CONSEJERÍA DE HACIENDA, INDUSTRIA Y ENERGÍA

JUNTA DE ANDALUCIA
CONSEJERÍA DE HACIENDA, INDUSTRIA Y ENERGÍA

Formación a usuarios y personal técnico:
Plataforma de tramitación w@ndA

Dirección General de Transformación Digital
Junio de 2019

Plataforma de tramitación w@ndA

Plataforma de tramitación w@ndA

CONSEJERÍA DE HACIENDA, INDUSTRIA Y ENERGÍA

Introducción

La plataforma de tramitación PTw@ndA es el sistema de tramitación que se ofrece dentro del marco del proyecto w@ndA, utilizando el resto de componentes de Administración Electrónica ya existentes, integrándolos y ofreciendo una solución final al usuario tramitador. Como componente horizontal de tramitación cumple los siguientes requisitos:

- **Reutilizable para la tramitación de cualquier familia de procedimientos**, sirviendo como punto de partida y como software de base para abordar los desarrollos verticales y particulares de cada implantación.

- Funcionalmente deberá poder ser **ampliable mediante la instalación de nuevos módulos funcionales** construidos bajo unas especificaciones técnicas definidas.

- Completo **alineamiento con los requerimientos de la Ley 39/2015, de 1 de octubre, del Procedimiento Administrativo Común de las Administraciones Públicas, y Ley 40/2015, de 1 de octubre, de Régimen Jurídico del Sector Público**, respecto de la tramitación electrónica del procedimiento, favoreciendo la eficiencia en la gestión y el acceso electrónico a la información.

- Arquitectura totalmente **alineada con los componentes de Administración Electrónica**, **integrándolos y garantizando un uso correcto y controlado** de cada uno de ellos.

- **Parametrización de aspecto visual, permisos y módulos funcionales** utilizados en cada implantación, unidad organizativa y procedimiento.

- **Minimiza las labores de programación** necesarias para implantar una solución de tramitación electrónica de procedimientos administrativos.

- **Adaptación a normativas técnicas e interoperabilidad** (ENI/ENS).

- **Reduce el tiempo y coste** de puesta en marcha de un procedimiento electrónico.

Introducción

Características principales:

- Solución extensible de instrucción electrónica de procedimientos.
- Aplicación sobre Trew@ de integración de los sistemas y plataformas habilitantes de Administración Electrónica.
- Elimina la necesidad de desarrollar para cada procedimiento la funcionalidad común: transición de fases, gestión de interesados, generación de documentos, etc.
- Arquitectura tecnológica basada en Struts2, Spring e Hibernate, proporcionando una capa de servicios web para ser desplegados en un BUS de integración.
- Existen Guías de desarrollo de módulos funcionales para facilitar la construcción de los mismos.

Ventajas:

- **Reducción de esfuerzo de desarrollo:** sólo se desarrollan aquellos aspectos de la tramitación específicos de cada procedimiento.
- **Reducción de esfuerzo de mantenimiento:** la evolución del núcleo de PT garantiza que en todo momento los sistemas de tramitación estarán actualizados respecto al resto de plataformas y sistemas de AE sin necesidad de requerir esfuerzos adicionales de mantenimiento.
- **Homogeneización: disponibilidad de una aplicación común para la tramitación de expedientes de cualquier familia de procedimientos.** La incorporación o modificación de los procedimientos no requerirá esfuerzo de aprendizaje para los usuarios tramitadores.

Plataforma de tramitación w@ndA

Arquitectura de ejecución

La Plataforma de Tramitación w@ndA (PTw@ndA) y, en especial, su Escritorio de Tramitación se constituye actualmente como el componente central de tramitación en la cual se potencian los siguientes principios:

- **Principio de Modularidad.** PTw@ndA es modular, favoreciendo la adaptabilidad del modelo a los distintos escenarios que puedan darse de forma progresiva.

- **Principio de Reutilización de Componentes.** PTw@ndA reutiliza componentes ya existentes, de propiedad de las Administraciones Públicas, lo que abarata su desarrollo y mantenimiento.

- **Principio de Adaptabilidad y No Intrusión.** PTw@ndA es capaz de interoperar de una manera eficaz y eficiente con los componentes y *back-offices* ya existentes.

- **Principio de Normalización de los Servicios.** PTw@ndA está destinada a soportar procesos y servicios previamente optimizados y normalizados, de manera que se favorezca la eficacia interna.

- **Principio de Flexibilidad.** A pesar de la orientación a la normalización de los servicios, PTw@ndA es capaz de dar una respuesta ágil y eficaz a las particularidades de cada área y perfil.

- **PTw@ndA soporta servicios normalizados sin perder flexibilidad.** Esto es posible gracias a la incorporación de una herramienta de gestión de flujos que soportará los metaflujos y flujos asociados a los servicios telemáticos.

Arquitectura de ejecución

La utilización de la Plataforma de Tramitación w@ndA en un proyecto de tramitación electrónica englobando al resto de componentes w@ndA, aporta los siguientes beneficios:

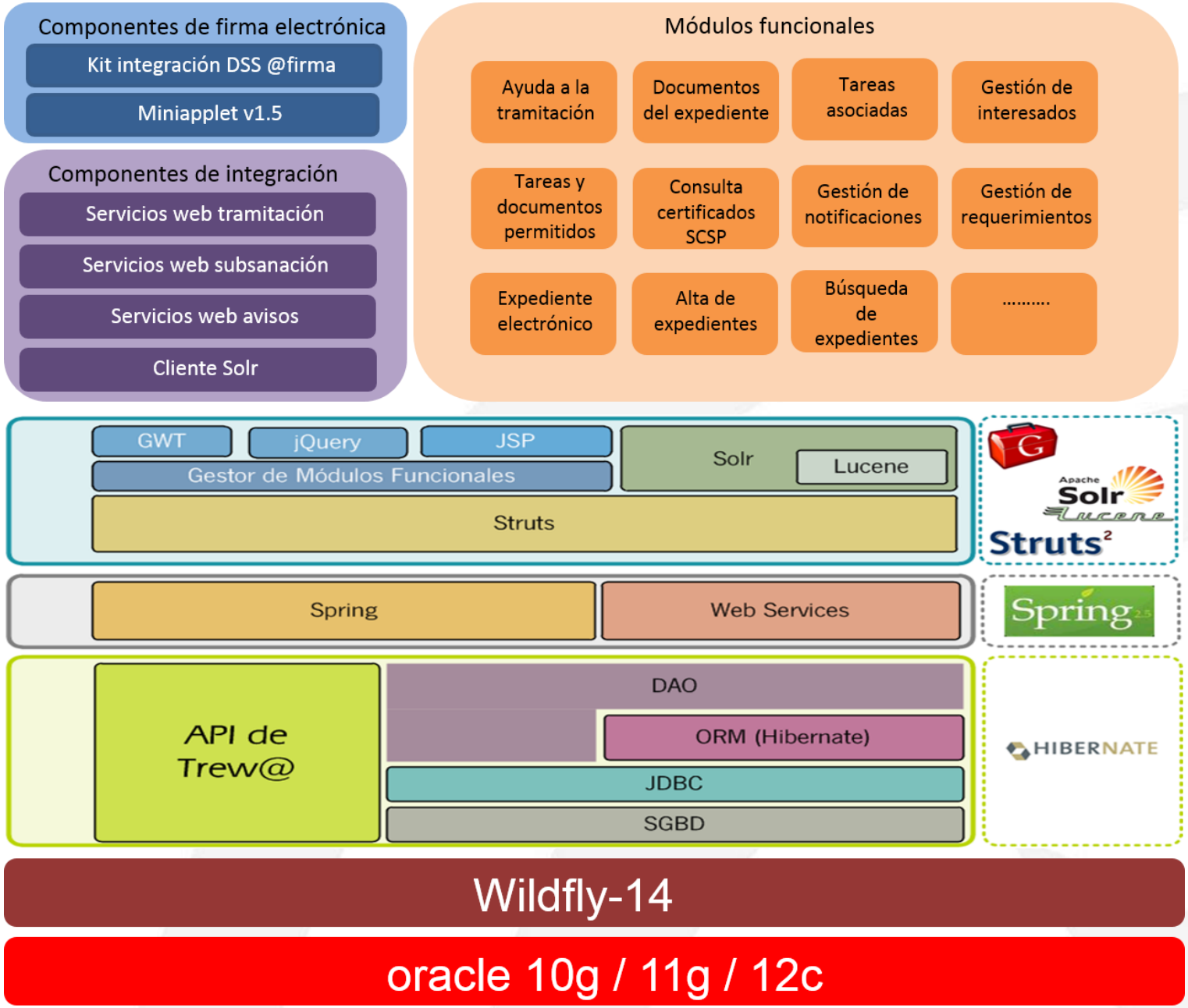
- **Entorno operativo global y homogéneo para los distintos usuarios gestores**, reduciendo considerablemente las tareas y por tanto los tiempos de tramitación. Entorno reutilizable, homogéneo y flexible para la incorporación de nuevas funcionalidades futuras sin impacto entre los distintos módulos.

- **Aseguramiento de la viabilidad y prestaciones de la solución**. Se toma como punto de partida una Plataforma de Tramitación ya desarrollada, lo que minimiza el riesgo de desarrollo y adaptación, así como permite acortar los tiempos de proyecto. La plataforma ofrecerá un conjunto de funcionalidades muy avanzadas.

- **Escalabilidad y Mantenimiento de la Plataforma**. La utilización de la Plataforma Corporativa de w@ndA asegura la escalabilidad, evolución y mantenimiento de la misma.

- Se proporciona un conjunto de utilidades y funcionalidades (*framework*) que **simplifican notablemente el desarrollo de proyectos que incorporen funcionalidades de gestión y tramitación electrónica**, pudiendo dar cobertura tanto a los procesos de *front-office* (servicios a "clientes") como a los procesos de *back-office* (gestión administrativa interna).

Arquitectura de ejecución



Plataforma de tramitación w@ndA

Módulos funcionales

Funcionalidad

Localización de expedientes

- Búsqueda genérica de expedientes
- Búsqueda avanzada de expedientes
- Datos del expediente
- Mi trabajo
- Gestión de avisos
- Gestión de caducidades
- Expedientes trasladados

Tramitación básica de expedientes

- Creación de expedientes
- Alta semi-telemática
- Escritorio de tramitación
- Tramitación
- Realización de tareas de manipulación de datos
- Incorporación de documentos
- Generación de documentos
- Firma, registro y notificaciones telemáticas
- Gestión de interesados

Tramitación avanzada de expedientes

- Asignación de usuarios
- Tramitación masiva
- Compulsa digital
- Evolución del expediente
- Incorporación de documentos por referencia
- Model@
- Expedientes relacionados
- Reserva y bloqueo de expedientes
- Editor de párrafos
- Subsanaciones

Apoyo a la tramitación

- Notas del expediente
- Mensajes
- Noticias
- Estadísticas
- Exportaciones de resultados
- Ayuda contextual

Integraciones

- Componentes w@ndA
- Motor de formularios
- SCSP
- Plataforma de pago telemático
- Sistema de gestión contable

Administración

- Gestión de módulos funcionales
- Indexación de expedientes
- Configuración de visibilidad
- Administración delegada de Trew@
- Datos paramétricos de configuración

PTw@ndA

Administración y configuración

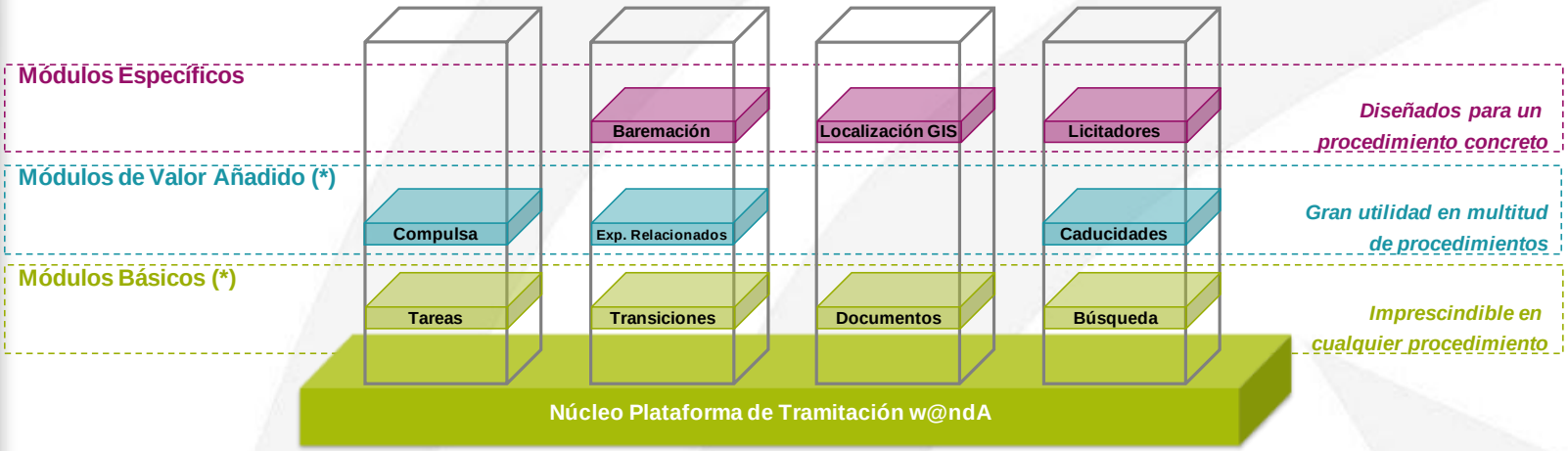
Módulos funcionales

Definición

La arquitectura y diseño modular de la Plataforma de Tramitación w@ndA, permite la inclusión de funcionalidad específica de un determinado negocio o implantación. Para ello se define un Módulo Funcional como el componente específico desarrollado para dar solución a una problemática particular de una entidad, independientemente de la naturaleza del negocio particular a gestionar (baremación, integración con GIS, altas específicas, ...)

Existen tres tipos de módulos, en función de su nivel de difusión:

- **Básicos**: presentes en todos los procedimientos.
- **De valor añadido**: presentes en gran número de procedimientos.
- **Específicos o verticales**: particulares a procedimientos concretos.



(*) Los Módulos Funcionales Básicos y de Valor añadido están incluidos de serien en PTw@ndA

Módulos funcionales

Características de los Componentes Funcionales

La plataforma aceptará la instalación de paquetes, plug-ins o componentes funcionales verticales y específicos de la familia de procedimientos que se desea tramitar:

- Un módulo podrá incorporar los siguientes recursos bajo un archivo ZIP: librerías (jars, páginas JSP, imágenes, CSS, etc.).
- Los módulos podrán implementar las reglas de navegación bajo cualquier versión de Struts.
- El módulo irá acompañado de un descriptor (archivo XML).
- La plataforma validará de forma automática en la instalación de un módulo:
  - Empaquetado correcto del ZIP.
  - Estructura del módulo.
  - El descriptor.
  - Las dependencias del módulo.
  - La URL asignado, de manera que no se encuentre ocupada por un módulo instalado anteriormente.
- Se ha elaborado una guía de desarrollo describiendo las directrices para la construcción de nuevos módulos funcionales sobre la plataforma.
- Una vez instalado el módulo, desde la herramienta de administración de la plataforma se configurará los aspectos relacionados con su presentación: posición, orden, asignación de roles, tamaño, etc..

Módulos funcionales

Tipología

Por su naturaleza funcional, existen dos tipos de componentes funcionales o módulos en la Plataforma de Tramitación w@ndA, aunque ambos son desarrollados de la misma forma:

- **Componentes funcionales de tramitación**, globales a cualquier familia de procedimientos.
- **Componentes funcionales verticales**, específicos de un procedimiento o familia de procedimientos.

