JUNTA DE ANDALUCIA

CONSEJERÍA DE JUSTICIA Y ADMINISTRACIÓN PÚBLICA

Plataforma de Tramitación w@ndA

Manual del Desarrollador

13/10/2008

JUNTA DE ANDALUCIA
CONSEJERÍA DE JUSTICIA Y ADMINISTRACIÓN PÚBLICA

Índice

1. Introducción a la Tramitación con PT_w@ndA

Uno de los primeros objetivos que se pretende conseguir en el marco del Proyecto w@ndA, es la definición de las **Guías de Tramitación de Familias de Procedimientos Administrativos** en la Junta de Andalucía.

Estas Guías contendrán un esquema simplificado de los **procedimientos administrativos**, incluyendo su tramitación, documentación de entrada y de salida, información básica para la tramitación e información a proporcionar al ciudadano. Además, permitirán normalizar elementos comunes de los procedimientos y servirán de base para las Guías de Tramitación de Procedimientos.

Es por ello, que cobra gran importancia la definición de las familias de procedimientos, entendiendo por tales la "agrupación no arbitraria de procedimientos bajo el criterio de similitud en el esquema básico de tramitación, documentación de entrada y salida e información".

Las familias se han agrupado, basándose fundamentalmente en el criterio de similitudes en la tramitación, dejando a un lado los criterios de semejanza en la materia objeto del procedimiento, órgano competente, etc.

La metodología seguida para la definición inicial de la clasificación de familias de procedimientos consistió en seguir los siguientes pasos:

- ***Recopilación de información:*** *inventario de procedimientos, legislación estatal y autonómica, derecho administrativo)*
- ***Definición de familias***
- ***Análisis y ajustes hasta su validación final***

JUNTA DE ANDALUCIA
CONSEJERÍA DE JUSTICIA Y ADMINISTRACIÓN PÚBLICA

Objetivo

El objetivo fundamental para la elaboración de las Guías de Tramitación es recoger toda la información necesaria y suficiente para la implementación de los procedimientos en una plataforma de tramitación electrónica para su tramitación electrónica.

Las Guías de Tramitación deben elaborarse :

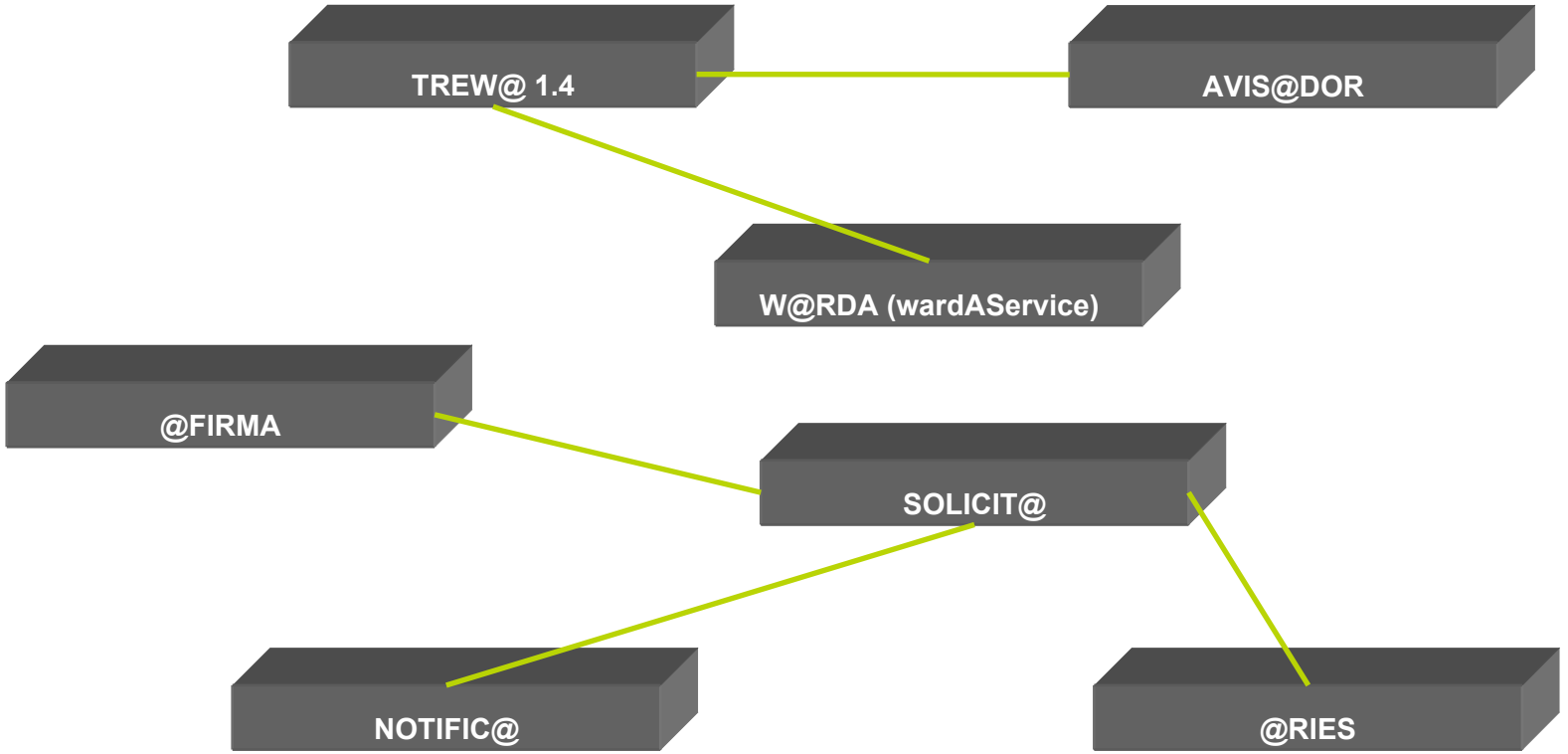
- ***siguiendo el principio de simplificación de trámites administrativos,***
- ***eliminando los trámites innecesarios,***
- ***y de normalización de elementos comunes de los procedimientos,***
- ***homogenizando aquello que sea posible.***

