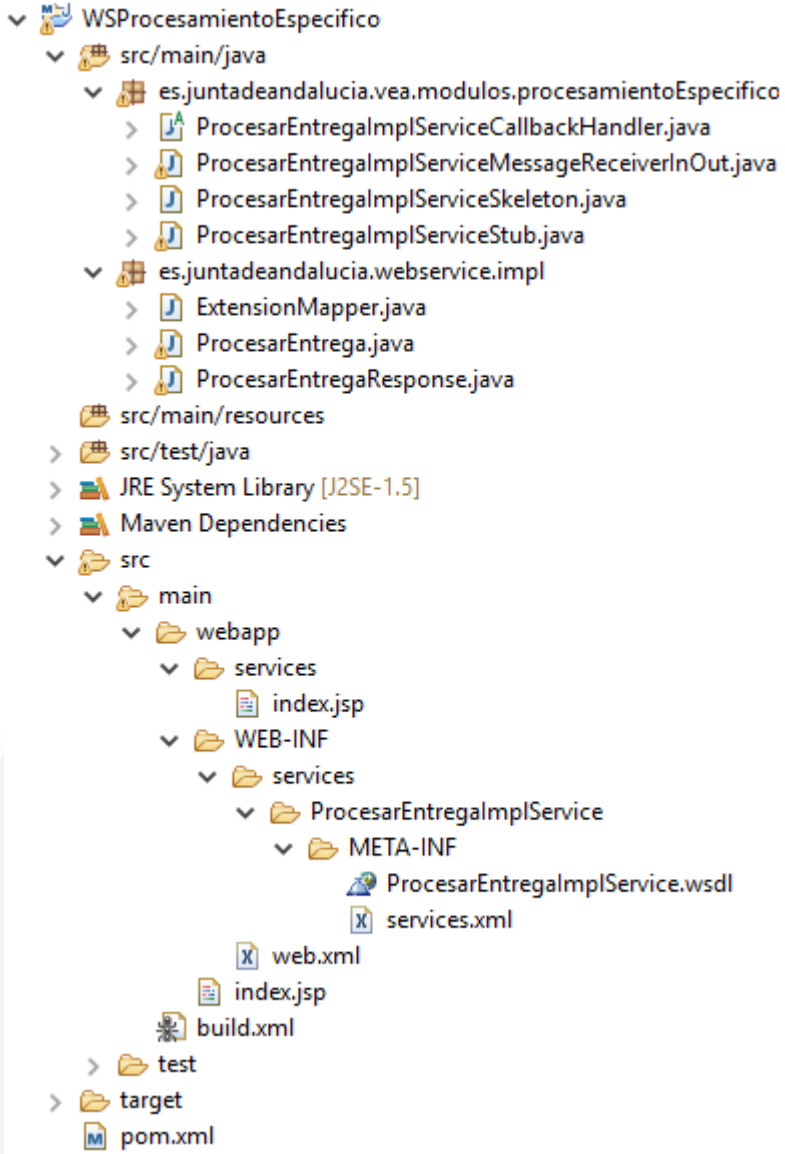
La generación y adecuación del código fuente es igual a los servicios web de procesamiento específico y numerador salvo que **en este caso la versión de Axis utilizada es Axis2 v1.5.4**

En este caso, el WSDL de referencia se encuentra en la ruta:
[CD_INSTALACION]/Recursos_desarrollo/Procesamiento_Especifico/ProcesarEntregaImpl.wsdl

Para ilustrar este caso, se parte directamente del proyecto ya creado que se proporciona en el CD de VEA.

Procesamiento específico

Estructura del proyecto Java



Procesamiento específico

Implementación del servicio web

En la estructura de paquetes del proyecto "WSProcesamientoEspecifico", se puede observar la existencia del paquete es.juntadeandalucia.vea.modulos.procesamientoEspecifico, en él se encuentra la clase **ProcesarEntregaImplServiceSkeleton**, la cual contiene el método procesarEntrega, este método será el que un usuario desarrollador deberá implementar con la lógica necesaria en el procesamiento específico de la entrega.

El método recibe de VEA un objeto ProcesarEntrega y dentro un String con la información asociada a la entrega en estructura XML **igual a la del numerador** y debe devolver un objeto de tipo ProcesarEntregaResponse con un String dentro indicando si se ha realizado el procesamiento específico con éxito (*CORRECTO*)

En caso de error, el servicio web devolverá a VEA la cadena de texto (*ERROR*) lo que implica que VEA detenga el proceso de alta de expediente y se desencadene la gestión de errores que permitirá en un momento posterior la continuación del alta del expediente desde la consola de administración.

En caso de error el expediente se habrá creado en Trew@ pero no se habrá lanzado la indexación ni la gestión en Notific@.