En función de su forma de representación y de su alcance funcional, son 8 los tipos de módulos definidos que pueden ser desarrollados e implantados sobre la Plataforma de Tramitación:

- **Portlet**: Su funcionalidad es accesible a través de una sección ubicada en el escritorio de tramitación de expedientes. *Ej: tareas pendientes, documentos del expediente, transiciones.*
- **Externo**: Su funcionalidad es accesible desde el menú global de la aplicación no estando accesibles desde el escritorio de tramitación de expedientes. *Ej: alta de expedientes del procedimiento X, buscadores personalizados, estadísticas de tramitación.*
- **Utilidades**: Son invocados desde el escritorio de tramitación y ofrecen soporte y ayuda a la tramitación de los expedientes. *Ej: gestión de datos de interesados, incorporación de documentos, seguimiento de la tramitación efectuada hasta el momento.*
- **Procedimiento**: Implementan las tareas, acciones, condiciones y lógica de sustitución de variables en plantillas de documentos propias del modelado de un procedimiento concreto. En las tareas de manipulación de datos del expediente se requiere el desarrollo específico de formularios para el tratamiento de los datos, o bien, se habilita la integración con Formula para eliminar la necesidad de la programación a medida de este tipo de pantallas.

Módulos funcionales

Tipología

- **Administración**: Su funcionalidad es accesible desde la herramienta de administración, con objeto de realizar tareas de administración, parametrización o mantenimiento. *Ej: administración delegada de Trew@, mantenimiento de tablas parámetricas.*

- **Consulta**: Módulos cuya funcionalidad es añadir acciones que sean de interés para los trámites directamente sobre el listado de expedientes que se muestran tras la realización de una búsqueda de expedientes. *Ej: gestión de datos del expediente.*

- **Web Service**: Son módulos cuya funcionalidad es incluir el servicio web implementado en el archivo de descripción de web service que define el catálogo de servicios de la Plataforma de Tramitación. *Ej: servicios web V2, requisitos subsanar WS.*

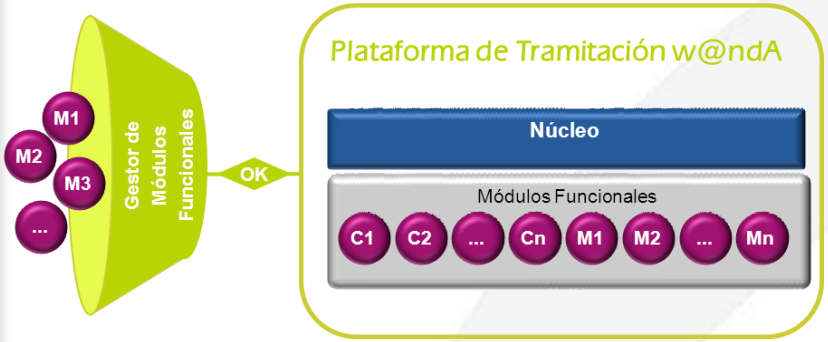
- **Utilidades masivas**: Módulos invocados desde el módulo de búsqueda de expedientes cuando se accede al detalle de un conjunto de expedientes.

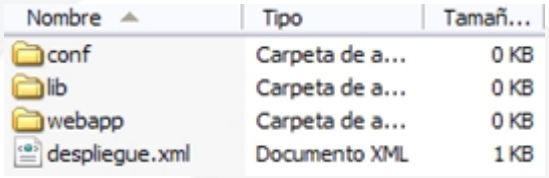
Módulos funcionales

Estructura

Los módulos específicos que deban diseñarse para solucionar una problemática concreta de un procedimiento o implantación, se incorporarán a PTw@ndA a través del **gestor de módulos funcionales**, que proporciona un mecanismo sencillo y perfectamente definido para la construcción de estos módulos específicos.

Los módulos construidos para ser desplegados en PTw@ndA deben empaquetarse en formato *ZIP*, siguiendo una estructura interna común predefinida para todos los tipos de módulos existentes:





- **Despliegue.xml**: Contiene información básica sobre el módulo que determinará el tratamiento que PTw@ndA le dará en su despliegue.
- **Lib**: Contiene clases java empaquetadas en uno o más ficheros .jar. Estas clases java pueden ser desarrollos propios del módulo a desplegar o librerías de terceros requeridas por los desarrollos propios. Es vital evitar el despliegue de librerías que puedan entrar en conflicto con las propias de PTw@ndA por lo que se hace imprescindible el uso de una gestión automatizada de los posibles conflictos. **Solución: MAVEN.**
- **Conf**: Configuración propia de servicios Spring y/o Struts2 del módulo: *beans*, *actions*, etc. El directorio puede estar vacío pero ha de estar siempre presente.
- **Webapp**: Recursos web del módulo: páginas HTML/JSP, imágenes, hojas de estilos, etc.

Módulos funcionales

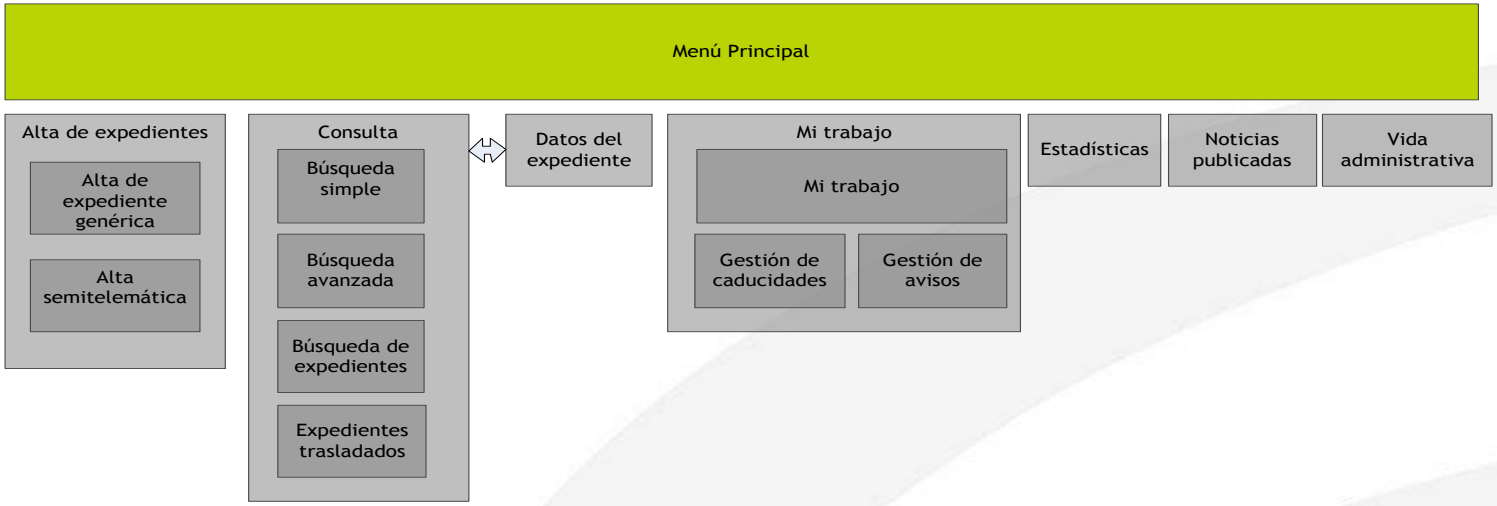
Estructura (despliegue.xml)

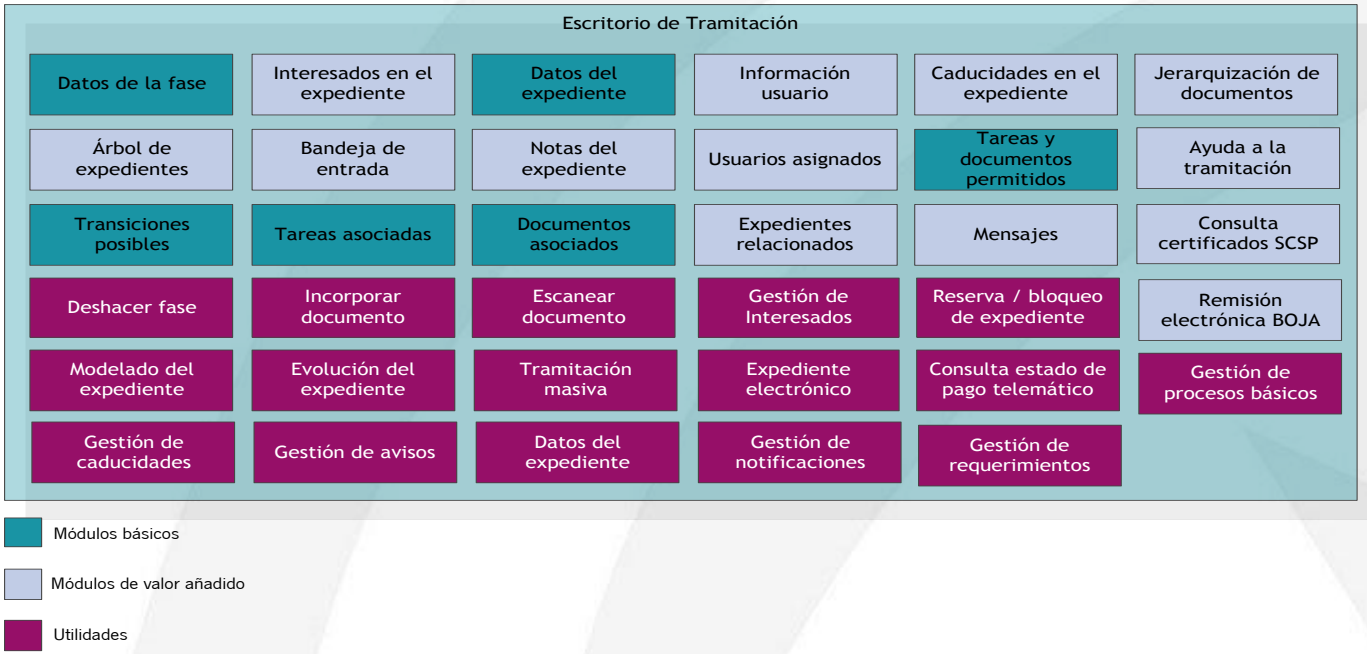
El archivo despliegue.xml es un descriptor del módulo en el que se detallan aspectos descriptivos e informativos del módulo. Los campos que estructuran este documento en formato xml son:

- **nombre:** nombre del módulo (deberá ser único en la Plataforma de Tramitación para que el módulo se pueda instalar correctamente).
- **version:** número de la versión del módulo.
- **titulo:** título mostrado en el escritorio de tramitación en caso de presentarse el módulo como un portlet en el mismo.
- **descripcion:** texto descriptivo del módulo.
- **url:** dirección correspondiente a la página inicial del módulo.
- **icono-on:** ruta del icono que se visualiza al posicionar el cursor sobre el acceso al módulo, en el caso de tratarse de un módulo de tipo utilidad.
- **icono-off:** ruta del icono que se visualiza, en el caso de tratarse de un módulo de tipo utilidad.
- **icono-menu-principal:** ruta del icono que se visualiza en el contenedor, en el caso de tratarse de un módulo de tipo externo.
- **autor:** datos del autor del módulo.
- **type:** tecnología con la que se ha desarrollado el módulo (puede tomar los valores STRUTS-2 o NONE).
- **tipoInstalacion:** forma de visualización del módulo (puede tomar los valores PORTLET, EXTERNO, UTILIDADES, NONE, WS, ADMINISTRACIÓN, CONSULTA, UTILIDADMASIVA).
- **postFuncion:** permite ejecutar una función javascript tras la recarga del módulo.
- **observados:** conjunto de módulos por los que tiene que ser informado el módulo cuando se produzcan cambios en ellos.

Módulos funcionales

Repositorio de módulos





Módulos funcionales

Repositorio de módulos

MÓDULO	VERS	TIPO	DESCRIPCIÓN
Alta de expediente genérica	2.0.1	Menú	Permite el alta de expediente en el motor de tramitación a partir de los datos genéricos requeridos (nº de expediente, título, fecha de alta, unidad organizativa, observaciones, ...). Una vez creado el expediente, se permite la asociación de interesados al expediente.
Búsqueda genérica	2.0.1	Menú	Localización de expedientes a partir de los datos indexados relativos al expediente: fase, título, número, interesados, observaciones, otros datos del expediente, ...
Búsqueda avanzada	2.0.1	Menú	Localización de expedientes a partir de los datos indexados relativos al expediente, filtrados en función de cada uno de los campos: fase, título, número, interesados, observaciones, otros datos del expediente, ...
Tareas pendientes	2.0.1	Menú	Gestión de las tareas pendientes asociadas a cada usuario, en función de las tareas de tramitación que puede realizar según su perfil en los expedientes del sistema. Este módulo será eliminado en la versión 2.1.0, sustituyéndose por "Mi trabajo"
Estadísticas	2.0.1	Menú	Permite consultar las estadísticas asociadas a la información de tramitación asociada a la implantación, pudiéndose consultar: trámites por fase, trámites por procedimientos y número de días de tramitación.
Caducidades	2.0.1	Menú	Visualización de caducidades del expediente, mostrándose las caducidades existentes en expedientes asignados al usuario tramitador. Este módulo será eliminado en la versión 2.1.0, sustituyéndose por "Mi trabajo" y "Gestión de caducidades".
Información del expediente	2.0.1	Portlet	Muestra la información del expediente en el escritorio de tramitación: procedimiento, número, título y fecha de alta.
Información del usuario	2.0.1	Portlet	Permite visualizar la información del usuario actual conectado a PTw@ndA, ofreciendo las opciones de acceso del menú principal y consulta de
Datos de la fase	2.0.1	Portlet	Muestra la información relativa a la fase actual del expediente, indicando la fase actual, transición de origen y fecha de entrada del expediente en la fase. Además en caso de existir eventos en el flujo de tramitación, permite cambiar la fase del expediente entre el evento ejecutado, y la propia fase del flujo de tramitación.
Transiciones disponibles	2.0.1	Portlet	Ofrece la relación de posibles transiciones en la evolución de un expediente dentro del modelado definido.

Módulos funcionales

Repositorio de módulos

MÓDULO	VERS	TIPO	DESCRIPCIÓN
Tareas y documentos permitidos	2.0.1	Portlet	Listado de tareas a realizar definidas en la fase de tramitación actual del expediente, distinguiéndose entre tareas de incorporación de documentos, tareas de generación de documentos y tareas de manipulación de datos (web).
Tareas asociadas	2.0.1	Portlet	Listado de tareas asociados al expediente actual, mostrándose para cada una de ellas la opción de borrar, descartar, reanudar o finalizar la tarea.
Documentos asociados	2.0.1	Portlet	Listado de documentos asociados al expediente actual, mostrándose para cada uno de los documentos las opciones de borrar, descartar, editar, enviar a firmar, firmar digitalmente, registrar telemáticamente, notificar telemáticamente, descargar documento y ver firmantes.
Interesados del expediente	2.0.1	Portlet	Listado de interesados asociados al expediente, permitiéndose la opción de consultar, modificar o eliminar la asociación con el expediente actual.
Interesados	2.0.1	Utilidad	Gestión de interesados, permitiéndose el alta y modificación de interesados, modificación de los datos de contacto, y asociación con el expediente actual.
Caducidades del expediente	2.0.1	Portlet	Módulo que muestra las caducidades existentes en el expediente actual que se está tramitando, con las opciones de modificar caducidad, ampliar o reanudar el plazo.
Usuarios asignados	2.0.1	Portlet	Listado de usuarios tramitadores que están asignados al expediente actual, siendo preferentemente tramitados por ellos, aunque no de forma obligatoria.
Mensajes	2.0.1	Portlet	Módulo que muestra ál usuario tramitador los mensajes que el sistema, u otros usuarios le han enviado, sirviendo como canal de comunicación entre los distintos usuarios de la aplicación.
Utilidades	2.0.1	--	Permite incorporar una barra de herramientas con enlaces a utilidades con visibilidad controlada por PTw@ndA. Es posible la incorporación de nuevas utilidades desarrollando nuevos módulos funcionales de tipo "Utilidad"
Deshacer	2.0.1	Utilidad	Utilidad que permite deshacer el estado actual del expediente, si no existen tareas no finalizas, ni documentos comenzados, volviendo a la fase anterior.
Incorporar documentos	2.0.1	Utilidad	Módulo de incorporación de documentos al expediente, pudiéndose realizar a partir de un fichero electrónico disponible en el puesto local del usuario, o bien existente en otro expediente del sistema.