Directrices en el diseño de PT_w@ndA

- La plataforma será 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.
- Minimizar al máximo las horas de programación necesarias para implantar una solución de tramitación de expedientes en el mundo *w@ndA.*
- Su arquitectura debe estar totalmente alineada con los componentes del proyecto *w@ndA* de la Junta de Andalucía. Integrándolos y garantizando un uso correcto y controlado de cada uno de ellos.
- La plataforma, bajo configuración, será parametrizable en cuanto a los componentes que se desean utilizar en cada implantación.
- Actualizada con respecto a las diferentes tecnologías empleadas en el diseño de su arquitectura.
- Su arquitectura debe soportar ampliaciones y nuevas funcionalidades.
- Funcionalmente deberá poder ser ampliable en base a la instalación de nuevos componentes funcionales construidos bajo unas especificaciones perfectamente definidas. Estos componentes funcionales darán cobertura a funciones específicas de procedimientos concretos, como podría ser el módulo de baremación para la tramitación de subvenciones.

1. Introducción a la Tramitación con PT_w@ndA (IV)

El Proyecto w@ndA SIN Plataforma de Tramitación



1. Introducción a la Tramitación con PT_w@ndA (V)

El Proyecto w@ndA CON Plataforma de Tramitación

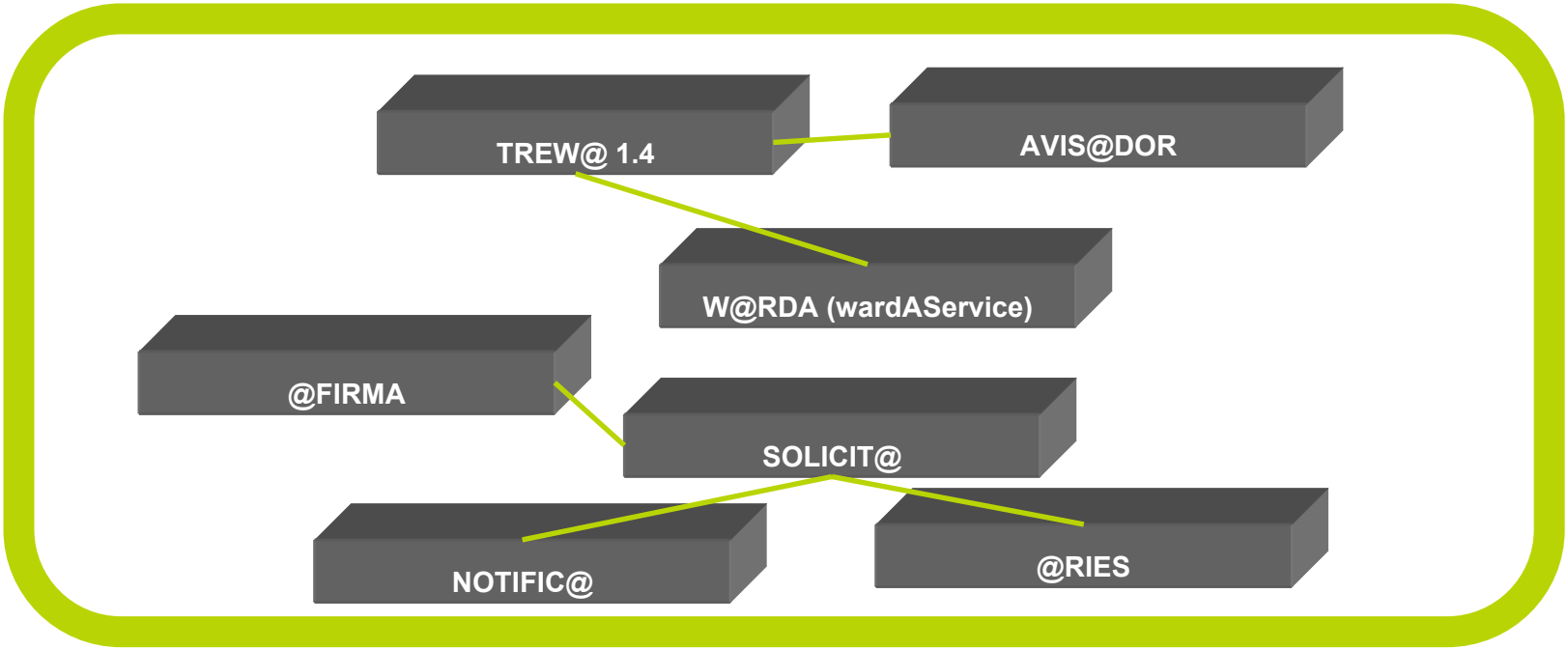
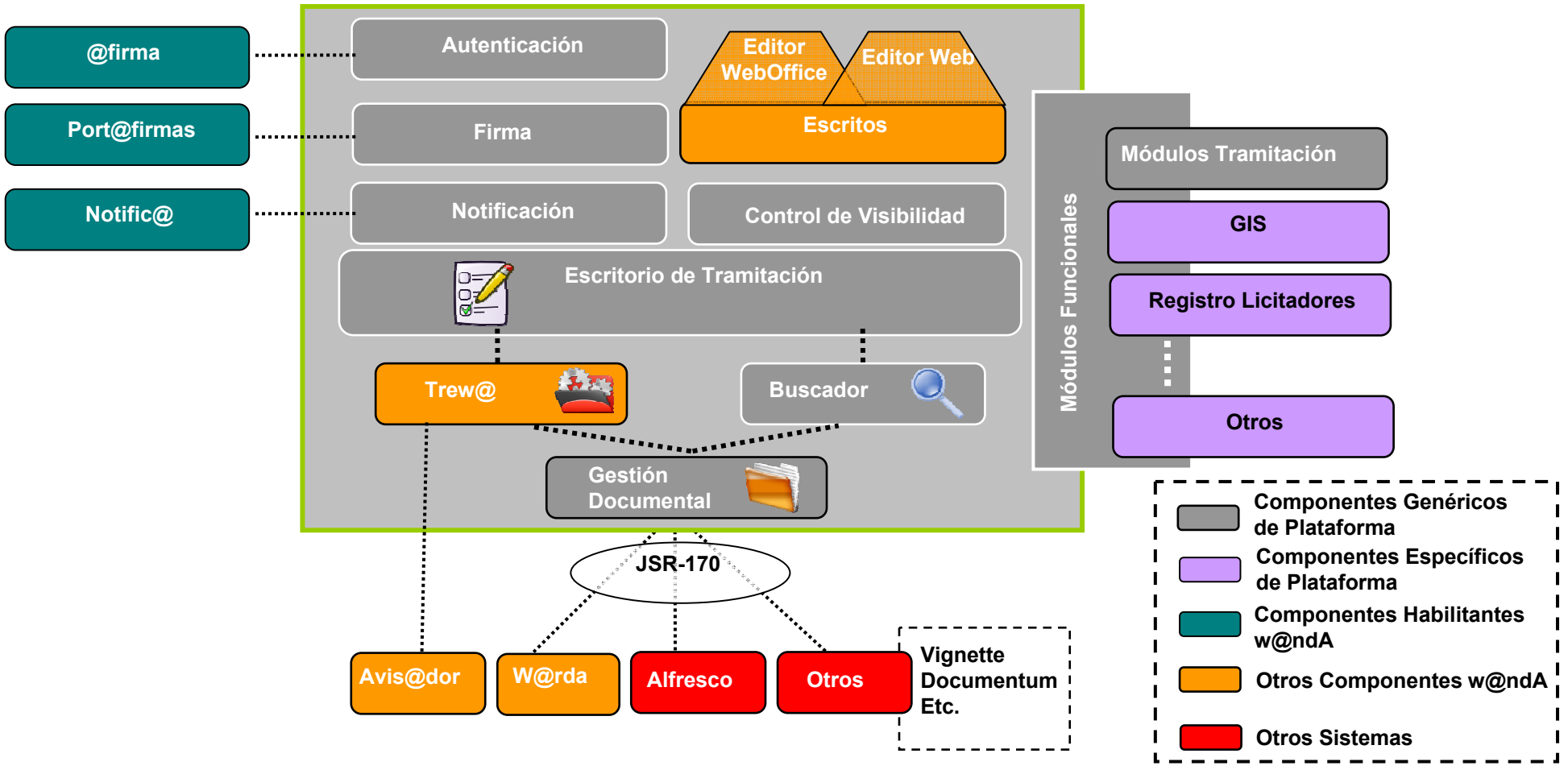