Procesamiento específico

Implementación del servicio web

Ejemplo de implementación

```
/**
 * Permite interactuar con el sistema de logs.
 */
private static final Log LOGGER = LogFactory.getLog(ProcesarEntregaImplServiceSkeleton.class);

/**
 * Indica si todo se ha ejecutado de forma correcta.
 */
public static final String BIEN_RECIBIDO = "CORRECTO";

/**
 * Indica si ha ocurrido algún fallo en la ejecución.
 */
public static final String MAL_RECIBIDO = "ERROR";

/**
 * Auto generated method signature
 *
 * @param procesarEntrega
 */

public es.juntadeandalucia.webservice.impl.ProcesarEntregaResponse procesarEntrega(
        es.juntadeandalucia.webservice.impl.ProcesarEntrega procesarEntrega) {
    LOGGER.info("\n\n2) EJECUTANDO PROCESAMIENTO ESPECÍFICO");
    ProcesarEntregaResponse respuesta = new ProcesarEntregaResponse();
    try {
        if (StringUtils.isNotEmpty(procesarEntrega.getXmlEntrega())) {
            respuesta.setProcesarEntregaReturn(procesarDatosEntrega(procesarEntrega.getXmlEntrega()));
        } else {
            LOGGER.error(
                    "No se han recibido datos de la entrega de VEA. NO se ejecuta el procesamiento específico.");
        }
    } catch (final Exception e) {
        LOGGER.error("Ha ocurrido un error al ejecutar el procesamiento específico.", e);
    }
    return respuesta;
}
```

Procesamiento específico

Implementación del servicio web

```
// Se obtienen los documentos adjuntos
nodeList = doc.getElementsByTagName("documentos-adjuntos");
node = nodeList.item(0);
if (node.getNodeType() == Node.ELEMENT_NODE) {
    elemento = (Element) node;
    NodeList listaDocumentos = elemento.getChildNodes();
    int length = listaDocumentos.getLength();
    resumen.append("\n\t· Existen ");
    resumen.append(length);
    resumen.append(" documentos adjuntos en la entrega y sus datos son:");
    for (int i = 0; i < length; i++) {
        if (listaDocumentos.item(i).getNodeType() == Node.ELEMENT_NODE) {
            Element docAdj = (Element) listaDocumentos.item(i);
            if ("docAdjunto".equals(docAdj.getNodeName())) {
                String tipoDocTr = docAdj.getAttribute("cod-doc");
                String firmado = docAdj.getAttribute("firmado");
                String incorporado = docAdj.getAttribute("incorporado");
                String csv = docAdj.getAttribute("csv-doc");
                String eni = docAdj.getAttribute("eni-doc");

                if (BooleanUtils.toBoolean(incorporado)) {
                    resumen.append("\n\t\t- ");
                    resumen.append(tipoDocTr);
                    resumen.append("\n\t\t\t# Firmado: ");
                    resumen.append(firmado);
                    resumen.append("\n\t\t\t# CSV: ");
                    resumen.append(csv);
                    resumen.append("\n\t\t\t# Identificador ENI: ");
                    resumen.append(eni);
                }
            }
        }
    }
}
result = BIEN_RECIBIDO;
LOGGER.info("Se han obtenido los siguientes datos de la entrega: " + resumen.toString());
```

Procesamiento específico

Configuración del procesamiento específico en VEA

Para configurar el procesamiento específico en VEA, se accederá a la consola de administración y se ejecutará la siguiente secuencia *Gestión de trámites -> Seleccionar trámite y editar -> Ver convocatorias -> Seleccionar convocatoria y editar -> Seleccionar entrega y editar*

En este punto se debe establecer el procesamiento mediante servicio web, y establecer del mismo modo la Ruta donde se ha desplegado el Servicio Web:

Ventanilla Electrónica de la Administración (VEA)

- I Introducción
- II Conexión con servicios externos
- III Componente numerador
- IV Procesamiento específico
- V Validación específica
- VI Visibilidad
- VII Requisitos a subsanar

Validación específica

Introducción

Para que VEA permita firmar y/o presentar una entrega deben cumplirse ciertas condiciones, las cuales son fijas (todos los documentos y formularios obligatorios deben estar cumplimentados, han tenido que indicarse todos los datos de los interesados, se han debido realizar todas las firmas requeridas, etc).

Si se desea añadir una condición específica distinta que impida la firma y/o presentación de la entrega se debe recurrir al servicio web de validación específica.

Cuando el usuario intente realizar o bien la tarea de firma o bien la de presentación de la entrega, VEA puede invocar a un servicio web del que obtiene la lista de motivos por los que la entrega no puede ser firmada y/o presentada.

La generación y adecuación del código fuente es igual a los servicios web de procesamiento específico y numerador salvo que **en este caso la versión de Axis utilizada es Axis2 v1.5.4**

Su implementación es distinta, ya que en este caso el servicio web no devuelve un String, **sino que devuelve un objeto que contiene todos los motivos por los que la entrega no puede ser presentada.** No obstante, el servicio web **recibe el mismo parámetro**, esto es, el xml con la información de la entrega.