Módulos funcionales

Repositorio de módulos

MÓDULO	VERS	TIPO	DESCRIPCIÓN
Model@	2.0.1	Utilidad	Applet gráfico que muestra el estado actual del expediente en el flujo de tramitación, y el histórico de fases por las que ha ido pasando.
Reservar expediente	2.0.1	Utilidad	Gestiona la reserva del expediente actual, permitiendo ser reservado para un usuario, de tal manera que únicamente pueda ser tramitado por él.
Escaner	2.0.1	Utilidad	Utilidad que posibilita la incorporación de documentos al expediente desde un escaner instalado y configurado en el puesto cliente del usuario tramitador.
Evolución del expediente	2.0.1	Utilidad	Muestra la evolución por la que ha pasado el expediente que se está tramitando, identificando las fases por las que ha pasado, y dentro de cada fase, las tareas que se han realizado.
Cambiar de procedimiento	2.0.1	Utilidad	Permite cambiar el contexto de tramitación a otro procedimiento.
Expedientes relacionados	2.0.1	Utilidad	Visualiza los expedientes relacionados con el expediente actual, mostrando un listado con la lista de expedientes relacionados, y permitiendo
Tramitación en lote	2.0.1	Portlet	Tramitación en bloque de expedientes, posibilitando la tramitación a una fase del procedimiento que se está tramitando, y permitiendo la generación de documentos en varios expedientes.
Gestión de noticias	2.0.1	Menú	Gestión de noticias publicadas en la unidad organizativa o entidad en la que se encuentre implantada PTw@ndA. Permite a los distintos usuarios tramitadores consultar las novedades de la organización, en función de su perfil y puesto de trabajo.
Notas del expediente	2.0.1	Portlet	Permite a los usuarios tramitadores gestionar notas asociadas al expediente, posibilitando compartir determinada información entre los usuarios similar a post-it de un calendario.
Gestión de datos del expediente	2.1.0	Menú	Este módulo permite visualizar para cada tipo de procedimiento los datos específicos, mostrando los distintos formularios definidos para el procedimiento. Posibilita la opción de realizar operaciones sobre el propio expediente, como la subrogación y el traslado.
Mi trabajo	2.1.0	Menú	Módulo que ofrece la posibilidad de consultar las tareas pendientes de los usuarios en función del perfil de tramitación asociado, visualizándose por procedimiento, fase y unidad organizativa. De forma adicional se muestran los avisos asociados a expedientes que tengan acceso, y caducidades definidas con fecha próxima de cumplimiento.

Módulos funcionales

Repositorio de módulos

MÓDULO	VERS	TIPO	DESCRIPCIÓN
Gestión de avisos	2.1.0	Menú	Módulo de gestión de avisos que permite la definición de avisos en el sistema, relacionados con procedimientos, fases o tareas asociados a expedientes. Cada definición de aviso tendrá asociado un mensaje, fecha de inicio del aviso y fecha de vigencia.
Gestión de caducidades	2.1.0	Menú	Módulo de gestión de caducidades que permite la gestión de caducidades definidas sobre tipos de expediente, pudiéndose realizar modificaciones en el plazo de la caducidad o proceder a su suspensión.
Expediente electrónico	2.1.0	Utilidad	Permite obtener los datos asociados a un expediente, incluyéndose los datos básicos, tareas realizadas, tareas pendiente, documentos asociados, historial de tramitación, y anexo con los documentos del expediente.
Integración plataforma de pago telemático	2.1.0	Utilidad	Utilidad que permite la consulta de un pago realizado a través de la plataforma de pago telemática de la Junta de Andalucía, permitiendo la descarga del documento justificante del pago.
Integración sistema de gestión contable	2.1.0	Utilidad	Integración con el sistema de gestión contable Júpiter a partir del componente Pasarela. Esta integración permitirá el envío y recepción de documentos contables.
Servicios de respuesta inmediata (SERIs)	2.1.0	Utilidad	Inclusión de acciones y servicios necesarios para permitir el modelado y desarrollo de servicios de respuesta inmediata, realizándose la generación de documentos y firma con certificado de servidor de manera instantánea.
Integración SCSP	2.1.0	Utilidad	Integración con los servicios de Supresión de Certificados en Soporte Papel, de forma que para cada interesado en función del consentimiento expreso que haya realizado, se permite la consulta de determinados certificados y documentos.
Tramitación masiva	2.1.0	Utilidad	Utilidad de tramitación masiva que posibilita las operaciones de transitar a una fase del procedimiento, y la generación de documentos de forma masiva entre una lista de expedientes que se encuentren en la misma fase.
Árbol de expedientes	2.1.0	Portlet	Bandeja de entrada de expedientes en el escritorio de tramitación, permitiendo la visualización de expedientes según la unidad organizativa del usuario y perfil, mostrando únicamente los expedientes sobre los que puede realizar tareas pendientes.
Gestión de ayuda	2.1.0	Menú	Módulo de gestión de ayuda de la aplicación, mostrándose la ayuda contextual asociada a los distintos módulos funcionales mostrados en la instalación.

Módulos funcionales

Repositorio de módulos

MÓDULO	VERS	TIPO	DESCRIPCIÓN
Vida administrativa	2.1.1	Menú	Permitir la consulta de toda la vida administrativa de un interesado dentro de una implantación.
Jerarquización de documentos	2.1.1	Portlet	Incluir la posibilidad de mostrar el listado de documentos asociados al expediente de forma jerárquica, permitiendo el filtrado de documentos y la realización de acciones de forma sencilla.
Búsqueda de expedientes	2.1.2	Menú	Proporcionar al tramitador una localización más ágil de los expedientes, permitiendo el acceso inmediato a los datos asociados al expediente sin necesidad de requerir pantallas adicionales.
Ayuda a la tramitación	2.1.2	Portlet	Ayudar al tramitador durante el proceso de tramitación de los expedientes de los procedimientos de las distintas familias, indicándole que actuaciones se pueden realizar sobre el expediente en cada fase del procedimiento y que componen el flujo de tramitación del mismo.
Administración delegada de Trew@	2.1.2	Administración	Permitir al usuario administrador la gestión de información relacionada con los usuarios tramitadores, sin necesidad de acceder a la herramienta de administración de Trew@, dando de alta nuevos usuarios, modificando usuarios existentes y gestionando los perfiles asociados a cada usuario.
Gestión personalizada de avisos	2.2.0	Menú	Permitir al usuario tramitador recibir avisos personalizados sobre la tramitación de expedientes y mostrar su información asociada.
Notificaciones	2.2.0	Utilidad	Gestión de las notificaciones ordinarias, telemáticas y por comparecencia a los interesados de los expedientes administrativos tramitados.
Requerimientos	2.2.0	Utilidad	Permitir realizar la gestión de los requerimientos realizados a los distintos interesados de los expedientes administrativos tramitados desde la plataforma de tramitación.
Gestión de procesos básicos	2.2.0	Utilidad	Proporcionar al usuario tramitador con el perfil adecuado la posibilidad de reiniciar o eliminar un expediente electrónico en cualquier fase del flujo de tramitación.
Remisión electrónica de disposiciones a BOJA	2.2.0	Procedimiento	Permitir la remisión electrónica de disposiciones al BOJA mediante la tramitación de los procedimientos “Preparar disposición” y “Enviar disposición”.

Módulos funcionales

Repositorio de módulos

MÓDULO	VERS	TIPO	DESCRIPCIÓN
Consulta de histórico de operaciones masivas	2.2.0	Menú	Proporcionar al usuario tramitador acceso al histórico de las operaciones masivas realizadas en el módulo “Búsqueda de expedientes”, tanto de tramitación en bloque como de documentos generados en bloque.
Servicios web v2 securizados	2.2.0	WS	Nueva versión de los servicios web de la plataforma de tramitación securizados mediante el uso de certificado digital en las cabeceras de las peticiones SOAP.
Consulta de datos de residencia con fecha de la última variación padronal	2.3.0	Procedimiento	Verificar y consultar los datos de residencia, incluyendo los datos de la fecha y causa de la última variación padronal consolidada en la base padronal del Instituto Nacional de Estadística.
Servicio web de generación de avisos	2.3.0	WS	Servicio web que permite que aplicaciones terceras ala plataforma de tramitación w@ndA puedan generar avisos que podrán ser consultados desde el módulo de gestión personalizada de avisos.
Integración con GIRO	2.4.1	Procedimiento	Gestión de documentación contable integrada con GIRO, Sistema Integrado de Gestión Presupuestaria, Contable y Financiera de la Junta de Andalucía. El módulo proporciona una interfaz de servicios intermedia para realizar peticiones a los servicios web proporcionados por GIRO para las áreas funcionales integradas.
Consulta de estar al corriente de pago con la Seguridad Social	2.4.2	Procedimiento	Integración con el servicio de consulta de estar al corriente de pago con la Seguridad Social, el cual será el encargado de verificar si una persona física o jurídica se encuentra al corriente de pago con la Seguridad Social

Plataforma de tramitación w@ndA

Desarrollo de un módulo funcional

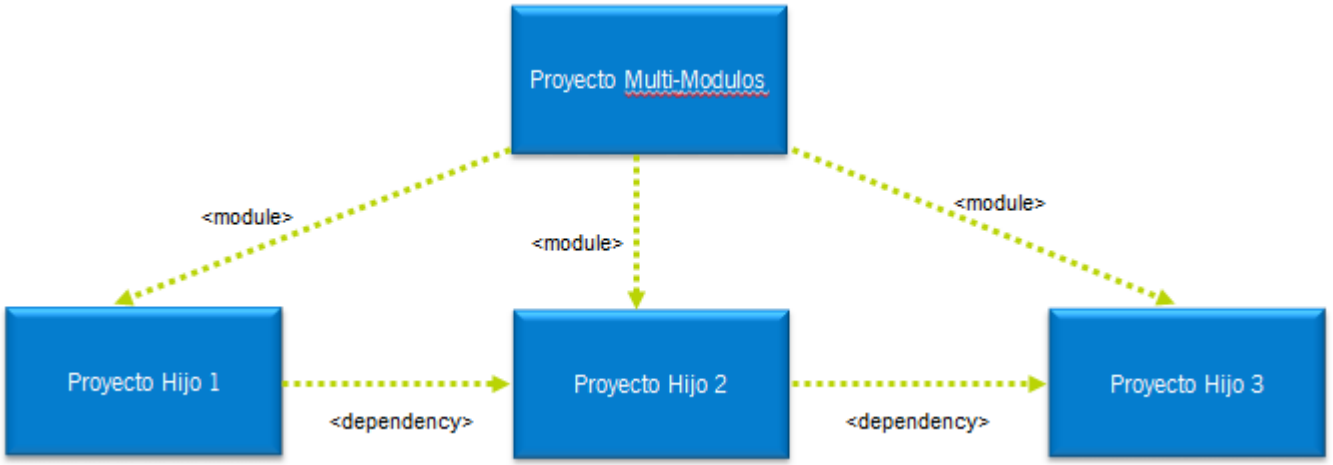
Creación del proyecto Maven

Para desarrollar un nuevo módulo funcional en PTw@ndA será necesario desarrollar y configurar una serie de componentes que formarán el desarrollo completo del módulo.

Apache Maven proporciona una estructura multi-modulos para el desarrollo que es muy recomendable utilizar cuando se vayan a implementar varios módulos.

Un proyecto multi-modulos es un tipo particular de proyecto, no produce ningún artefacto final y está compuesto de otros proyectos conocidos con el nombre módulos. Cuando se ejecuta un comando en el proyecto, lo ejecutará en todos los proyectos hijos. Maven es capaz, a través de su componente de reactor, descubrir el orden correcto de ejecución y detectar dependencias circulares. De esta forma obtenemos un fichero padre donde podremos declarar dependencias que serán comunes a todos los hijos, como por ejemplo, el core de plataforma.

La estructura de nuestro proyecto deberá ser la siguiente:



Desarrollo de un módulo funcional

Creación del proyecto Maven (cont.)

Los módulos hijos podrán tener dependencias entre ellos, siempre y cuando no se cree una dependencia circular.
El primer paso será crear nuestro proyecto multi-modulos. En este punto puede optarse por crear un proyecto maven desde linea de comandos ó crear “a mano” un pom.xml que identificará a nuestro proyecto multi-modulos. Como comentamos anteriormente, el proyecto multi-modulos no generará ningún artefacto, por lo cuál no tendrá una estructura de proyecto como tal, simplemente quedará instanciado con la creación del pom.xml.

Desarrollo de un módulo funcional

Creación del proyecto Maven (cont.)

```
<?xml version="1.0" encoding="UTF-8"?>
<project  xmlns="http://maven.apache.org/POM/4.0.0"  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0  http://maven.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>es.juntadeandalucia.ptwanda</groupId>
  <artifactId>modulosSubAdm</artifactId>
  <packaging>pom</packaging>
  <version>1.0</version>
  <name>Plataforma de tramitación</name>
  <url>http://maven.apache.org</url>
  <build>
      <plugins>
         <plugin>
           <groupId>org.apache.maven.plugins</groupId>
           <artifactId>maven-compiler-plugin</artifactId>
           <configuration>
             <encoding>ISO-8859-1</encoding>
             <source>1.8</source>
             <target>1.8</target>
             <showDeprecation>false</showDeprecation>
             <showWarnings>false</showWarnings>
           </configuration>
         </plugin>
      </plugins>
  </build>
</project>
```

Para la creación de un proyecto multi-modulos es necesario indicar tipo pom en el packaging.

Especificar el nivel de compilación para todos los proyectos hijos.

Desarrollo de un módulo funcional

Creación del proyecto Maven (cont.)

A continuación, creamos los distintos proyectos hijos. En este punto se crearán tantos proyectos hijos como módulos funcionales queramos desarrollar.
Los proyectos se crearán desde línea de comandos, lanzando el comando desde el directorio donde se encuentre el pom.xml del proyecto multi-modular ó bien desde la ruta donde se encuentre nuestro workspace si no se hace uso de dicha estructura.
Al crear el proyecto hijo, Maven actualizará el pom.xml del proyecto multi-modular, insertando la referencia al módulo hijo creado.
El comando a lanzar es:

```
mvn archetype:generate -DarchetypeGroupId=org.apache.maven.archetypes -DgroupId=es.juntadeandalucia.ptwanda -DartifactId=SubvencionesAdm -Dversion=1.0
```

Una vez ejecutado el comando anterior y establecido los valores solicitados para la creación del proyecto maven, se habrá modificado el pom.xml de la estructura multi-modular, incorporando el nuevo módulo creado.

Desarrollo de un módulo funcional

Creación del proyecto Maven (cont.)

```
<?xml version="1.0" encoding="UTF-8"?>
<project    xmlns="http://maven.apache.org/POM/4.0.0"    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>es.juntadeandalucia.ptwanda</groupId>
  <artifactId>modulosSubAdm</artifactId>
  <packaging>pom</packaging>
  <version>1.0</version>
  <name>Plataforma de tramitación</name>
  <url>http://maven.apache.org</url>
  <build>
      <plugins>
         <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-compiler-plugin</artifactId>
            <configuration>
              <encoding>ISO-8859-1</encoding>
              <source>1.8</source>
              <target>1.8</target>
              <showDeprecation>false</showDeprecation>
              <showWarnings>false</showWarnings>
            </configuration>
         </plugin>
      </plugins>
  </build>
<modules>
  <module>SubvencionesADM</module>
</modules>
</project>
```

Nuevo módulo creado

Desarrollo de un módulo funcional

Creación del proyecto Maven (cont.)