Diagrama de Componentes



JUNTA DE ANDALUCIA
CONSEJERÍA DE JUSTICIA Y ADMINISTRACIÓN PÚBLICA

El tramitador de procedimientos Trew@

- Herramienta **workflow**
- Gestionar procedimientos administrativos
- Coordinar actividades y usuarios participantes en el proceso

Ventajas

- Automatiza tareas y pasos a dar en los procedimientos.
- Ayuda a tener procedimientos homogéneos ➡ optimización y eficiencia.
- Facilita el trabajo ofreciendo a los usuarios participantes qué hacer y en qué momento ➡ usuarios noveles con escaso conocimiento del negocio.
- Mantenimiento sin modificar código ➡ gracias a herramienta gráfica de modelado y módulo de administración.

JUNTA DE ANDALUCIA
CONSEJERÍA DE JUSTICIA Y ADMINISTRACIÓN PÚBLICA

Conceptos Básicos

Procedimiento:

- conjunto de actividades relacionadas
- conforme a unas reglas preestablecidas
- ejecutadas por uno o varios agentes
- cada materialización de un procedimiento es lo que denominamos "**Expediente**"

Tramitador

- componente software que permite definir, implementar e instanciar procedimientos

Perfil de Usuario

- usuario o conjunto de usuarios participantes en un procedimiento

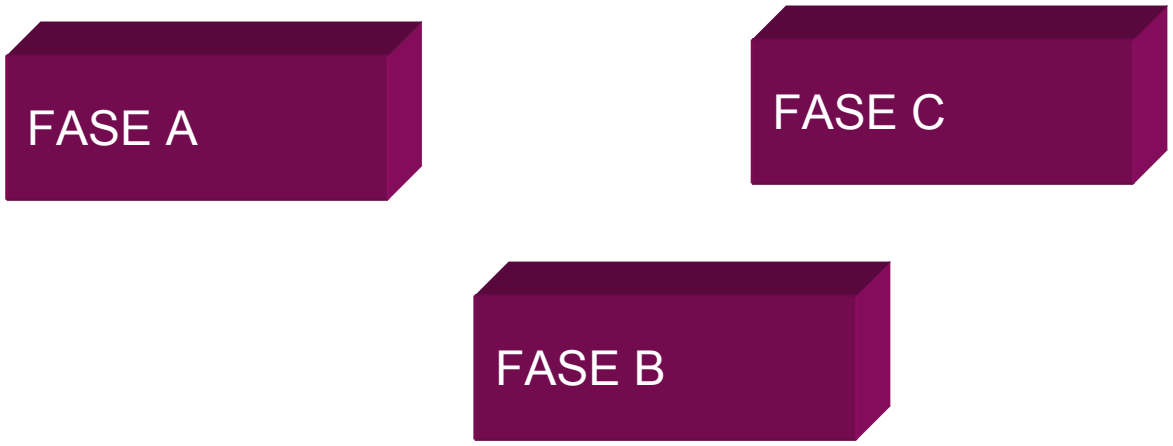
Modelado de procedimientos

- definición formal de un procedimiento real

2. Introducción al Modelado de Procedimientos (III)

Fase: se llama fase a un conjunto homogéneo de tareas desde el punto de vista del tramitador, cuya sucesión en el tiempo componen un determinado procedimiento una vez ha sido modelado y constituyen la unidad elemental de tramitación dentro del mismo.