Validación específica

Generación del código fuente a partir del descriptor del servicio web

El fichero .wsdl que debe usarse para la generación de la implementación se encuentra en la ruta:
[CD_INSTALACION]/Recursos_desarrollo/ValidacionEntrega/WSValidarEntrega.wsdl

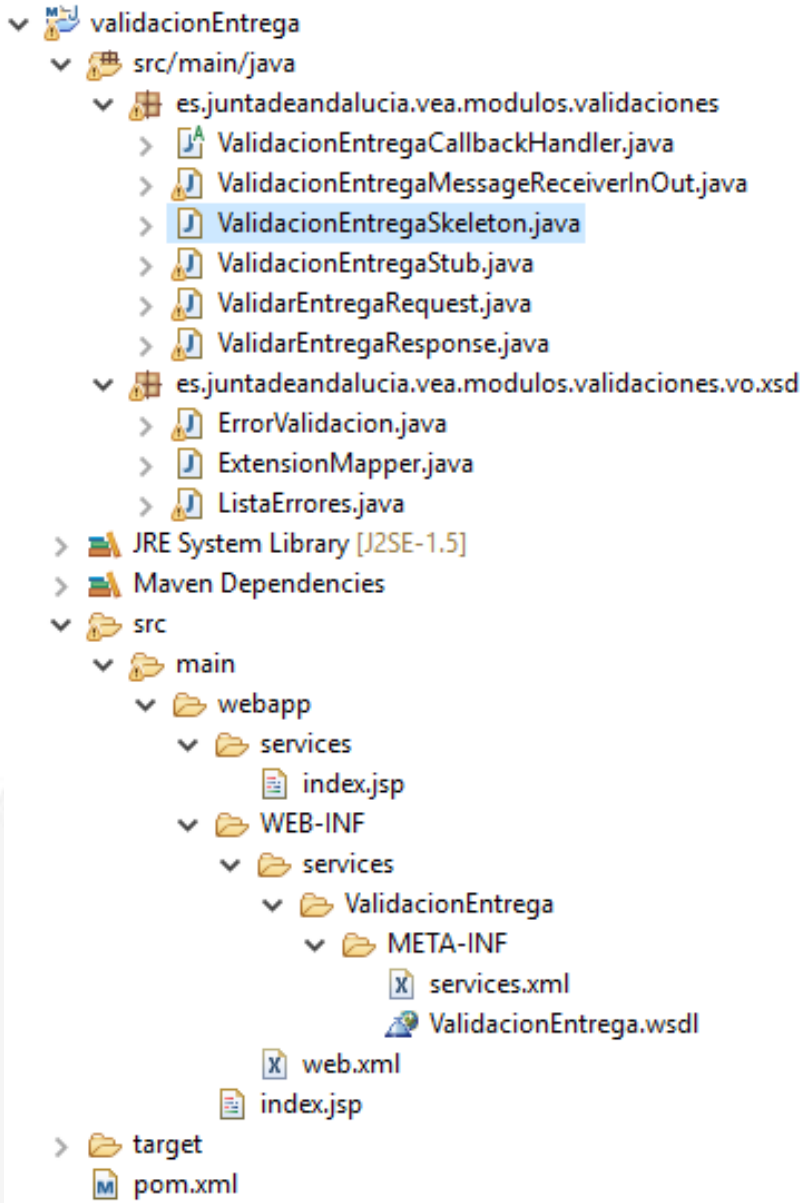
Para crear el espacio de trabajo nuevo se debe ejecutar el comando:

*mvn archetype:generate -DgroupId=*es.juntadeandalucia.vea.modulos.validacionEntrega *-DartifactId=*WSValidacionEntrega *-DarchetypeArtifactId=maven-archetype-webapp -Dversion=2.4.1 -Darchetype.encoding=UTF-8 -DpackageName=****es.juntadeandalucia.vea.modulos.validaciones***

Para ilustrar este caso, se parte directamente del proyecto ya creado que se proporciona en el CD de VEA.

Validación específica

Estructura del proyecto Java



Validación específica

Implementación del servicio web

En la estructura de paquetes del proyecto "validacionEntrega", se puede observar la existencia del paquete es.juntadeandalucia.vea.modulos.validaciones, en él se encuentra la clase **ValidacionEntregaSkeleton**, la cual contiene el método **validarEntrega**, este método será el que un usuario desarrollador deberá implementar con la lógica necesaria para validar la entrega.

En el caso del servicio web de validación, el resultado difiere del resto de servicios web vistos hasta ahora (que devuelven una cadena de texto).

En este servicio web **es necesario construir los objetos que contienen la respuesta** (en este caso, los motivos por los que una entrega no es válida).