El módulo creado tiene una estructura por defecto, que es la que implementa el plugin utilizado en la generación del proyecto. Esta estructura habrá que adaptarla para que coincida con la estructura general de un módulo vertical para la plataforma, ya que a la hora de generar el empaquetado zip final, Maven localiza los ficheros a incluir en el zip basándose en esta estructura definida.

Todo módulo debe estar formado por los siguientes componentes:

- Fichero pom.xml
- Fichero modulo-vertical-pt.xml
- Fichero despliegue.xml
- Ficheros de configuración
- Clases Java

A continuación se especificarán las normas y consejos a seguir para cada uno de los componentes citados.

Desarrollo de un módulo funcional

Fichero pom.xml

El fichero Project Object Model (pom.xml) describe el proyecto de software a construir, sus dependencias de otros módulos y componentes externos, y el orden de construcción de los elementos.
Viene con objetivos predefinidos para realizar ciertas tareas claramente definidas, como la **compilación del código** y su **empaquetado** en diferentes formatos.
El contenido del fichero pom.xml se divide en tres bloques principales:

1. **Identificación del producto** generado por el código fuente que acompaña al documento pom.xml. Se hace mediante cuatro parámetros.
   a) **groupId:** Identificador del proyecto o módulo funcional. Se identifica con la estructura jerárquica de directorios que supone la localización de la librería en el repositorio que la aloje. Idealmente, se corresponde con la estructura de carpetas interna que tiene la librería, aunque no es una práctica obligatoria.
   b) **artifactId:** Identificador del módulo vertical.
   c) **name:** Texto con la descripción del módulo vertical.
   d) **version:** versión del producto.
2. **Dependencias:** Librerías o recursos externos requeridos para la generación del módulo funcional.
3. **Repositorios** donde se encuentran alojadas las citadas librerías. Deberá declararse el repositorio perteneciente al Servicio de Coordinación de Administración Electrónica: **https://ws024.juntadeandalucia.es/maven/**

Desarrollo de un módulo funcional

Fichero pom.xml (cont.)

El fichero Project Object Model (pom.xml) describe el proyecto de software a construir, sus dependencias de otros módulos y componentes

```
<project xmlns="http://maven.apache.org/POM/4.0.0" mlns:xsi=http://www.w3.org/2001/XMLSchema-
instance  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-
v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <artifactId>modulosSubAdm</artifactId>
    <groupId>es.juntadeandalucia.ptwanda</groupId>
    <version>1.0</version>
  </parent>
  <groupId>es.juntadeandalucia.ptwanda</groupId>
  <artifactId>SubvencionesADM</artifactId>
  <packaging>jar</packaging>
  <version>${project.parent.version}</version>
  <name>SubvencionesADM</name>
  <url>http://maven.apache.org</url>

  <dependencies>
       <dependency>
              <groupId>es.juntadeandalucia.ptwanda</groupId>
              <artifactId>ptwanda-core</artifactId>
              <version>2.2.0</version>
              <scope>provided</scope>
       </dependency>
  </dependencies>
```

El pom.xml del proyecto hijo tiene una referencia al proyecto multi-modulos.

Declarar una dependencia a otro proyecto hijo. Resaltar la importancia de declarar esta dependencia de tipo PROVIDED. De esta forma se indica a Maven que utilice el módulo hermano SÓLO para la compilación y no lo incluya en el zip de este módulo.

Desarrollo de un módulo funcional

Fichero pom.xml (cont.)

```
<build>
  <plugins>

    <!-- NO SE INCLUYEN EN EL JAR NI LOS RECURSOS WEB NI LOS XML DE CONFIGURACION  -->
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-jar-plugin</artifactId>
      <configuration>
        <excludes>
          <exclude>webapp/**</exclude>
          <exclude>despliegue.xml</exclude>
          <exclude>conf/**</exclude>
          <exclude>lib/**</exclude>
        </excludes>
      </configuration>
    </plugin>

     <!-- CON ESTE PLUGIN SE CONSTRUYE EL ZIP DEL MODULO -->
     <plugin>
        <artifactId>maven-assembly-plugin</artifactId>
        <configuration>
           <descriptors>
              <descriptor>modulo-vertical-pt.xml</descriptor>
           </descriptors>
        </configuration>
        <executions> <!-- FUERZO EL ASSEMBLY A LA FASE PACKAGE DEL PROYECTO -->
           <execution>
              <id>make-assembly</id>
              <phase>package</phase>
              <goals>
                 <goal>single</goal>
              </goals>
           </execution>
        </executions>
     </plugin>

  </plugins>
</build>

</project>
```

Desarrollo de un módulo funcional

Fichero pom.xml (cont.)

Para el desarrollo de todo módulo vertical de la plataforma es necesario especificar la dependencia al core de plataforma residente en el repositorio de CHIE. En el pom.xml de ejemplo se observa cómo declarar esta dependencia.

El plugin maven-jar-plugin es utilizado para generar el jar con las clases compiladas y los archivos de properties, que será añadido en el directorio /lib del empaquetado zip final. Para generarlo se excluyen los directorios conf, lib, webapp y el fichero despliegue.xml.

Para la construcción del zip del módulo se utiliza el plugin maven-assembly-plugin, indicándole la localización del fichero de configuración *modulo-vertical-pt.xml* y forzando el assembly en la etapa "package". Una vez ejecutado desde la línea de comando la instrucción mvn package, se generará el zip del módulo.

A la hora de especificar dependencias hay que tener especial cuidado que las librerías indicadas no sean incompatibles con las librerías propias de la plataforma de tramitación. Para ello puede observarse el pom.xml de la plataforma de tramitación para saber las dependencias que tiene.

Desarrollo de un módulo funcional

Fichero modulo-vertical-pt.xml

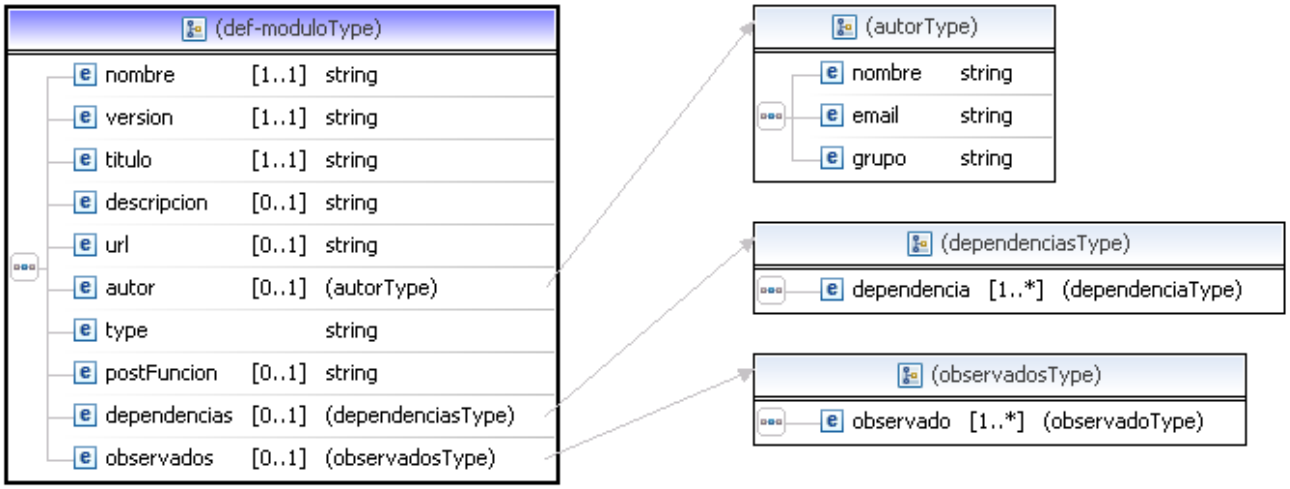
La configuración para la realización del assembly viene definida en el fichero modulo-vertical-pt.xml. Este fichero permanece invariable para cualquier módulo vertical, independientemente del tipo de módulo a generar. El contenido del fichero se muestra a continuación:

```
<assembly>
   <id>modulo-vertical-pt</id>
   <formats>
      <format>zip</format>
   </formats>
   <includeBaseDirectory>false</includeBaseDirectory>

  <dependencySets>
       <dependencySet>
          <outputDirectory>/lib</outputDirectory>
          <includes>
             <include>*:jar</include>
          </includes>
       </dependencySet>
   </dependencySets>


   <fileSets>
      <fileSet>
         <directory>${project.basedir}/src/main/resources</directory>
         <outputDirectory>/</outputDirectory>
         <includes>
            <include>webapp/**</include>
            <include>despliegue.xml</include>
            <include>conf/**</include>
            <include>lib/**</include>
         </includes>
      </fileSet>
      <fileSet>
         <directory>${project.basedir}/src/main/webapp</directory>
         <outputDirectory>webapp</outputDirectory>
         <includes>
            <include>**</include>
         </includes>
      </fileSet>
   </fileSets>

</assembly>
```

Desarrollo de un módulo funcional

Fichero despliegue.xml

Es obligatorio definir un fichero denominado despliegue.xml, en el cual se detallan aspectos descriptivos e informativos del módulo, actuando como fichero descriptor del módulo funcional. Este fichero se estructurará con los siguientes elementos:



El nombre del módulo especificado en la etiqueta <nombre> del archivo despliegue.xml será utilizado para desplegar los correspondientes archivos dentro del war de la plataforma. Por ejemplo, si el nombre de nuestro módulo es subadm, los recursos web se ubicarán en /modulos/subadm.
Resaltar la importancia de ausencia de espacios en blancos y otros caracteres especiales en la composición del nombre del módulo, ya que será utilizado para la construcción de rutas.

Desarrollo de un módulo funcional

Fichero despliegue.xml (cont.)

Los campos que estructuran este documento en formato xml son:

- **nombre:** nombre del módulo (deberá ser único en la Plataforma de Tramitación para que el módulo se pueda instalar correctamente).
- **version:** número de la versión del módulo.
- **titulo:** título mostrado en el escritorio de tramitación en caso de presentarse el módulo como un portlet en el mismo.
- **descripcion:** texto descriptivo del módulo.
- **url:** dirección correspondiente a la página inicial del módulo.
- **icono-on:** ruta del icono que se visualiza al posicionar el cursor sobre el acceso al módulo, en el caso de tratarse de un módulo de tipo utilidad.
- **icono-off:** ruta del icono que se visualiza, en el caso de tratarse de un módulo de tipo utilidad.
- **icono-menu-principal:** ruta del icono que se visualiza en el contenedor, en el caso de tratarse de un módulo de tipo externo.
- **autor:** datos del autor del módulo.
- **type:** tecnología con la que se ha desarrollado el módulo (puede tomar los valores STRUTS-2 o NONE).
- **tipoInstalacion:** forma de visualización del módulo (puede tomar los valores PORTLET, EXTERNO, UTILIDADES, NONE, WS, ADMINISTRACIÓN, CONSULTA, UTILIDADMASIVA).
- **postFuncion:** permite ejecutar una función javascript tras la recarga del módulo.
- **observados:** conjunto de módulos por los que tiene que ser informado el módulo cuando se produzcan cambios en ellos.

Desarrollo de un módulo funcional

Ficheros de configuración

Para implementar un nuevo módulo funcional, es necesario introducir ficheros de configuración que permitan adjuntar el módulo a PTw@ndA, y en los que se definan los recursos incorporados y su comportamiento.
Concretamente, es necesario definir por cada módulo dos ficheros correspondientes a la configuración de la navegación y presentación del módulo (Struts 2), y otro con la configuración de los componentes de negocio y acceso a datos (Spring).
A continuación se indica la estructura y características de cada uno de estos dos ficheros:

- Fichero de configuración Struts:

Este fichero incluye las nuevas definiciones de acciones (actions) que se invocarán desde la capa de presentación (páginas JSP), y las relaciones y redirecciones entre ellas.
Es necesario seguir las siguientes pautas para el correcto desarrollo y despliegue del módulo:

- ✓ La nomenclatura debe ser struts-<nombre-modulo>.xml
- ✓ El fichero de configuración para el alta debe tener el namespace: modulos/<nombre-modulo>
- ✓ El fichero de configuración para las tareas debe tener el namespace: agenda/tareas
- ✓ No puede declararse una acción (action) cuyo nombre ya exista en PTw@ndA.

Desarrollo de un módulo funcional

Ficheros de configuración (cont.)

A continuación se muestra un ejemplo de fichero de configuración de Struts:

```
<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 2.0//EN"
    "http://struts.apache.org/dtds/struts-2.0.dtd">

<struts>

        <package name="moduloNuevo" namespace="/modulos/moduloNuevo" extends="struts-
default">


                <action name="accion1" method="metodoAccion1"
                        class="claseAction">
                        <result name="success">/modulos/moduloNuevo/bbb.jsp</result>

                        <result name="otraAccion">/modulos/moduloNuevo/ccc.jsp</result>

                </action>

</struts>
```

- Fichero de configuración de Spring:

Este fichero incluye las definiciones de beans de negocio y de acceso a datos, y las referencias o inyecciones de otros componentes ya existentes en Ptw@ndA.

Es necesario seguir las siguientes pautas para el correcto desarrollo y despliegue del módulo:

- ✓ Solo existirá un único fichero de configuración por módulo.
- ✓ No pueden utilizarse como id de beans incorporados los ya existentes en PTw@ndA. (Dentro de la ruta /WEB-INF/clases/ se encuentran los ficheros de configuración de Spring donde podrán consultarse los ids de los beans ya existentes en la plataforma).

Desarrollo de un módulo funcional

Ficheros de configuración (cont.)

A continuación se muestra un ejemplo de fichero de configuración de Spring. Es importante destacar que la cabecera de los ficheros de configuración debe ser siempre la indicada en el ejemplo siguiente:

```
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:aop="http://www.springframework.org/schema/aop"
       xsi:schemaLocation="
                    http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-2.5.xsd
                    http://www.springframework.org/schema/aop
http://www.springframework.org/schema/aop/spring-aop-2.5.xsd">

       <bean id="servicio" class="es.juntadeandalucia.plataforma.aaa.ServicioNuevo"
       scope="prototype" parent="PTWandaService">
              <property name="servicioAInyectar">
                    <ref bean="referenciaServicioAInyectar"/>
              </property>
       </bean>

       <bean id="servicioTrewa" class="es.juntadeandalucia.plataforma.aaa.ServicioTrewa"
       scope="prototype" parent="ConfiguracionTramitacionService">
              <property name="servicioAInyectar">
                    <ref bean="referenciaServicioAInyectar"/>
              </property>
       </bean>
```

Es importante nombrar estos ficheros con el nombre "*applicationContext.xml*", para que el framework Spring los inyecte (cargue) automáticamente al estar definido de tal forma en el fichero "web.xml", de la aplicación Plataforma de Tramitación:

```
<context-param>
        <param-name>contextConfigLocation</param-name>
        <param-value>
                classpath*:/applicationContext.xml,
        </param-value>
</context-param>
```

Desarrollo de un módulo funcional

Clases JAVA

Dependiendo del tipo de módulo que se quiera desarrollar y de su objetivo, se requerirá el desarrollo de clases Java que implementen la funcionalidad deseada, implementen variables, nuevos servicios, etc...
Es recomendable que se siga como convección para la nomenclatura de paquetes la ruta *es.juntadeandalucia.plataforma.modulo.* <NOMBRE_MODULO>

Dependiendo del tipo de módulo es posible que se incorporen los siguientes tipos de clases Java:

- **Portlet:** Clases Action, Servicios, Clases DAO, Clases DTO.
- **Externo:** Clases Action, Servicios, Clases DAO, Clases DTO.
- **Utilidades:** Clases Action, Servicios, Clases DAO, Clases DTO.
- **Procedimiento:** Clases Condiciones, Clases Acciones, Clases Variables, Clases Action, Servicios, Clases DAO, Clases DTO.