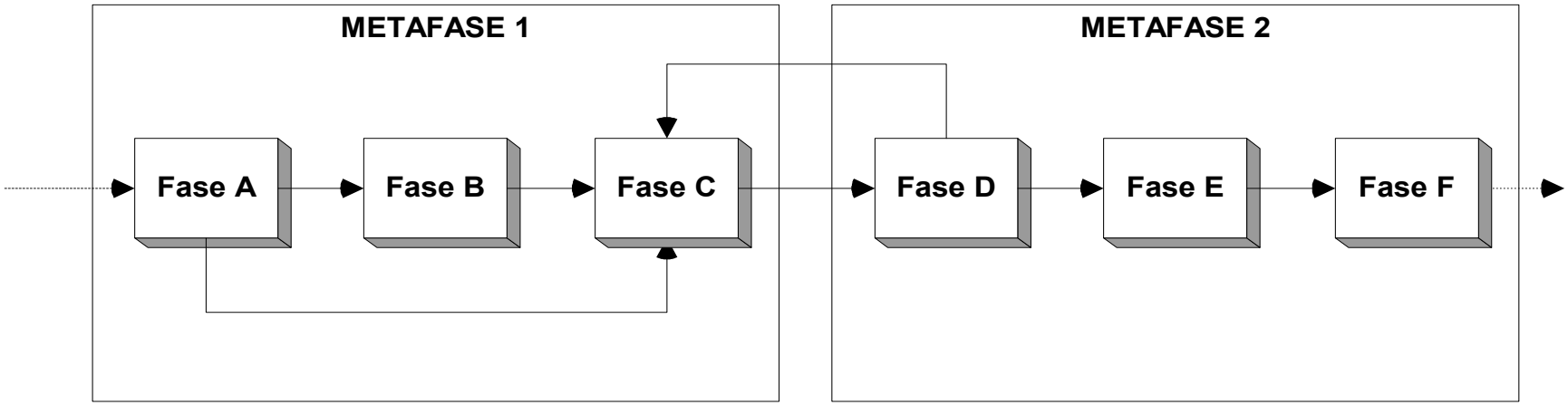
- Cada una de los pasos que componen un procedimiento. Unidad elemental de tramitación
- Tienen duración en el tiempo
- El expediente “esta en...”
- Tipos: Tramitación, entrada/salida, externas



2. Introducción al Modelado de Procedimientos (IV)

Metafase: conjunto de fases dentro de un procedimiento que comparten una serie de características comunes que permiten al Tramitador dar un tratamiento diferenciado del resto pero común a todas ellas.

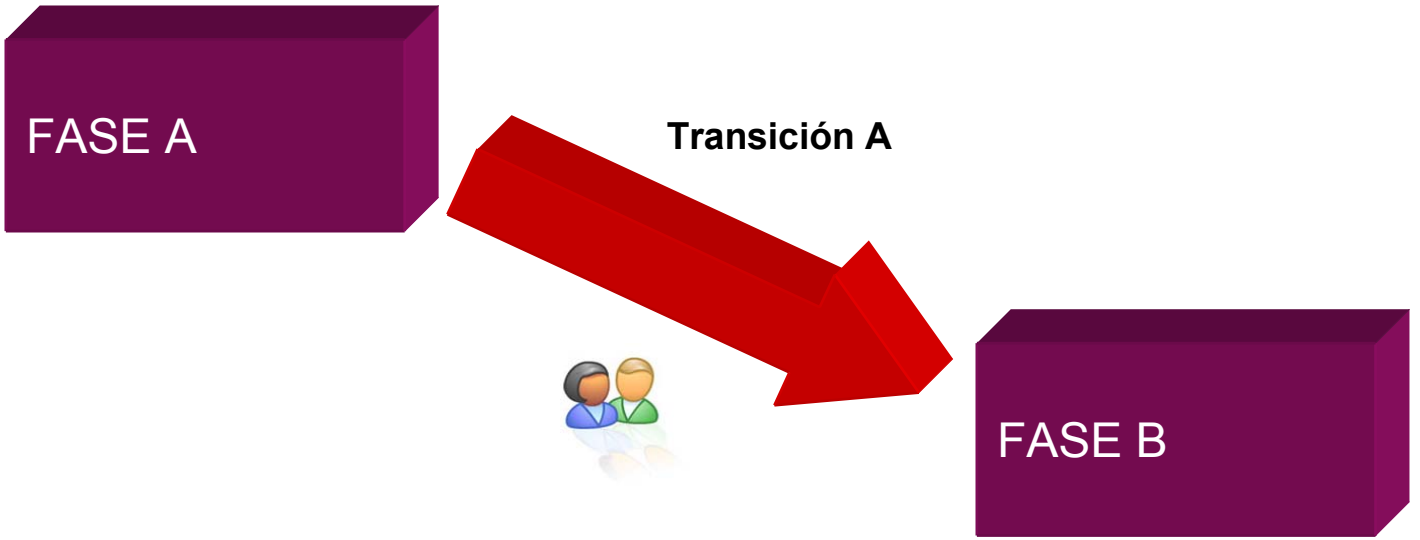
- Agrupación de fases homogéneas desde el punto de vista administrativo
- facilita la visión global del conjunto
- permite detallar más las fases