A continuación se muestra un ejemplo:

Validación específica

Implementación del servicio web

Ejemplo de implementación

```
/**
 * Permite interactuar con el sistema de logs.
 */
private static final Log LOGGER = LogFactory.getLog(ValidacionEntregaSkeleton.class);

/**
 * Auto generated method signature
 *
 * @param validarEntregaRequest
 */

public es.juntadeandalucia.vea.modulos.validaciones.ValidarEntregaResponse validarEntrega(
        es.juntadeandalucia.vea.modulos.validaciones.ValidarEntregaRequest validarEntregaRequest) {
    LOGGER.info("\n\n3) EJECUTANDO VALIDACIÓN DE LA ENTREGA");
    final ValidarEntregaResponse validar = new ValidarEntregaResponse();

    try {
        final String xmlEntrega = validarEntregaRequest.getXMLEntrega();
        if (StringUtils.isNotEmpty(xmlEntrega)) {
            validar.setListaErrores(comprobarEntrega(xmlEntrega));
        } else {
            LOGGER.error(
                    "No se han recibido datos de la entrega de VEA. NO se ejecuta la validación de la entrega.");
        }
    } catch (final Exception e) {
        LOGGER.error("Ha ocurrido un error al ejecutar la validación de la entrega.", e);
    }
    return validar;

}
```

Validación específica

Implementación del servicio web

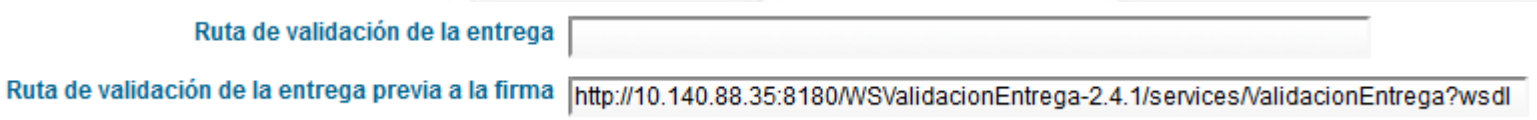
```
// Si se requiere comprobación de documentos se ejecuta
if (comprobarDocumento) {
    // Se recorren los documentos hasta obtener el que se quiere
    // comprobar
    nodeList = doc.getElementsByTagName("documentos-adjuntos");
    node = nodeList.item(0);
    if (node.getNodeType() == Node.ELEMENT_NODE) {
        elemento = (Element) node;
        NodeList listaDocumentos = elemento.getChildNodes();
        int length = listaDocumentos.getLength();
        for (int i = 0; i < length; i++) {
            if (listaDocumentos.item(i).getNodeType() == Node.ELEMENT_NODE) {
                Element el = (Element) listaDocumentos.item(i);
                if ("docAdjunto".equals(el.getNodeName())) {
                    String tipoDoc = el.getAttribute("cod-doc");
                    String incorporado = el.getAttribute("incorporado");
                    if ("ANEXOIII_RL".equals(tipoDoc)) {
                        if (!BooleanUtils.toBoolean(incorporado)) {
                            final ErrorValidacion errorWS = new ErrorValidacion();
                            errorWS.setCodigo("COD_ERROR");
                            errorWS.setTipoError("TIPO_ERROR");
                            errorWS.setDescripcion(
                                    "No se puede firmar la entrega hasta que no se incorpore el documento "
                                    + "'Anexo III: Certificado de no estar incurso en incompatibilidad "
                                    + "o prohibicion para contratar'");
                            errores.addErrorValidacion(errorWS);
                        }
                        break;
                    }
                }
            }
        }
    }
}
LOGGER.info("Se ha realizado la validación de la entrega.");
```

Validación específica

Configuración de la validación de la entrega en VEA

Para configurar la validación de la entrega en VEA, se accederá a la consola de administración y se ejecutará la siguiente secuencia *Gestión de trámites -> Seleccionar trámite y editar -> Ver convocatorias -> Seleccionar convocatoria y editar -> Seleccionar entrega y editar*

En este punto se debe establecer la Ruta donde se ha desplegado el Servicio Web, bien para la validación previa a la firma o bien para la validación previa a la presentación.



Ventanilla Electrónica de la Administración (VEA)

I Introducción

II Conexión con servicios externos

III Componente numerador

IV Procesamiento específico

V Validación específica

VI Visibilidad

VII Requisitos a subsanar

Visibilidad

Introducción

Todos los formularios y documentos de la entrega son visibles por defecto, por lo que el orden en el que se deben cumplimentar los formularios no tiene porqué cumplirse.