PTw@ndA ofrece una amplia capa de Servicios de negocio que permiten interactuar con el motor de tramitación Trew@, gestionar la configuración y visualización de módulos, noticias, notas del expediente, ...
En el caso que la implementación del módulo funcional requiera la creación de un nuevo Servicio, es necesario que se especifique la herencia del Servicio base de PTw@ndA: **PTWandaServiceImpl**. Para ello es necesario que se especifique en la definición de la clase Java:

Desarrollo de un módulo funcional

Clases JAVA (cont.)

```
/**
 * <p>
 * Servicio nuevo destinado a ...
 * </p>
 *
 * @author UTE
 * @version 1.0
 */
public class ServicioNuevoImpl extends PTWandaServiceImpl implements IServicioNuevo {
```

A la hora de definir este servicio dentro del fichero de configuración applicationContext.xml, habrá de definirse expresamente el padre del nuevo servicio:

```
<bean id="servicioNuevo" class="es.juntadeandalucia.plataforma.ServicioNuevoImpl"
parent="PTWandaService">

        ...

</bean>
```

Si el Servicio desarrollado requiere interaccionar con el motor de tramitación Trew@, es conveniente que la herencia se realice sobre el servicio **ConfiguracionTramitacionServiceImpl**, en el que se definen métodos que permiten interactuar y gestionar la utilización de la API de acceso a Trew@.

Plataforma de tramitación w@ndA

Motor de indexación Solr

Introducción

El motor de indexación y búsqueda de la Plataforma de Tramitación w@ndA está compuesto en su núcleo por el proyecto Solr de Apache Lucene. Las características de dicho sistema son:

- Avanzadas capacidades de búsqueda full-text.
- Optimizado para soportar un volumen de tráfico elevado.
- Basado en interfaces abiertas, como XML y http.
- Escalable.
- Flexible y parametrizable en base a archivos de configuración en formato XML.
- Arquitectura extensible en base a plug-ins.

Este componente de la Plataforma de Tramitación habilitará búsqueda sobre el motor de tramitación Trew@.

Motor de indexación Solr

SolrCloud

Solr 4.0 incluye una nueva función de alta disponibilidad llamado SolrCloud, que utiliza fragmentación de índices y replicación del esquema para lograr la persistencia de datos. Emplea Apache ZooKeeper para mantener la configuración y coordinar el estado entre los fragmentos.

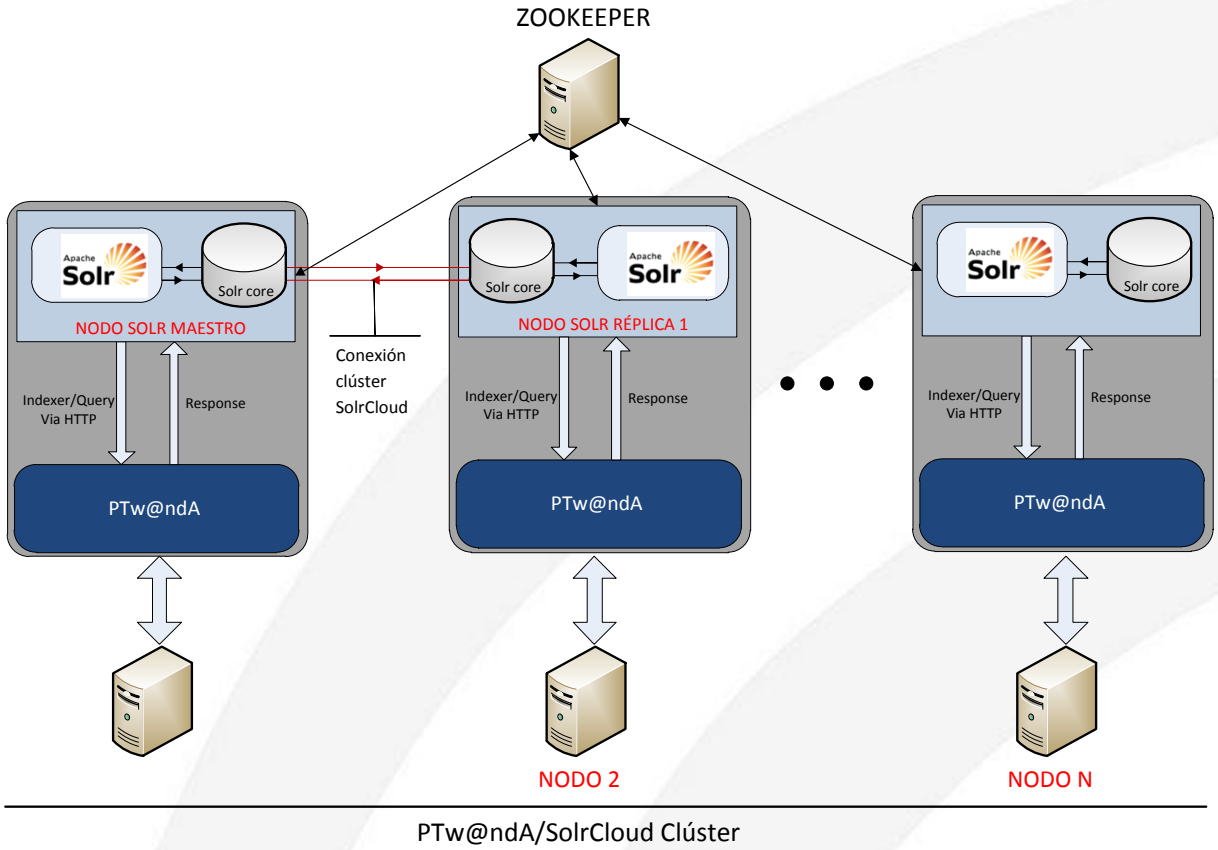
El proyecto Solr mantiene una wiki señalando tres implementaciones básicas de SolrCloud. Para la aplicación simple, SolrCloud incluye una versión incorporada de ZooKeeper. Las configuraciones señaladas por el wiki aprovechan la aplicación embedded ZooKeeper. Aunque esto es aceptable para fines de desarrollo y pruebas, el rendimiento y alta disponibilidad nos llevan a la necesidad de una separación de Solr y ZooKeeper.

La implementación de SolrCloud propuesta para un entorno en modo clúster de PTw@ndA no hace uso de la funcionalidad de fragmentación de índices, utilizándose un factor de replicación 1, esto es, el conjunto de índices se almacena en un nodo maestro, existiendo otro nodo réplica para lograr la persistencia de datos.

Para una configuración en clúster de la plataforma formada por dos nodos, uno de los nodos de PT ejercería el rol de líder y el otro nodo el rol de réplica.

Motor de indexación Solr

SolrCloud (cont.)



Motor de indexación Solr

Invocación de los servicios de indexación

La Plataforma de Tramitación implementa un servicio exclusivo para la indexación de la información especificada en el fichero schema.xml. Para realizar el mapeo de los diferentes campos a poder indexar se ha creado un objeto que mapea dichos campos.

El objeto **ExpedienteTrewa**, localizado en el paquete **es.juntadeandalucia.plataforma.expediente**, contiene toda la información necesaria que será utilizada por el servicio de indexación. La información contenida en el objeto de Trew@ TrExpediente será almacenada íntegramente como un atributo en este objeto.

El servicio encargado de recibir la información e indexarla en el motor de indexación Solr se denomina IIndexacionService, el cuál implementa 3 métodos para la comunicación con el motor de indexación:

- **crearExpediente:** método que recibe un objeto IExpediente e indexa su contenido en función de los campos que se han definido en el schema de indexación de Solr (schema.xml).
- **actualizarExpediente:** método que recibe un objeto IExpediente y actualiza el contenido de la información en el núcleo de Solr. En caso de no existir el objeto en el núcleo, se creará un índice nuevo que contendrá la información correspondiente.
- **eliminarExpediente:** método que recibe una cadena con la referencia del expediente y elimina del núcleo de Solr toda la información asociada a él.

La interfaz de este servicio es la siguiente:

Motor de indexación Solr

Invocación de los servicios de indexación (cont.)

```
package es.juntadeandalucia.plataforma.service.busqueda;

/**
 * <p>
 * Interfaz que define las operaciones p&uacute;blicas disponibles para la
 * indexaci&oacute;n de contenidos en el motor de b&uacute;squeda de PTw@ndA.
 * </p>
 *
 * @author everis
 * @version 1.0
 */
public interface IIndexacionService extends IPublicService, IPTWandaService {

    /**
     * Método que extrae la información del expediente para indexarlo.
     * @param expediente Expediente a indexar
     * @throws ArchitectureException Excepción de arquitectura
     * @throws BusinessException Excepción de negocio
     */
    public void crearExpediente(IExpediente expediente) throws ArchitectureException, BusinessException;


    /**
     * Actualiza la información de un expediente en el motor de indexación.
     * @param expediente Expediente a actualizar.
     * @throws ArchitectureException Excepción de arquitectura
     * @throws BusinessException Excepción de Negocio
     */
    public void actualizarExpediente(IExpediente expediente) throws ArchitectureException, BusinessException;


    /**
     * Método que elimina un expediente del motor de indexación
     * @param numExpediente Identificador del expediente a eliminar
     * @throws ArchitectureException Expepción de arquitectura
     */
    public void eliminarExpediente(String numExpediente) throws ArchitectureException;

}
```

Motor de indexación Solr

Indexación de campos específicos

Desde la versión 2.1.1 de la Plataforma de Tramitación se permite la posibilidad de indexar campos específicos en el esquema de indexación de Solr, manteniendo siempre por defecto el esquema proporcionado por la plataforma.

El esquema de indexación se ha modificado para contemplar la indexación de campos definidos de forma dinámica. Se han incluído los siguientes campos:

```
<dynamicField name="*_i" type="int" indexed="true" multiValued="true" stored="true" />
<dynamicField name="*_s" type="string" indexed="true" multiValued="true" stored="true" />
<dynamicField name="*_l" type="long" indexed="true" multiValued="true" stored="true" />
<dynamicField name="*_t" type="text" indexed="true"  multiValued="true" stored="true" />
<dynamicField name="*_es" type="text_es" indexed="true"  multiValued="true" stored="true" />
<dynamicField name="*_b" type="boolean" indexed="true" multiValued="true" stored="true" />
<dynamicField name="*_f" type="float" indexed="true" multiValued="true" stored="true" />
<dynamicField name="*_d" type="double" indexed="true" multiValued="true" stored="true" />
<dynamicField name="*_dt" type="date" indexed="true" multiValued="true" stored="true" />
```

Para hacer uso de la funcionalidad de indexación dinámica es necesario la creación de una clase que será la encargada de establecer los nuevos campos a indexar junto con sus valores asociados. Esta clase deberá implementar la interface *IExtractorInformacion*.

Una vez creada la clase será necesario configurar el parámetro de configuración IMPLEMENTACION_EXTRACTOR_SOLR. Este parámetro debe contener la ruta completa de la clase creada, incluido el package, por ejemplo, 'es.juntadeandalucia.plataforma.busqueda.ClaseIndexador'.

Es importante destacar que la nueva clase no deberá contener los campos de indexación por defecto ya que la plataforma seguirá gestionando el esquema inicial con independencia de tener una clase extractor específica.

Motor de indexación Solr

Indexación de campos específicos

A continuación se muestra un ejemplo de una clase de prueba que realiza la indexación de campos específicos:

```
package es.juntadeandalucia.plataforma.busqueda;

import java.util.HashMap;
import java.util.Map;
import es.juntadeandalucia.plataforma.comunes.excepciones.ArchitectureException;
import es.juntadeandalucia.plataforma.comunes.excepciones.BusinessException;
import es.juntadeandalucia.plataforma.service.busqueda.IExtractorInformacion;
import es.juntadeandalucia.plataforma.service.expediente.IExpediente;

public class ClaseIndexador implements IExtractorInformacion {

   @Override
   public Map<String, Object> extraerInformacion(IExpediente expediente,
         String path) throws BusinessException, ArchitectureException {

      Map<String, Object> mapaIndexador = new HashMap< String, Object>();

      mapaIndexador.put("interesados_s", "00000000T");
      mapaIndexador.put("precio_d", "1045.45");

      return mapaIndexador;
   }

}
```

Motor de indexación Solr

Consulta de campos específicos

Para la consulta de expedientes a Solr mediante campos específicos será necesario realizar el desarrollo de un servicio de búsqueda que haga uso del API de Solrj. Como paso previo se realizará la construcción de la consulta a enviar a Solr.

BusquedaAction.java

```
public final String buscar() throws ArchitectureException {
   ...
   this.listaExpedientes =
busquedaService.realizaBusqueda(parametrosBusqueda, null,
webuser.getSistema(), this.getUsuariosExp());
   ...
```

BusquedaServiceImpl.java

```
public List<ExpedienteSolrBean> realizaBusqueda(Map<String, String>
parametros, ISistema sistema,
                        String usuarioExpediente) {

String aux = new
ConstructorConsultaTrewaSigloCon().construirConsulta(parametros);

  consulta = aux + this.modificadorConsulta;
  this.consultaSolr(sistema, paginacion, consulta);
  listaExpedientesEncontrados = paginacion.getExpedientes();
  return listaExpedientesEncontrados;
}
```

ConstructorConsultaTrewa.java

Motor de indexación Solr

Consulta de campos específicos

BusquedaServiceImpl.java

```
public List<ExpedienteSolrBean> realizaBusqueda(Map<String, String> parametros, ISistema sistema,
        String usuarioExpediente) throws ArchitectureException {
    List<ExpedienteSolrBean> listaExpedientesEncontrados;
    String consulta = null;

    String aux = new ConstructorConsultaTrewaSigloCon().construirConsulta(parametros);

    if (!"".equals(aux)) {
        aux = aux.concat(" AND ");
    }

    if (sistema != null) {
        aux = aux.concat("sistema:");
        aux = aux.concat(sistema.getCodigo());
    }

    consulta = aux + this.modificadorConsulta;

    Paginacion paginacion = new Paginacion();
    paginacion.setColumna(10);
    paginacion.setOrden(1);
    paginacion.setPagina(1);
    this.consultaSolr(sistema, paginacion, consulta);

    listaExpedientesEncontrados = paginacion.getExpedientes();

    return listaExpedientesEncontrados;
}
```

Motor de indexación Solr

Consulta de campos específicos

ConstructorConsultaTrewa.java

```
public String construirConsulta(Map<String, String> parametrosBusqueda) throws ArchitectureException {
    StringBuilder result = new StringBuilder("");
    // Indica si debe mostrarse o no el resumen con campos resaltados de

    // Búsqueda por términos (activamos el resumen)
    result = construyeQuery(parametrosBusqueda, result);
    result = contruyeQuery2(parametrosBusqueda, result);
    if (!"".equals(parametroFechaPublicacion(parametrosBusqueda))) {
        if (result.length() > 2) {
            result.append(AND);
        }
        result.append(parametroFechaPublicacion(parametrosBusqueda));
    }

    if (null != (String) parametrosBusqueda.get("contador")
            && !"1".equals((String) parametrosBusqueda.get("contador"))) {
        int contador = Integer.parseInt((String) parametrosBusqueda.get("contador"));
        for (int i = 1; i < contador; i++) {
            if (result.length() > 2) {
                result.append(AND);
            }
            result.append(parametroCasillaSolicita(parametrosBusqueda, i));
        }
    }

    result = construyeQueryProcFase(parametrosBusqueda, result);

    return result.toString();
}
```

Motor de indexación Solr

Consulta de campos específicos

ConstructorConsultaTrewa.java

```
public StringBuilder construyeQuery(Map<String, String> parametrosBusqueda, StringBuilder result) {

    if (!"".equals(parametroPorDefecto(parametrosBusqueda))) {
        result.append(parametroPorDefecto(parametrosBusqueda));
    }

    if (!"".equals(parametroNumeroExpediente(parametrosBusqueda))) {
        if (result.length() > 2) {
            result.append(AND);
        }
        result.append(parametroNumeroExpediente(parametrosBusqueda));
    }
    if (!"".equals(parametroArchivadoExpediente(parametrosBusqueda))) {
        if (result.length() > 0) {
            result.append(AND);
        }
        result.append(parametroArchivadoExpediente(parametrosBusqueda));
    }
    if (!"".equals(parametroTituloExpediente(parametrosBusqueda))) {
        if (result.length() > 2) {
            result.append(AND);
        }
        result.append(parametroTituloExpediente(parametrosBusqueda));
    }
    if (!"".equals(parametroOtrosDatosExpediente(parametrosBusqueda))) {
        if (result.length() > 2) {
            result.append(AND);
        }
        result.append(parametroOtrosDatosExpediente(parametrosBusqueda));
    }

    return result;

}
```