2. Introducción al Modelado de Procedimientos (V)

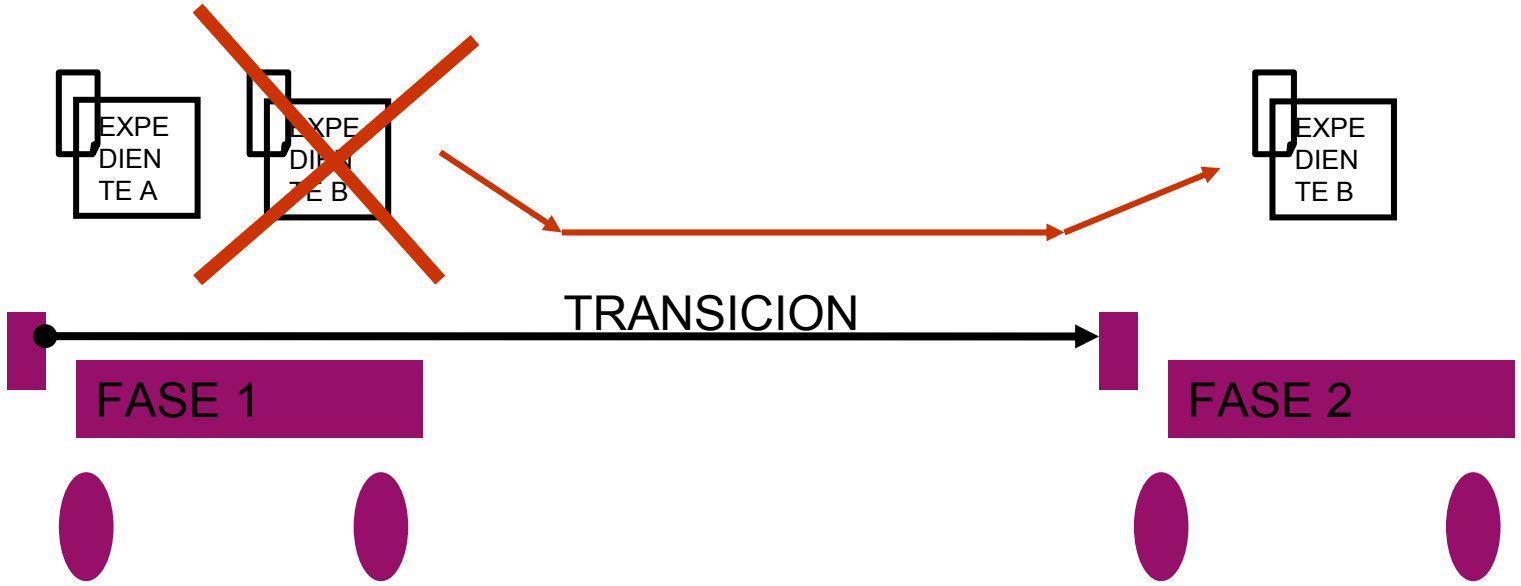
Transición: hito que provoca el paso de una fase a otra dentro de un determinado procedimiento. Una transición hace que un determinado expediente evolucione de una fase a otra según la definición del procedimiento.

- hito que provoca el paso de una fase a otra
- tienen duración “cero” (“fecha de ....”)
- tipos: E/S, Actos, Decisiones y Plazos



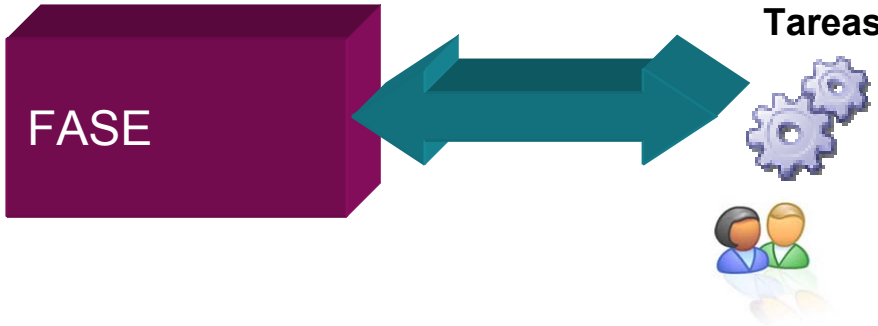
Fases y Transiciones

En Trew@ un expediente es una instancia de un procedimiento que puede encontrarse en una o varias FASES y que evoluciona de una a otra mediante TRANSICIONES.