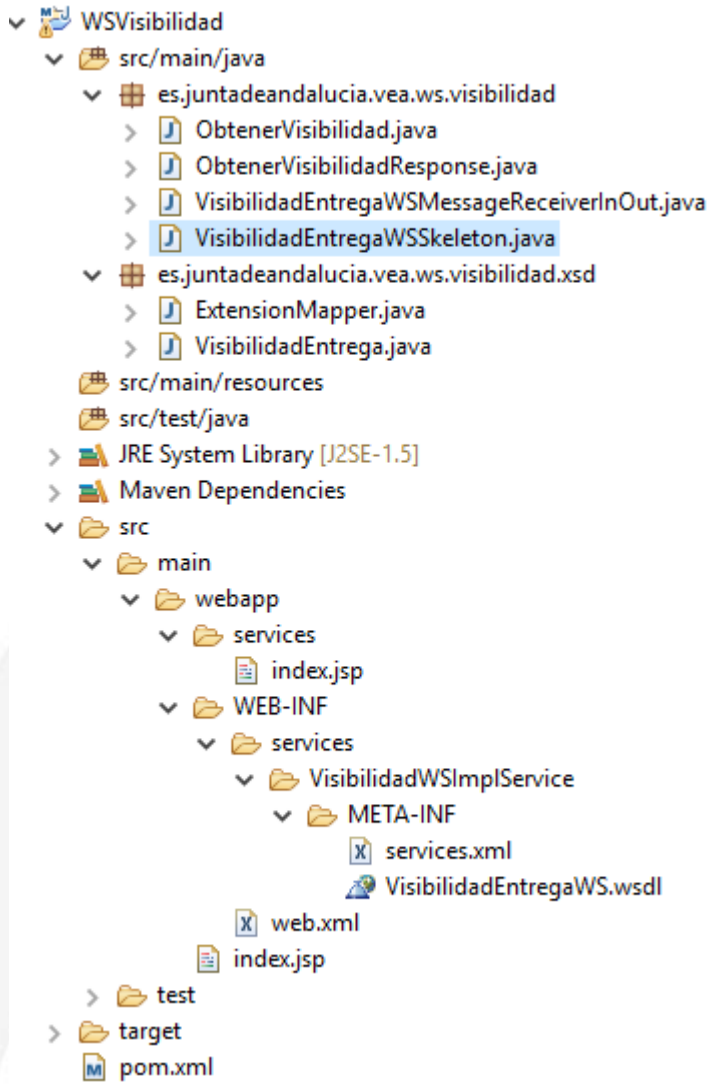
Cuando el orden en el que se rellenan los formularios es importante, o se desea controlar la visibilidad de un formulario o documento adjunto, se puede configurar un servicio web que indica la visibilidad del elemento.

La generación del código fuente es igual a la del servicio web de validación. El fichero .wsdl que debe usarse para la generación de la implementación se encuentra en la ruta:
[CD_INSTALACION]/Recursos_desarrollo/Visibilidad/VisibilidadEntregaWS.wsdl

Para crear el espacio de trabajo nuevo se debe ejecutar el comando:
*mvn archetype:generate -DgroupId=***es.juntadeandalucia.vea.modulos.visibilidad** *-DartifactId=***WSVisibilidad** *-DarchetypeArtifactId=maven-archetype-webapp -Dversion=2.4.1 -Darchetype.encoding=UTF-8 -DpackageName=****es.juntadeandalucia.vea.ws.visibilidad***

Visibilidad

Estructura del proyecto Java



Visibilidad

Implementación del servicio web

En la estructura de paquetes del proyecto "WSVisibilidad", se puede observar la existencia del paquete es.juntadeandalucia.vea.ws.visibilidad, en él se encuentra la clase **VisibilidadEntregaWSSkeleton**, la cual contiene el método **obtenerVisibilidad**, este método será el que un usuario desarrollador deberá implementar con la lógica necesaria para calcular la visibilidad de la entrega.

En el caso del servicio web de visibilidad, el resultado es un objeto al igual que en el servicio web de validación, por tanto también **es necesario construir los objetos que contienen la respuesta.**

A continuación se muestra un ejemplo:

Visibilidad

Implementación del servicio web

Ejemplo de implementación

```
/**
 * Implementación del servicio web de visibilidad.
 *
 * @param obtenerVisibilidad
 *            - Datos de la entrega.
 */
public es.juntadeandalucia.vea.ws.visibilidad.ObtenerVisibilidadResponse obtenerVisibilidad(
        final es.juntadeandalucia.vea.ws.visibilidad.ObtenerVisibilidad obtenerVisibilidad) {
    LOGGER.info("\n\n4) EJECUTANDO VISIBILIDAD DE LOS DOCUMENTOS DE LA ENTREGA");
    ObtenerVisibilidadResponse result = new ObtenerVisibilidadResponse();

    try {
        final String xmlEntrega = obtenerVisibilidad.getXmlEntrega();
        if (StringUtils.isNotEmpty(xmlEntrega)) {
            result.set_return(establecerVisibilidad(xmlEntrega));
        } else {
            LOGGER.error(
                    "No se han recibido datos de la entrega de VEA. NO se ejecuta la visibilidad de la entrega.");
        }
    } catch (final Exception e) {
        LOGGER.error("Ha ocurrido un error al ejecutar la validación de la entrega.", e);
    }
    return result;
}
```