Motor de indexación Solr

Consulta de campos específicos

ConstructorConsultaTrewa.java

```
public StringBuilder construyeQuery3(Map<String, String> parametrosBusqueda, StringBuilder result) {

    if (!"".equals(parametroUnidadOrganizativa(parametrosBusqueda))) {
        if (result.length() > 2) {
            result.append(AND);
        }
        result.append(parametroUnidadOrganizativa(parametrosBusqueda));
    }
    if (!"".equals(parametroTipoContrato(parametrosBusqueda))) {
        if (result.length() > 2) {
            result.append(AND);
        }
        result.append(parametroTipoContrato(parametrosBusqueda));
    }

    return result;

}
private String parametroFechaPublicacion(Map<String, String> parametrosBusqueda) throws ArchitectureException {
    try {
        if (parametrosBusqueda.get(FECHA_CREACION_EXPEDIENTE_FIELD) != null) {
            String[] rangoFechas = parametrosBusqueda.get(FECHA_CREACION_EXPEDIENTE_FIELD).split(",");
            Date datNulo = null;
            Date a = !"".equals(rangoFechas[0].trim()) ? new SimpleDateFormat("yyyy-MM-dd").parse(rangoFechas[0])
                    : datNulo;
            Date b = !"".equals(rangoFechas[1].trim()) ? new SimpleDateFormat("yyyy-MM-dd").parse(rangoFechas[1])
                    : datNulo;
            String fechaInicial = a != null ? new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss'Z'").format(a) : "*";
            String fechaFinal = b != null ? new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ss'Z'").format(b) : "*";

            return FECHA_CREACION_EXPEDIENTE_FIELD + ":" + "[" + fechaInicial + " TO " + fechaFinal + "]";
        }
    } catch (ParseException e) {
        throw new ArchitectureException(e.getMessage());
    }
    return "";
}
```

Motor de indexación Solr

Consulta de campos específicos

BusquedaServiceImpl.java

```
public void consultaSolr(ISistema sistema, Paginacion paginacion, String consulta) {
    List<ExpedienteSolrBean> exps = null;

    SolrServer server = FactoriaSolrServer.obtenerIntanciaServidorSolr();

    SolrQuery query = new SolrQuery(consulta);
    query = configuraMaximoResultado(query, paginacion, server, "");

    obtieneExps(paginacion, query, server, exps);
}
```

```
public Paginacion obtieneExps(Paginacion paginacion, SolrQuery query, SolrServer server,
        List<ExpedienteSolrBean> exps) {

    List<ExpedienteSolrBean> expedientes = exps;

    try {
        QueryResponse rsp = server.query(query, METHOD.POST);

        SolrDocumentList docs = rsp.getResults();

        Long total = docs.getNumFound();
        paginacion.setCantidad(total.intValue());

        expedientes = rsp.getBeans(ExpedienteSolrBean.class);

        paginacion.setExpedientes(expedientes);

    } catch (SolrServerException e) {
        String message = "Error in Solr";
        LOGGER.log(Level.SEVERE, message, e);
    }

    return paginacion;
}
```

Plataforma de tramitación w@ndA

Arquitectura Multi-Trew@

Introducción

Desde la versión 2.4.0, se evolucionó la arquitectura de la Plataforma de Tramitación de forma que es posible la comunicación con más de una instancia del motor de tramitación Trew@. Cada instancia del motor de tramitación se traduce en un nombre JNDI único declarado en los ficheros de configuración del servidor de aplicaciones WildFly.

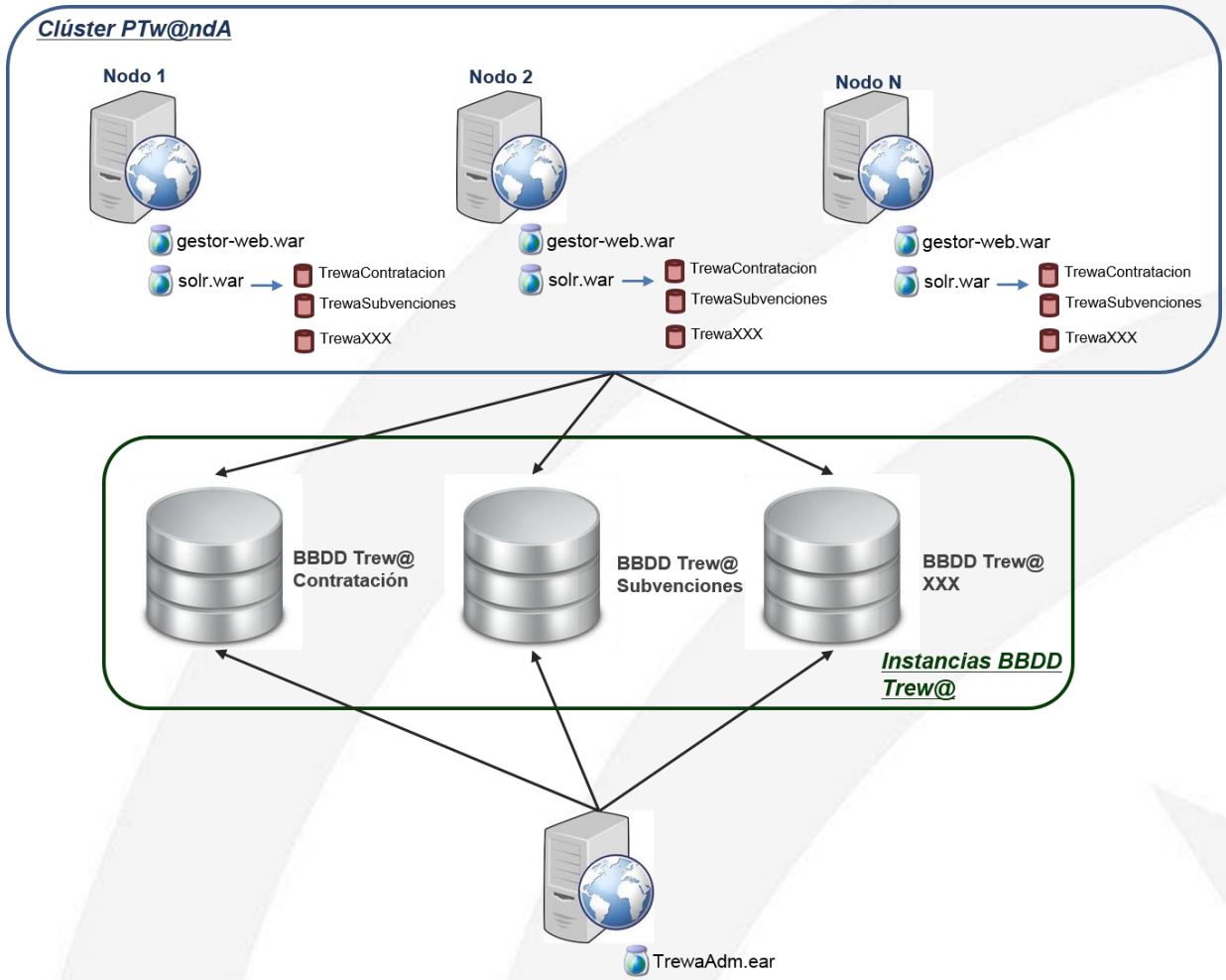
Esto supone importantes beneficios para los usuarios de la plataforma de tramitación ya que es posible desde la misma aplicación la creación, gestión y tramitación de expedientes que pertenezcan a distintos procedimientos que, a su vez, se encuentran creados en distintas instancias de Trew@, pudiéndose separar el ámbito de negocio específico de los procedimientos tramitados por la plataforma.

Mediante la evolución a una arquitectura Multi-Trew@ se garantiza una mejor gestión de la carga de usuarios del gestor, al desacoplarse los procedimientos, derivándose a instancias de bases de datos diferentes en función de la familia de procedimientos en cuestión. Esto proporciona un aumento en la mantenibilidad del sistema, ya que se logra separar la información de negocio de las distintas familias de procedimiento, facilitando la gestión y actualización de la información.

Arquitectura Multi-Trew@

Infraestructura tecnológica

Se muestra a continuación diagrama con la infraestructura tecnológica necesaria para disponer de la funcionalidad Multi-Trew@:



Arquitectura Multi-Trew@

Infraestructura tecnológica

Instancias de bases de datos

Se contará con un esquema de base de datos por cada ámbito de negocio o familia de procedimientos que necesite una instancia de Trew@. Cada uno de estos esquemas tendrá sus tablespaces propios y sus usuarios correspondientes, así como todo el juego de datos necesario para una instalación totalmente funcional.

Señalar que es requisito indispensable que todas las instancias de base de datos sean de la misma versión del motor de tramitación Trew@.

Herramienta de Administración Trew@

Para llevar a cabo la administración de todas las instalaciones de Trew@ existentes, se contará con la actual aplicación TrewaAdm que servirá de administración centralizada de todas las bases de datos Trew@ y que irá desplegada como empaquetado .ear sobre un contenedor web WildFly 14.

Dicho contenedor web, tendrá conexión directa con el servidor de base de datos en el que se encuentren todos los esquemas, y tendrá configurados los usuarios de conexión correspondientes.

Plataforma de Tramitación w@ndA

Se dispondrá de una instalación en clúster de PTw@ndA capaz de conectarse con las distintas instancias del motor de tramitación Trew@, correspondientes a cada ámbito de negocio o familia de procedimientos. Por cada base de datos Trew@ será necesario definir un datasource en el servidor de aplicaciones de PTw@ndA.

Motor de indexación Solr

Del mismo modo que existirá una base de datos Trew@ por cada ámbito de negocio o familia de procedimientos, será necesario crear tantos cores de indexación en Solr como instancias de bases de datos Trew@ existan.

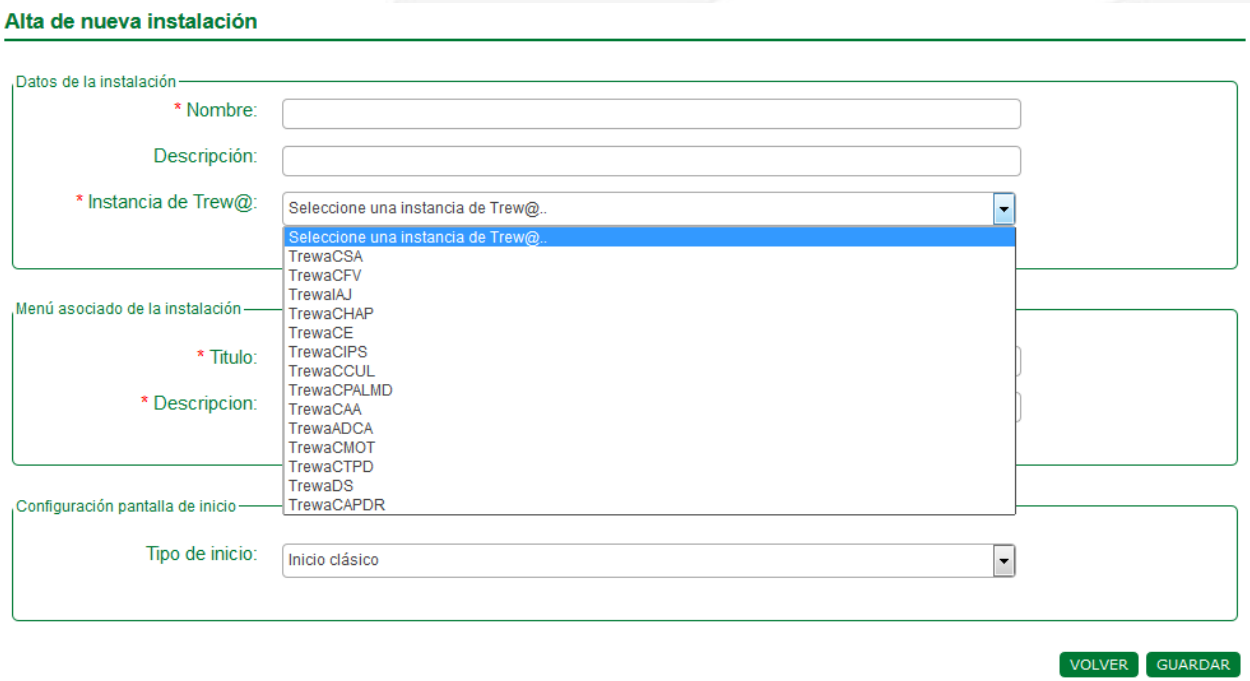
De esta manera los índices generados en Solr para la optimización de los procesos de búsqueda de expedientes estarán desacoplados por ámbito de negocio, no existiendo un único core de indexación que ralentice los procesos de búsqueda.

Arquitectura Multi-Trew@

Configuración arquitectura Multi-Trew@

Se detallan a continuación los pasos a realizar sobre la Plataforma de Tramitación w@ndA para activar el soporte Multi-Trew@:

1. Establecer el parámetro de configuración CONFIGURACION_MULTITREWA_ACTIVADO a true.
2. En el fichero de configuración del servidor de aplicaciones (standalone-ha.xml) será necesario incluir las cadenas de conexión a las diferentes bases de datos de Trew@. Es importante destacar que el nombre de los diferentes jndi deberán tener el prefijo Trewa.
3. Desde la sección de mantenimiento de instalaciones, de la administración de la plataforma, se generará una instalación por cada instancia de Trew@ existente. De este modo existirá una instalación CONTRATACION, otra SUBVENCIONES y así sucesivamente.



Plataforma de tramitación w@ndA

Plataforma de tramitación w@ndA

VEA

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.