2. Introducción al Modelado de Procedimientos (VII)

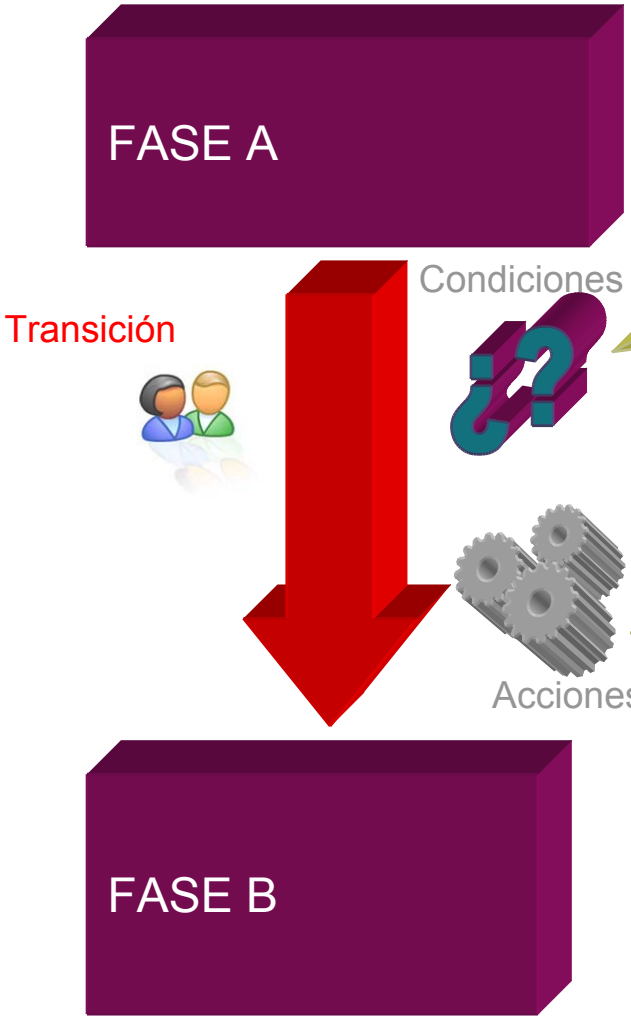
Tarea: unidades elementales de trabajo que pueden/deben realizarse en una determinada fase de un procedimiento por un usuario participante en el mismo. La especificación de qué tareas deben realizarse y quién debe ejecutarlas forma parte del modelado de procedimientos y por tanto queda especificado en la definición del mismo.



Tareas

- unidad elemental de trabajo dentro de una fase
- asignada a uno o varios perfiles de usuario
- una o varias con o sin orden establecido
- Tipos:
  - Introducción de datos
  - Manipular escritos
  - Instanciar otros procedimientos