Visibilidad

Implementación del servicio web

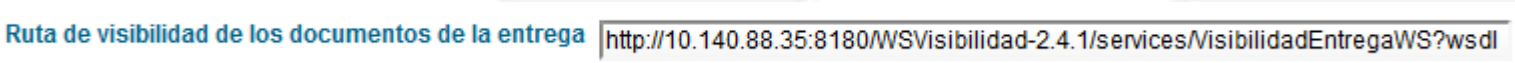
```
// Se recorren los formularios hasta obtener el que se quiere
// comprobar
nodeList = doc.getElementsByTagName("formularios");
node = nodeList.item(0);
if (node.getNodeType() == Node.ELEMENT_NODE) {
    elemento = (Element) node;
    NodeList listaFormularios = elemento.getChildNodes();
    int length = listaFormularios.getLength();
    for (int i = 0; i < length; i++) {
        if (listaFormularios.item(i).getNodeType() == Node.ELEMENT_NODE) {
            Element el = (Element) listaFormularios.item(i);
            if ("formulario".equals(el.getNodeName())) {
                String codProcesa = el.getAttribute("id-formulario");
                if ("RLCAA".equals(codProcesa)) {
                    NodeList campos = el.getChildNodes();
                    int tam = campos.getLength();
                    for (int j = 0; j < tam; j++) {
                        if (campos.item(j).getNodeType() == Node.ELEMENT_NODE) {
                            Element campo = (Element) campos.item(j);
                            if (campo.getFirstChild() != null
                                    && campo.getFirstChild().getNextSibling() != null
                                    && campo.getFirstChild().getNextSibling().getFirstChild() != null
                                    && campo.getLastChild() != null
                                    && campo.getLastChild().getPreviousSibling() != null
                                    && campo.getLastChild().getPreviousSibling()
                                            .getFirstChild() != null) {
                                String nombreCampo = campo.getFirstChild().getNextSibling()
                                        .getFirstChild().getNodeValue();
                                String valorCampo = campo.getLastChild().getPreviousSibling()
                                        .getFirstChild().getNodeValue();
                                if ("DOCU_FICHA".equals(nombreCampo)
                                        && BooleanUtils.toBoolean(valorCampo)) {
                                    visibilidad[0].setVisible(true);
                                }
                            }
                        }
                    }
                }
```

Visibilidad

Configuración de la visibilidad de la entrega en VEA

Para configurar la visibilidad de la entrega en VEA, se accederá a la consola de administración y se ejecutará la siguiente secuencia *Gestión de trámites -> Seleccionar trámite y editar -> Ver convocatorias -> Seleccionar convocatoria y editar -> Seleccionar entrega y editar*

En este punto se debe establecer la Ruta donde se ha desplegado el Servicio Web para la visibilidad.



Ventanilla Electrónica de la Administración (VEA)

- I Introducción
- II Conexión con servicios externos
- III Componente numerador
- IV Procesamiento específico
- V Validación específica
- VI Visibilidad
- VII Requisitos a subsanar

Requisitos a subsanar

Introducción

A lo largo de la vida administrativa de un expediente, es posible que el interesado necesite realizar alguna entrega de subsanación en la que solvente posibles deficiencias o errores en la documentación aportada en la creación del expediente.

Desde el detalle del expediente, VEA ofrece esta opción a los interesados de los expedientes, siempre que el mismo se encuentre en fase de subsanación y siempre que existan requisitos a subsanar. Es precisamente esta comprobación de requisitos a subsanar la que realiza VEA conectándose a un servicio web externo de requisitos.