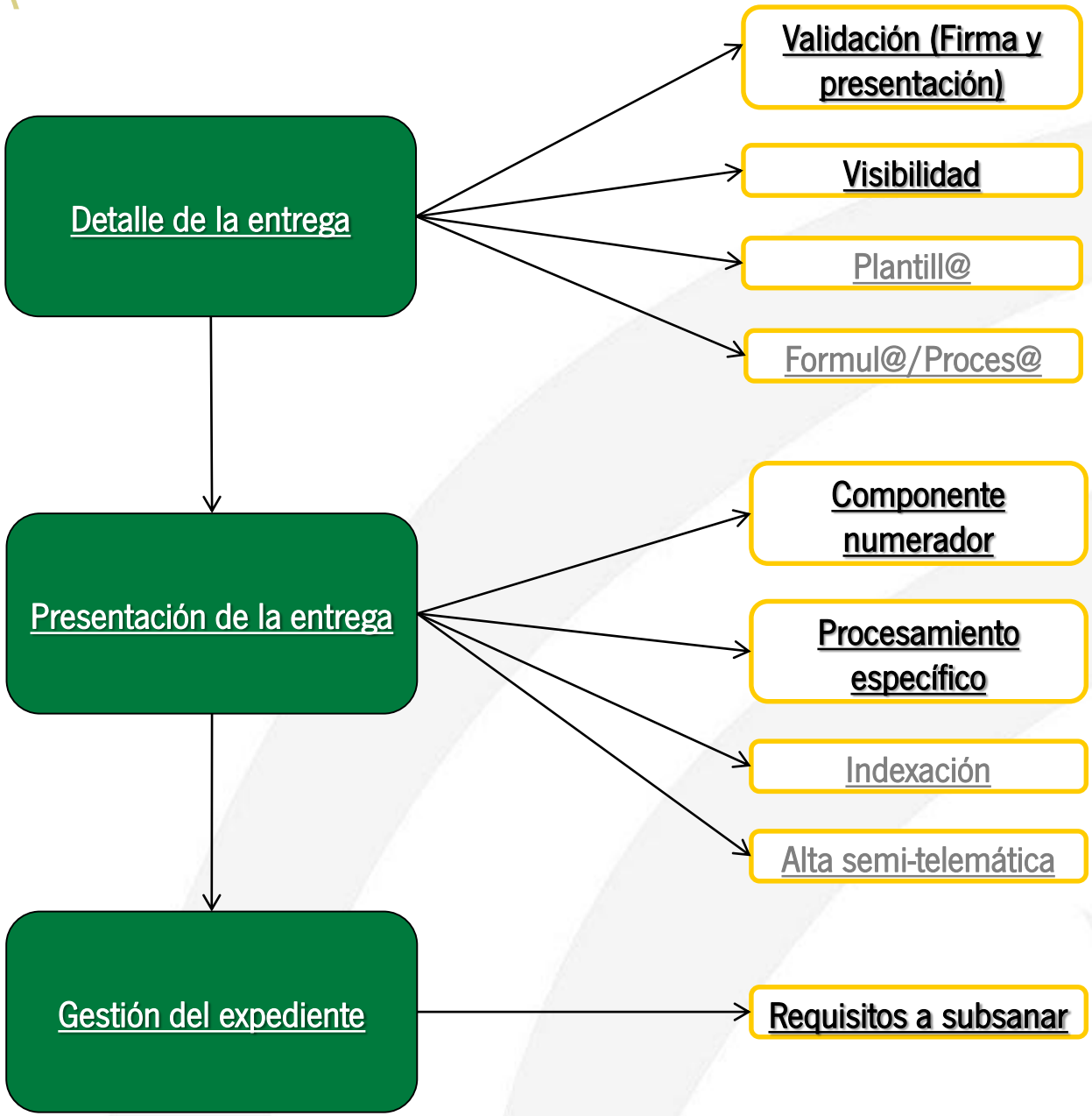
Esta invocación se realiza a través de clientes de servicios web predefinidos e invariables (exceptuando Plantill@, Formul@ y Proces@), de forma que los servicios deben acoplarse con estos clientes.

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.

Plataforma de tramitación w@ndA

VEA



Plataforma de tramitación w@ndA

VEA (Procesamiento específico)

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.

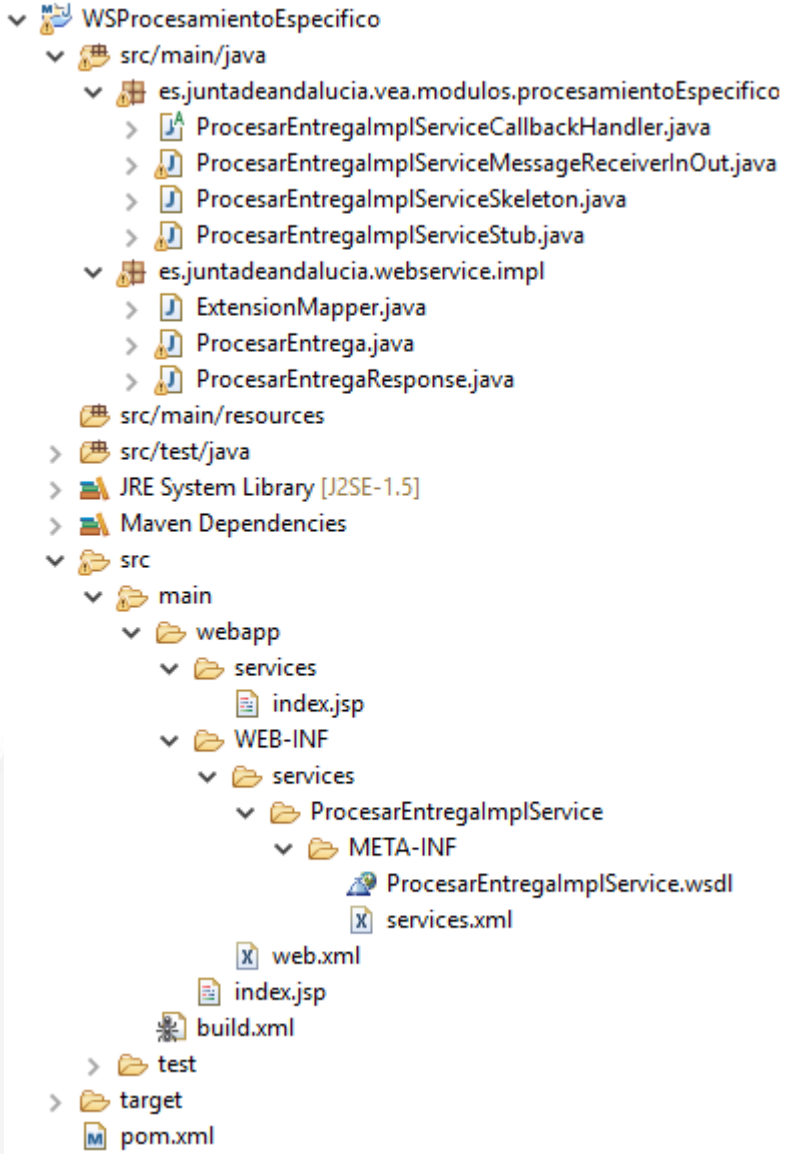
En este caso, el WSDL de referencia se encuentra en la ruta:

[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.

Plataforma de tramitación w@ndA

VEA (Procesamiento específico)



Plataforma de tramitación w@ndA

VEA (Procesamiento específico)

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@.

Plataforma de tramitación w@ndA

VEA (Procesamiento específico)

Implementación de módulo de tipo WS

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-2.0.xsd">

    <bean id="procesamientoEspecificoEndpoint"
        class="es.juntadeandalucia.plataforma.procesamientoEspecifico.ws.ProcesamientoEspecificoEndpoint">
        <constructor-arg ref="jaxb2MarshallerProcesamientoEspecifico" />
        <property name="consultaExpedienteService">
            <ref bean="consultaExpedienteService" />
        </property>
        <property name="tramitacionService">
            <ref bean="tramitacionService" />
        </property>
        <property name="gestionSistemasService">
            <ref bean="gestionSistemasService"/>
        </property>
    </bean>

    <bean id="jaxb2MarshallerProcesamientoEspecifico" class="org.springframework.oxm.jaxb.Jaxb2Marshaller">
        <property name="contextPath"
            value="es.juntadeandalucia.plataforma.procesamientoEspecifico.ws.dto" />
    </bean>

    <!-- Relacion entre mensajes xml y endpoints -->
    <bean
        class="org.springframework.ws.server.endpoint.mapping.PayloadRootQNameEndpointMapping">
        <property name="mappings">
            <props>
                <prop
                    key="{http://impl.webservice.juntadeandalucia.es}procesarEntrega">procesamientoEspecificoEndpoint</prop>
            </props>
        </property>
        <property name="interceptors">
            <bean
                class="org.springframework.ws.server.endpoint.interceptor.PayloadLoggingInterceptor" />
        </property>
    </bean>

    <!-- Generacion automatica del WSDL -->
    <bean id="procesar-entrega"
        class="org.springframework.ws.wsdl.wsdl11.DynamicWsdl11Definition">
        <property name="builder">
            <bean
                class="org.springframework.ws.wsdl.wsdl11.builder.XsdBasedSoap11Wsdl4jDefinitionBuilder">
                <property name="schema"
                    value="/modulos/procesamientoEspecificoWS/xsd/procesamientoEspecifico.xsd" />
                <property name="portTypeName" value="ProcesarEntregaImpl" />
                <property name="requestSuffix" value="" />
                <property name="responseSuffix" value="Response" />
                <property name="locationUri"
                    value="http://localhost:8080/ptwanda-web/WebServices/procesamientoEspecificoWS" />
            </bean>
        </property>
    </bean>
</beans>
```

Plataforma de tramitación w@ndA

VEA (Procesamiento específico)

Implementación de módulo de tipo WS (cont.)

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:pl="http://impl.webservice.juntadeandalucia.es"
    elementFormDefault="qualified" targetNamespace="http://impl.webservice.juntadeandalucia.es">

    <xs:element name="procesarEntrega">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="xmlEntrega" type="xs:string" />
            </xs:sequence>
        </xs:complexType>
    </xs:element>

    <xs:element name="procesarEntregaResponse">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="procesarEntregaReturn" type="xs:string" />
            </xs:sequence>
        </xs:complexType>
    </xs:element>

</xs:schema>
```

Plataforma de tramitación w@ndA

VEA (Procesamiento específico)

Implementación de módulo de tipo WS (cont.)

```
public class ProcesamientoEspecificoEndpoint extends BaseEndPoint {

    @Override
    protected Object invokeInternal(Object obj) {

        ProcesarEntrega peticion = (ProcesarEntrega) obj;
        ProcesarEntregaResponse respuesta = new ProcesarEntregaResponse();
        List<IExpediente> listaExp = new ArrayList<>();

        try {

            String xml = peticion.getXmlEntrega().trim();
            Document document = obtenerDocumentoAPartirDeRecurso(xml);

            // 1) Obtenemos jndi trewa, abreviatura procedimiento, ref
            // expediente
            NodeList nodeJndi = document.getElementsByTagName("JNDI");
            jndiTrewa = nodeJndi.item(0).getFirstChild().getNodeValue();

            NodeList nodeAbrevProc = document
                    .getElementsByTagName("procedimiento");
            abrevProc = nodeAbrevProc.item(0).getFirstChild().getNodeValue();

            NodeList nodeRefExpediente = document
                    .getElementsByTagName("numero-expediente");
            refExpediente = nodeRefExpediente.item(0).getFirstChild()
                    .getNodeValue();

            NodeList nodeSistema = document.getElementsByTagName("sistema");
            sistema = nodeSistema.item(0).getFirstChild().getNodeValue();

            // 2) Obtenemos los valores de los campos del formulario
            String clave = null;
            String val = null;

            NodeList listaFormularios = document
                    .getElementsByTagName("formulario");

            Node formulario = listaFormularios.item(0);

            NodeList listaNodosFormulario = formulario.getChildNodes();

            // Recorremos los hijos de la etiqueta formulario
            for (int j = 0; j < listaNodosFormulario.getLength(); j++) {

                Node hijo = listaNodosFormulario.item(j);

                // Sólo nos interesan los nodos de tipo campo
                if (hijo.getNodeName().equals("campo")) {

                    // Por cada campo
                    NodeList listaNodosCampo = hijo.getChildNodes();
                    clave = null;
                    val = null;
```

Plataforma de tramitación w@ndA

VEA (Procesamiento específico)

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

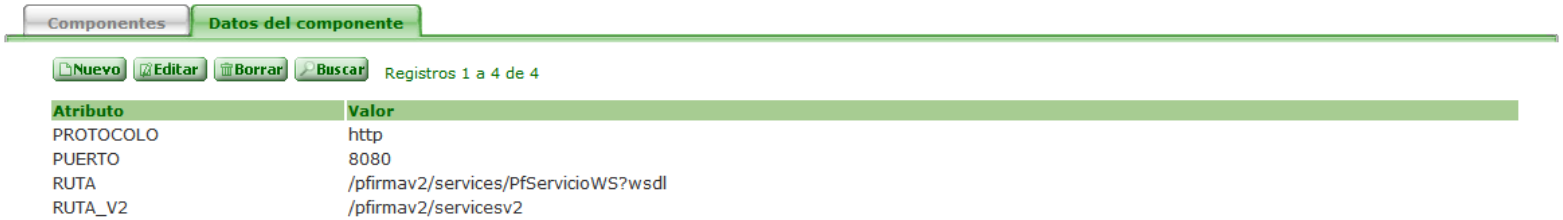
En este punto se debe establecer el procesamiento mediante servicio web, y establecer del mismo modo la Ruta donde se ha desplegado el Servicio Web:

Integración con sistemas externos

Port@firmas

Las configuración del componente Port@firmas se realiza sobre la instalación que se utilice de Trew@, habilitando el componente en cada uno de los sistemas a utilizar y estableciendo los parámetros adecuados de conexión. Esta configuración podrá ser realizada haciendo uso de la herramienta de administración de Trew@ (TrewaAdm).

Nombre	Descripción	Dirección IP	Usuario	Cód.w@ndA	Organismo donde está
@FIRMA	COMPONENTE @FIRMA PARA FIRMAR DOCUMENTOS				Consej. Hacienda y Admón. Púb. - (SEVILLA)
NOTIFIC@	COMPONENTE TREW@ DE NOTIFICACIONES	ws031.juntadeandalucia.es			Consej. Hacienda y Admón. Púb. - (SEVILLA)
PORT@FIRMAS	COMPONENTE PORT@FIRMAS	7.128.80.43			Consej. Hacienda y Admón. Púb. - (SEVILLA)
TREW@	COMPONENTE DE TRAMITACIÓN TREW@	NOMBRE_SERVIDOR			Consej. Hacienda y Admón. Púb. - (SEVILLA)
@VISADOR	COMPONENTE @VISADOR	7.128.80.16			Consej. Hacienda y Admón. Púb. - (SEVILLA)
WEBOFFICE	COMPONENTE WEBOFFICE				Consej. Hacienda y Admón. Púb. - (SEVILLA)



Atributo	Valor
PROTOCOLO	http
PUERTO	8080
RUTA	/pfirmav2/services/PfServicioWS?wsdl
RUTA_V2	/pfirmav2/servicesv2

Integración con sistemas externos

@firma

Las configuración del componente @firma se realizará desde la administración de la propia PTw@ndA. Se debe configurar los siguientes parámetros de configuración:

- **SERVIDOR_FIRMA:** IP o nombre DNS del servidor @firma.
- **ID_APLICACION_FIRMA:** Aplicación @firma a utilizar.
- **VERSION_FIRMA:** La versión de @firma.
- **USUARIO_FIRMA:** El nombre de la aplicación dada de alta en @firma.
- **LOGIN_FIRMA:** El password de la aplicación dada de alta en @firma.

Integración con sistemas externos

@ries

Las configuración del componente @ries se realizará desde la administración de la propia PTw@ndA. Se debe configurar los siguientes parámetros de configuración:

- **REGISTRO_USUARIO:** Usuario dado de alta en @ries.
- **REGISTRO_PASSWORD:** Password del usuario dado de alta en @ries.
- **REGISTRO_ASUNTO:** Código de asunto.
- **REGISTRO_CODDESTINO:** Código @ries destino.
- **REGISTRO_CODORIGEN:** Código @ries origen.
- **REGISTRO_URL:** Url del servicio Web utilizado para realizar el registro de salida.

Se hará uso tanto del parámetro REGISTRO_CODORIGEN como del REGISTRO_CODDESTINO si no existe un código @ries asociado al organismo origen o y/o al organismo destino respectivamente, configurado en el motor de tramitación Trew@.

Integración con sistemas externos

SCSP

La plataforma de tramitación w@ndA se integra con el servicio de **Supresión de Certificados en Soporte Papel**, permitiendo la consulta de los siguientes certificados:

- Certificado de datos de identidad.
- Certificado de datos de familia numerosa.
- Certificado de datos de residencia.
- Certificado de datos de discapacidad.
- Certificado de estar al corriente de pago con la Seguridad Social.

Para la correcta integración con los servicios SCSP es necesario realizar la siguiente configuración:

1. Creación de los siguientes parámetros de configuración:
   - **USUARIO_SCSP:** Usuario para conectarse a los servicios SCSP.
   - **CLAVE_SCSP:** Contraseña para conectarse a los servicios SCSP.