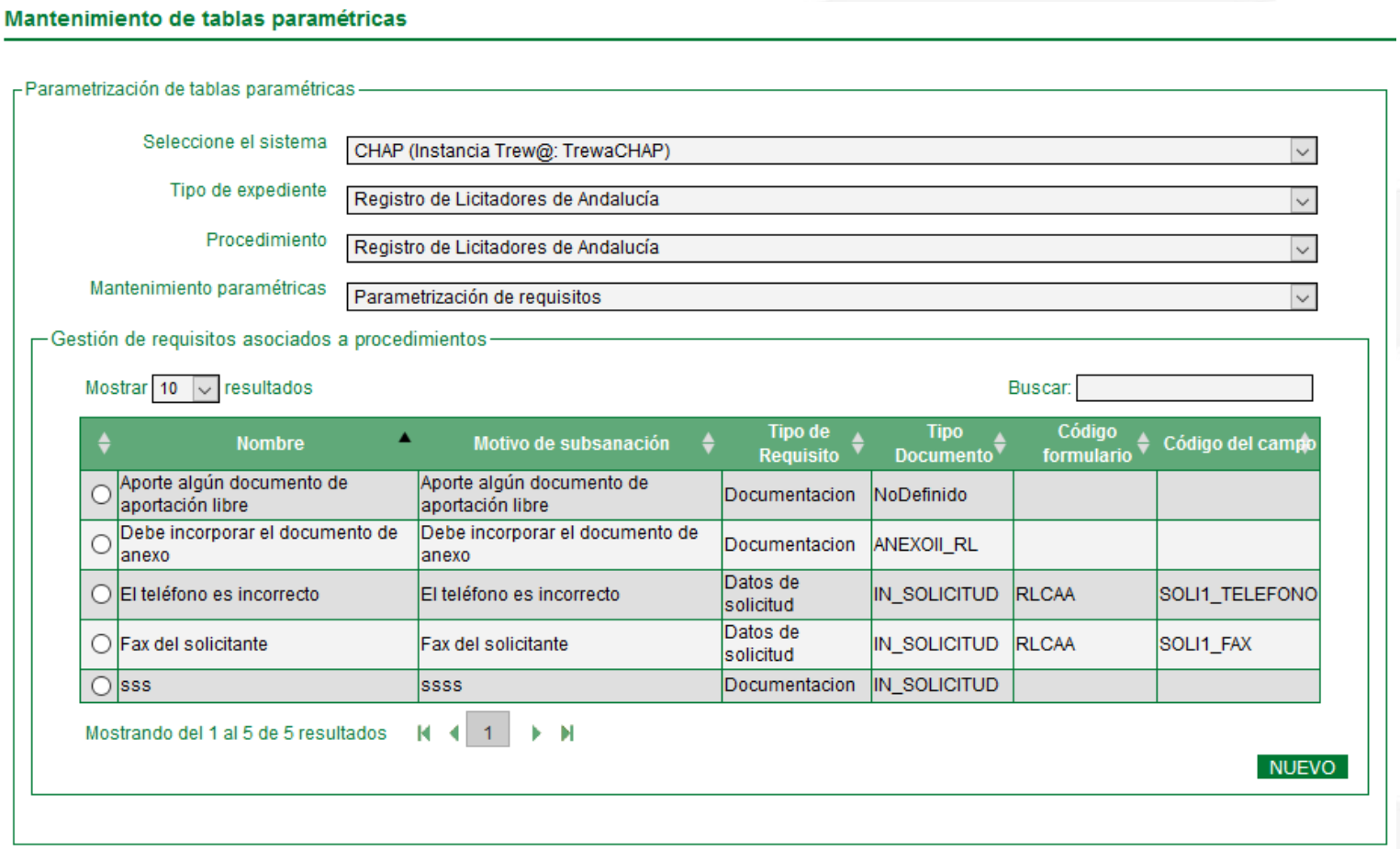
La integración para la gestión de requerimientos sobre expedientes, está perfectamente resuelta a través de la funcionalidad que proporciona la **Plataforma de Tramitación W@ndA**, utilizando el **módulo de requerimientos**.

Realizando una parametrización mínima en el módulo de administración de la plataforma, será posible configurar todo lo necesario para gestionar los posibles requisitos a subsanar sobre un expediente y de forma automática ofrecer esa información a VEA siempre que lo requiera.

Requisitos a subsanar

Gestión de requisitos en PTw@ndA

Como usuario administrador de PTw@ndA, utilizando la gestión de tablas de configuración de parámetros, se pueden editar y crear nuevos requisitos asociados a procedimientos.

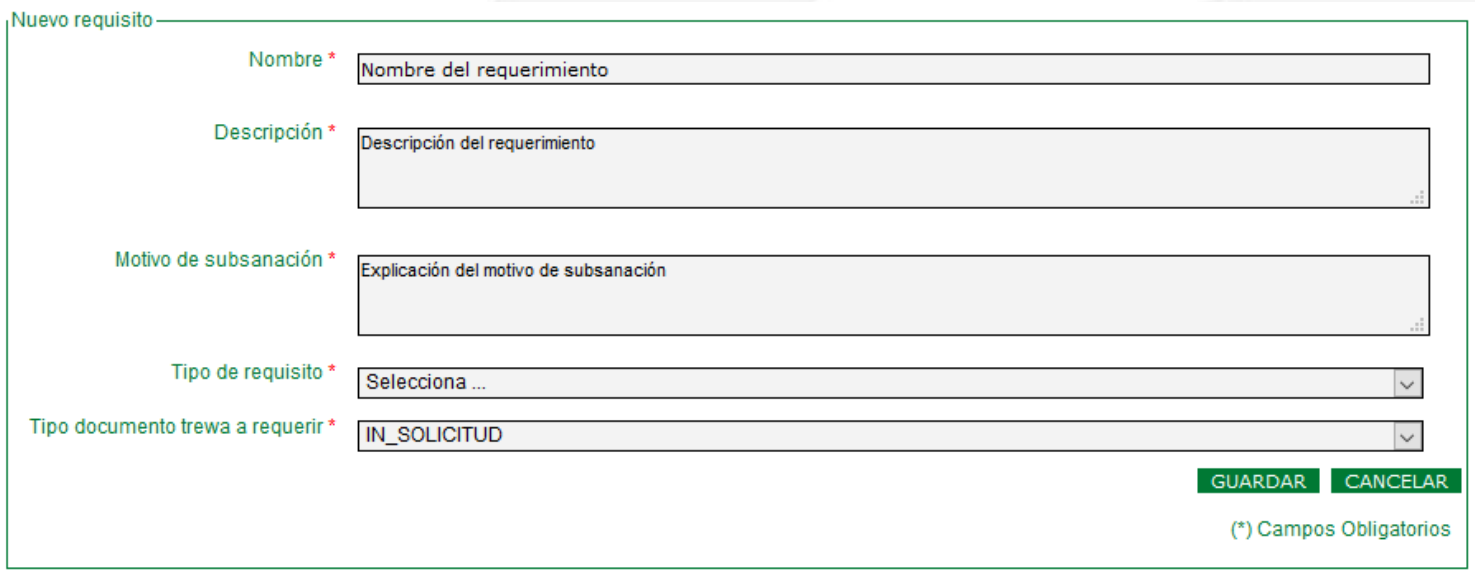


Requisitos a subsanar

Configuración manual de requisitos

Para cada requisito que se defina en el sistema se podrán parametrizar los siguientes datos:

- **Nombre**: Nombre del requerimiento.
- **Descripción**: Descripción del requerimiento.
- **Motivo** de subsanación: Explicación del motivo de subsanación.
- **Tipo de requisito**: Tipo de requisito a elegir entre Datos de solicitud o Documentación
- **Tipo de documento Trew@ a requerir**: Documento de Trew@ a requerir.

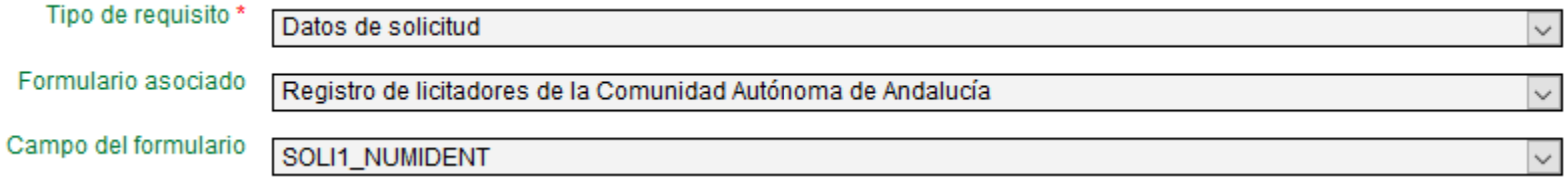


Requisitos a subsanar

Configuración manual de requisitos

Para los tipos de requisito **Documentación**, se requerirá subsanación sobre un documento incorporado, mientras que para los requisitos **Datos de solicitud** se requerirá subsanación sobre casillas concretas de un formulario web de solicitud.

Para este último caso será necesario además indicar el **formulario** dentro del motor de formularios Formul@ y el **campo** correspondiente.



Requisitos a subsanar

Configuración automática de requisitos

El sistema también permite la configuración de requisitos que calculan de forma automática si es necesario subsanar algún dato del expediente. Esta operación se puede realizar de dos formas diferentes.

Mediante una sentencia SQL: Si se introduce una sentencia SQL se puede realizar una comprobación sobre cualquier campo del modelo de datos. La respuesta debe ser una tupla de dos columnas, el primer valor debe ser '0' si el requisito se ha identificado como requisito a exigir o '1' en caso contrario. El segundo valor devuelve el motivo de subsanación si procede.

Dentro de la sentencia SQL es posible indicar el identificador del expediente como parámetro, de la siguiente forma: "*:idExpediente*".

La sentencia SQL se ejecuta sobre el esquema propio de PTw@ndA (PlataformaDatos) y no sobre la información del expediente en Trew@.

Requisitos a subsanar

Configuración automática de requisitos

Mediante la ejecución de un método de una clase: En este caso, en la definición del requisito, es necesario especificar la ruta de la clase junto con el método a ejecutar.

De forma análoga a la configuración automática del requisito mediante una sentencia SQL, el resultado de este método debe ser un Array de dos posiciones, la primera para indicar con '0' o '1' si el requisito es a exigir o no y la segunda posición para indicar el motivo de subsanación si procede.

La estructura del método a implementar debe ser la siguiente:

```
public Object[] nombreMetodo (Long idExpediente, Long idRequisito){
...Código del método
}
```

Gracias por su atención.