Integración con sistemas externos

SCSP

2. Creación de un tipo de certificado en Trew@ por cada servicio scsp a consultar. Para ello desde la opción **'Tipos de certificados'** se crean los 4 tipos consultables:

Tipos de certificado

Nuevo | Editar | Borrar | Buscar Registros 1 a 2 de 2

Nombre	Descripción	URL de consulta	Clase
minusvalia	certificado de minusvalia	com.everis.certs.minusvalia	CertificadoMinusvalia
familia numerosa	certificado de familia numerosa	com.everis.certs.familia_numerosa	FamiliaNumerosa

- **Nombre:** Campo identificativo para el tipo de certificado
- **Descripción:** Descripción del tipo de certificado
- **URL de consulta:** URL de consulta del certificado
- **Clase:** Clase que llama al tipo de certificado

Integración con sistemas externos

SCSP

El campo clase de los cuatro certificados es:

a. Certificado de identidad:
es.juntadeandalucia.plataforma.consultaSCSP.consultaIdentidad.ConsultaDatosIdentidad

b. Certificado de residencia:
es.juntadeandalucia.plataforma.consultaSCSP.consultaResidencia.ConsultaDatosResidencia

c. Certificado de familia numerosa:
es.juntadeandalucia.plataforma.consultaSCSP.consultaFamiliaNum.ConsultaDatosFamiliaNum

d. Certificado de discapacidad:
es.juntadeandalucia.plataforma.consultaSCSP.consultaDiscapacidad.ConsultaDatosDiscapacidad

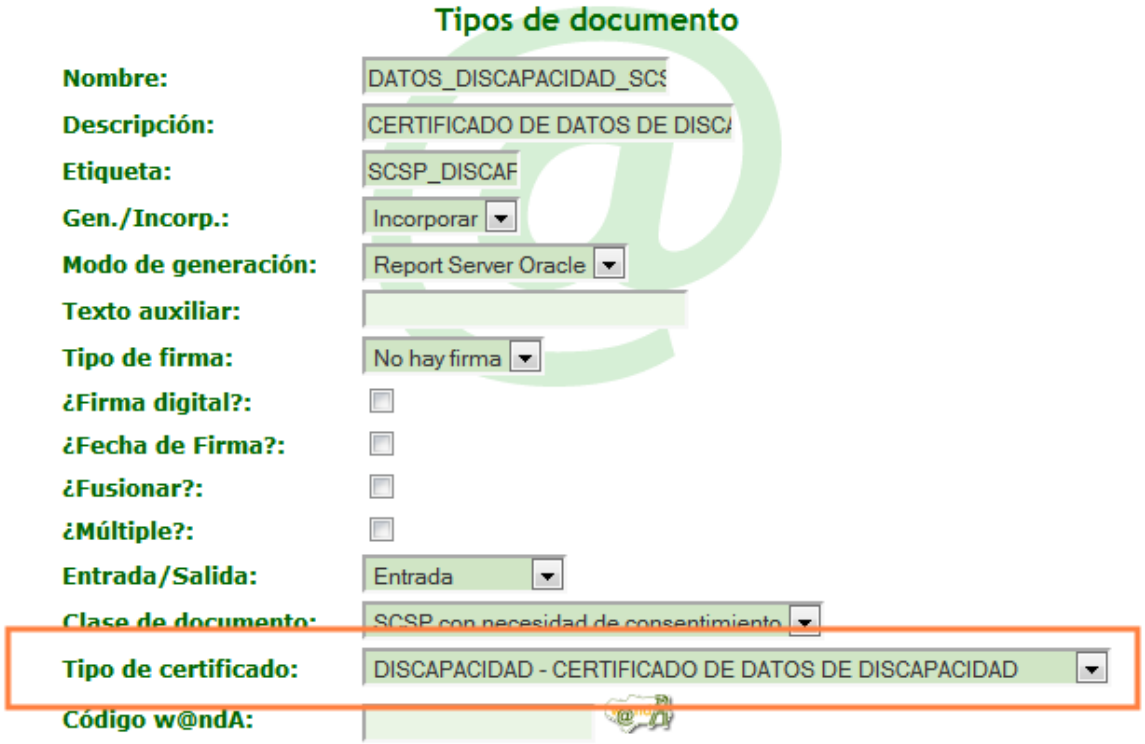
e. Certificado de estar al corriente de pago con la Seguridad Social

es.juntadeandalucia.plataforma.consultaSCSP.consultaCorrientePagosTGSS.ConsultaPagosTGSS

Integración con sistemas externos

SCSP

3. Asociar cada tipo de certificado al tipo de documento de clase scsp.



Para el correcto funcionamiento de las consultas a los certificados es necesario que la máquina virtual de java tenga las fuentes Verdana y Arial. Estas fuentes deben incluirse en el directorio jre/lib/font de la JVM.

Integración con sistemas externos

SCSP

Finalizado el proceso de configuración para la correcta integración con los servicios SCSP es necesario realizar la instalación de los módulos funcionales SCSP, proporcionados en la carpeta de distribución de la versión correspondiente.

En los procedimientos en los cuales se desee realizar una consulta a un determinado tipo de certificado será necesario modelar un documento permitido en la fase del procedimiento donde se llevará a cabo la consulta.

Integración con sistemas externos

Notific@

La Plataforma de Tramitación w@ndA proporciona un módulo de notificaciones telemáticas, mediante el cual podrán realizarse y gestionar todos los datos de las diferentes notificaciones realizadas en un determinado procedimiento administrativo.

Este módulo permite al usuario con los permisos adecuados que se encuentre tramitando un expediente, emitir los diferentes tipos de notificaciones, registrar sus datos de recepción e incluso recoger los datos de una posible publicación.

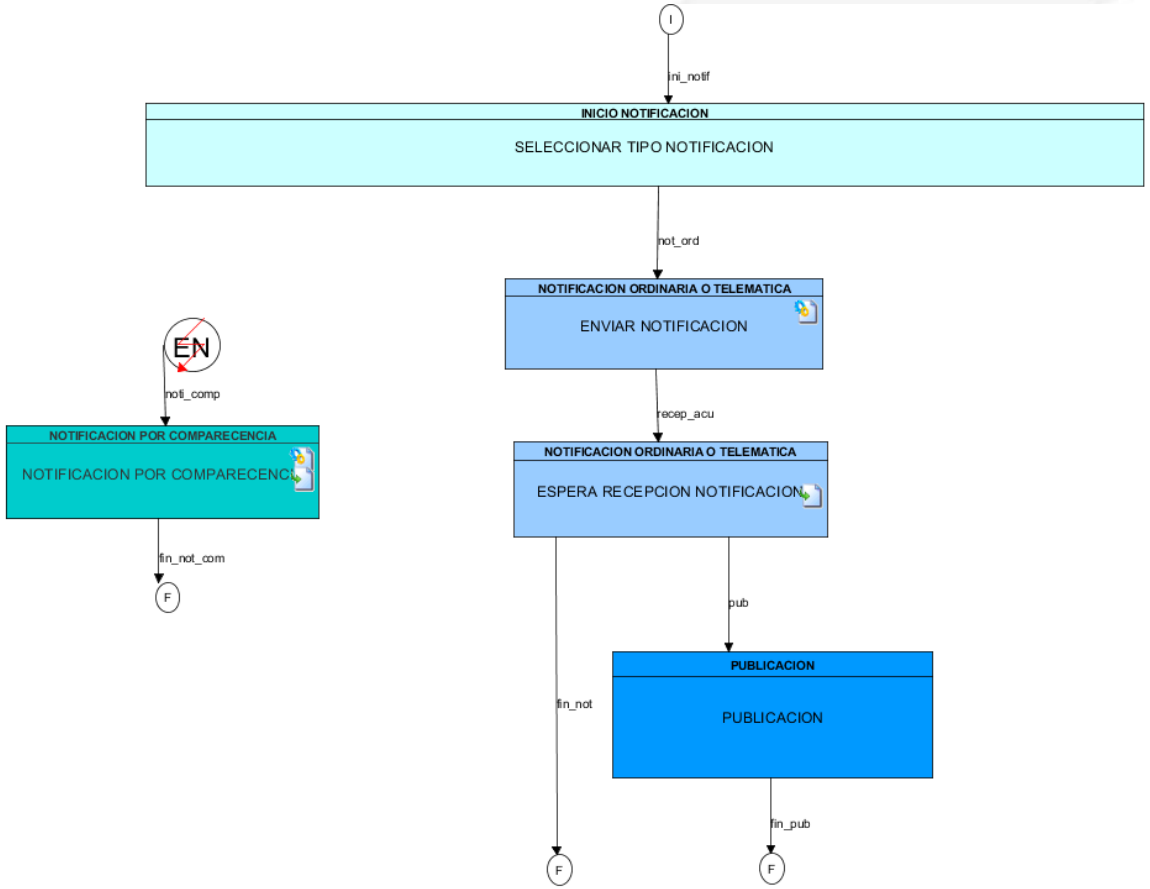
A su vez, este módulo pretende servir de consulta a todos los usuarios de tramitación con el objetivo de poder consultar las fechas de emisión, de acuse de recibo o el estado de las mismas. Esta gestión y control de notificaciones, incluye notificaciones ordinarias postales, telemáticas y por comparecencia.

El módulo de "Gestión de Notificaciones" se desarrolla como una utilidad presente en el escritorio de tramitación, al que cualquier usuario tramitador tendrá acceso para consultar los datos relativos al envío y recepción de notificaciones del expediente. Sin embargo, tan sólo los usuarios con un permiso o perfil determinado (PF_NOTIF_TECNICO) podrán realizar en él todas las funcionalidades anteriormente comentadas.

Integración con sistemas externos

Notific@

El módulo de notificaciones se basa en un flujo de tramitación formado por un procedimiento reutilizable que recoge las fases del procedimiento administrativo contempladas en el proceso de notificación. Este procedimiento reutilizable será replicado en el flujo principal de tramitación de cualquier procedimiento, allí donde proceda realizar la notificación de un determinado tipo de documento.



Integración con sistemas externos

Notific@

Para el correcto funcionamiento del módulo de notificaciones es necesario realizar la configuración del componente Notific@ en el motor de tramitación Trew@.

El componente se debe definir con los siguientes datos:

- Dirección IP.
- Nombre de la máquina o dirección IP en la que se encuentra Notific@
- Por ejemplo: **notifica01.chap.junta-andalucia.es**

A este componente se le deben añadir los siguientes datos de componente:

Integración con sistemas externos

Notific@

Obligatorios

PROTOCOLO	Protocolo para la comunicación con los servicios web de Notific@.	https
RUTA	Ruta para la comunicación con los servicios web de Notific@.	jboss-net/services/ServicioWEBSN
PUERTO	Puerto para la comunicación con los servicios web de Notific@.	443
PKCS12.ARCHIVO	Ruta del fichero PKCS#12 para firmar las solicitudes SOAP enviadas.	../deploy/ptwanda-web-2.1.1r01.war/WEB-INF/classes/PruebasNotif-cert-final.p12
PKCS12.PASS	Contraseña de acceso a la clave privada PKCS#12.	12345678
XML_LOG	Ruta del fichero de log.	../server/all/log/mcneLog4jConfig.xml
CODIGO_SERVICIO	Identificador de código de servicio al cual se enviarán las notificaciones	87, 2900

Integración con sistemas externos

Notific@

Opcionales

PKCS12.PASS.METODO	Metodo de obtención de la contraseña. Esta se podrá obtener a través del atributo pkcs12.pass o desde el método getPrivateKey() de una clase que implemente el interfaz IKeySec. Ver propiedad pkcs12.clasepwd.	• clase • propiedad
PKCS12.PASS.CLASEPWD	Clase que cumple el interfaz IKeySec y que permite implementar el método getPrivateKey(), encargado de obtener de una fuente segura la contraseña.	mi.paquete.seguridad.PassPrivateKey

Servidor proxy (solo si se utiliza servidor proxy de acceso a Notific@)

CONEXION_PROXY	Indica si se hará uso de servidor proxy o no para realizar la conexión al componente Notific@.	true, false
HOST_PROXY	Nombre de la máquina o dirección IP del servidor proxy que se va a utilizar.	271.125.9.10, proxy-notifica
PUERTO_PROXY	Puerto utilizado para realizar la conexión con el servidor proxy.	8080
USER_PROXY	Usuario de acceso al servidor proxy.	User
PASS_PROXY	Password de acceso al servidor proxy.	Pass

Plataforma de tramitación w@ndA

Novedades de las últimas versiones

Versión 2.4.2

- Módulo SCSP servicio de consulta de estar al corriente de pago con la Seguridad Social.
- Generación de componente de integración con el gestor documental Alfresco compatible con el conector trewa-contentManager proporcionado por el motor de tramitación Trew@.
- Evolución de la operación consulta de terceros a su versión 1.5 y comprobación de su correcta compatibilidad.
- Evolución del módulo de alta de expedientes para incorporar las siguientes funcionalidades:
  - ❑ Validación formularios: incorporación de lógica de validación de datos de aquellos campos obligatorios para el motor de tramitación Trew@.
  - ❑ Gestión de interesados: mecanismo de procesado por defecto de los datos delformulario, realizándose el alta de los interesados en el motor de tramitación en funciónde los datos cumplimentados en el formulario.
  - ❑ Procesamiento específico: permitir la configuración de un servicio web de procesamiento específico, servicio al cual se le enviará un fichero xml con los datos del expediente(formularios y documentos adjuntos).
- Evolución del módulo funcional de documentos asociados para permitir la edición de los datos código de registro y fecha de registro, en el caso de asientos registrales de entrada.

Novedades de las últimas versiones

Versión 2.4.3

- Migración del componente para permitir el despliegue del mismo en el servidor de aplicaciones WildFly versión 14.0.1 y Java 8.
- Actualización del API de Entidades Emisoras existente a su versión 3.3.1 y comprobación de su correcta compatibilidad.
- Evolución de la actual sección de observaciones de los documentos para incluir la consulta delos metadatos mínimos obligatorios del documento electrónico conforme a lo dispuesto en el Esquema Nacional de Interoperabilidad.
- Evolución de la actual utilidad de relación de expedientes para mostrar información más completa sobre los expedientes hijo, así como la consulta de los datos básicos del expediente padre.
- Se permite la incorporación de un documento alojado en un repositorio externo, como referencia, a partir de su Código Seguro de Verificación (CSV).
- La versión 2.4.3 de PTw@ndA permite la incorporación de varios documentos simultáneamente de manera convencional, esto es, haciendo uso del botón para inspeccionar el equipo y seleccionar los documentos deseados. Además, también es posible incorporar varios documentos utilizando el drag & drop mediante el cual, se arrastran los ficheros a la zona habilitada para ello.
- Obtención de interesados asociados a un expediente permitiéndose indicar el conjunto de razones de interés que no quieren ser mostradas en el portlet y la utilidad delos interesados, mediante la configuración de un nuevo parámetro de configuración RAZONES_INTERESADOS_OCULTAR.
- Evolución de la zona de datos básicos del expediente del escritorio de tramitación para mostrar su código seguro de verificación CSV en HCV

Novedades de las últimas versiones

Versión 2.5.0 (en desarrollo)

- Integración con el componente LibreOffice Online, el cual sustituirá a los componentes WebOffice y editor de textos web, ofreciéndose vía web de un editor completo de documentos.
- Implementación de un buscador avanzado de expedientes que permita realizar búsquedas por campos específicos del propio negocio.
- Nueva arquitectura de despliegue de módulos funcionales, desacoplando la integración de los módulos del propio núcleo de la plataforma.
- Actualización del núcleo del motor de indexación y búsqueda, Solr de Apache Lucene, a la última versión 8.0.0.
- Discontinuación del componente Applet de Model@, ofreciéndose una nueva alternativa basada en JavaScript.
- Evolución del módulo funcional de gestión de notificaciones, que gestiona la remisión de notificaciones administrativas a los interesados, añadiendo funcionalidades como la posibilidad de enviar más de un documento por remesa.
- Integración SCSP con el "Servicio de Consulta de Certificación de Titularidad" ofrecido por el Catastro para la certificación de de titularidad de un bien inmueble.
- Evolución del envío de petición de firma, permitiéndose seleccionar documentos no firmados asociados al expediente como documentación anexa a la petición de firma.
- Automatización de la actualización del estado de los documentos pendientes de firma y del estado de las notificaciones electrónicas enviadas.

Gracias por su atención.