2. Introducción al Modelado de Procedimientos (VIII)

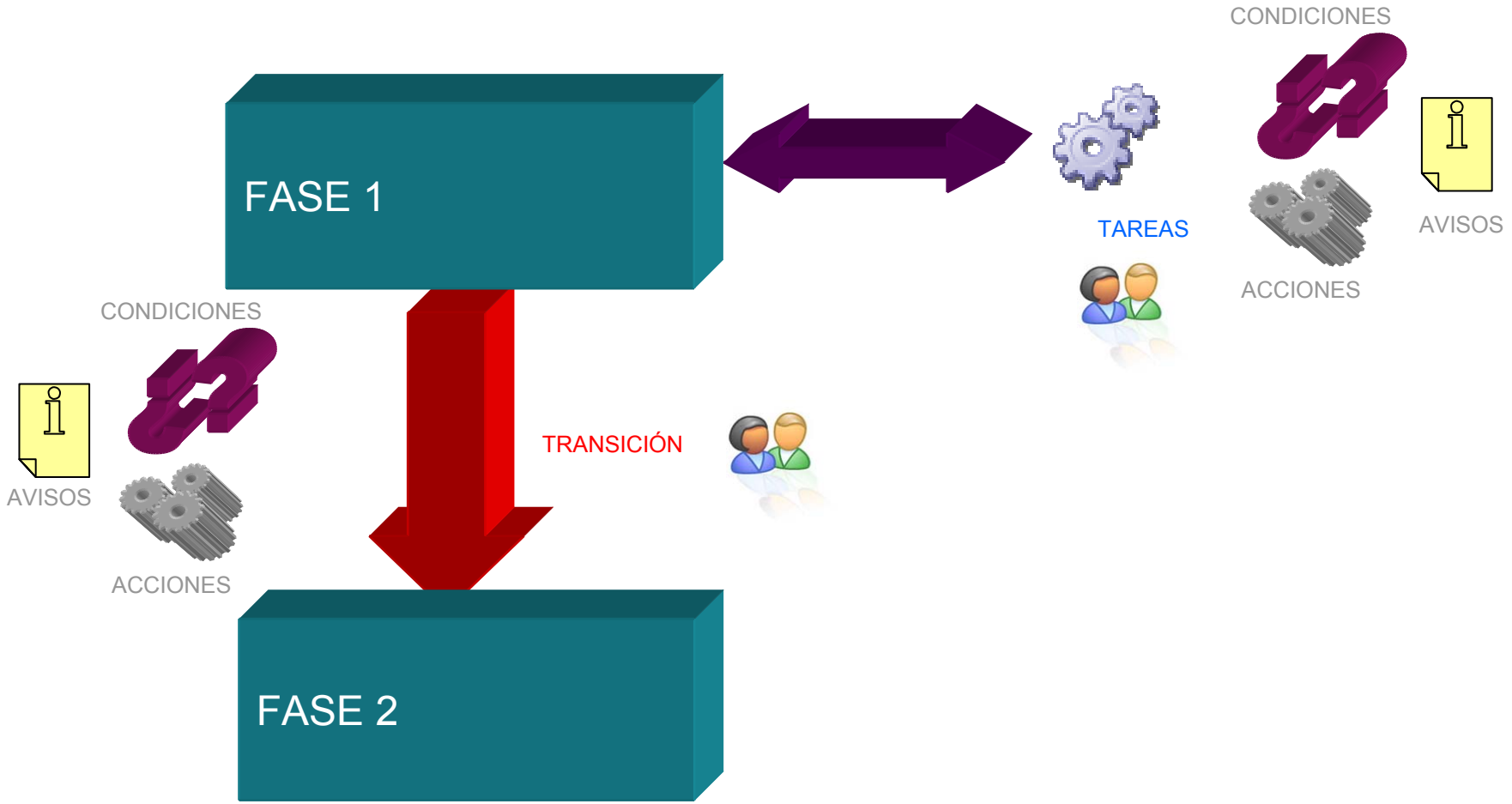


- circunstancia que se evalúan cuando se produce una transición
- pueden ser obligatorias (impiden que la transición se de si no se cumple) u opcionales (genera avisos)
- también en documentos

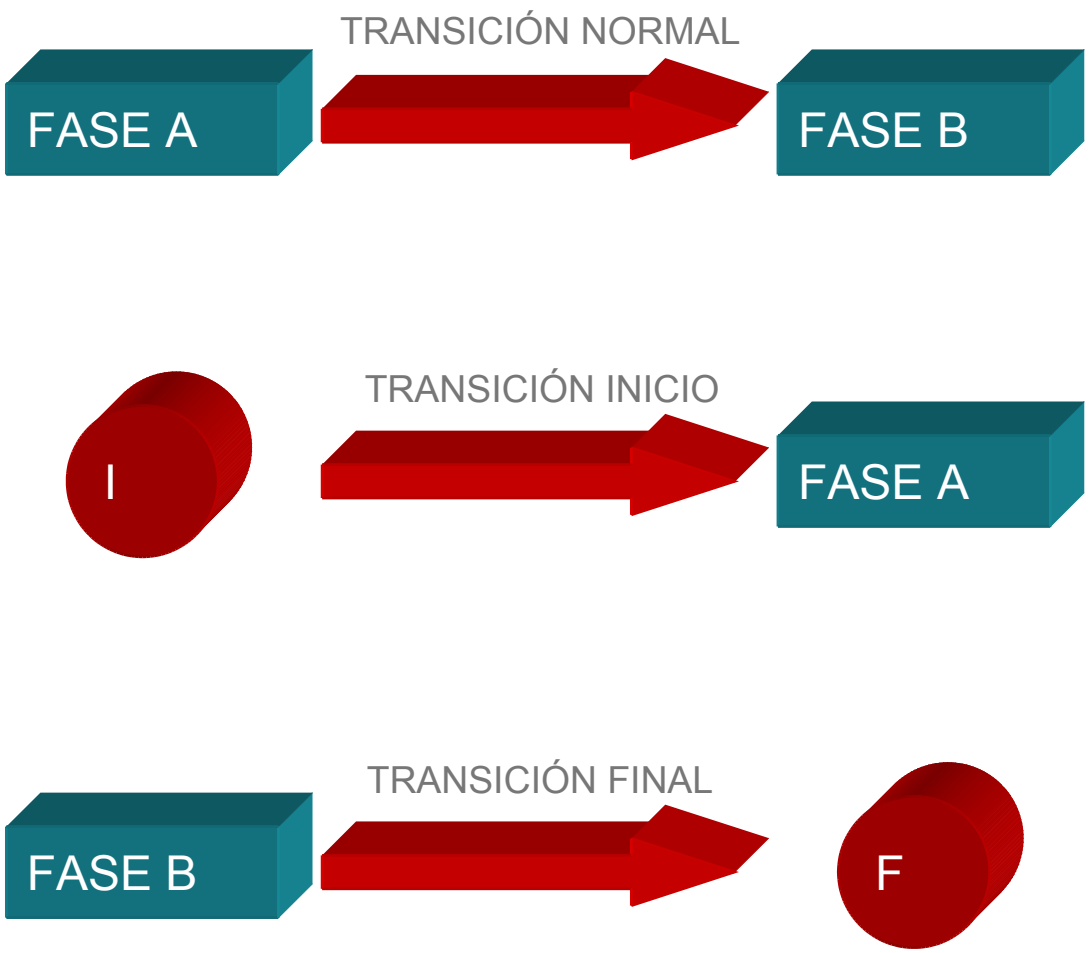
- mensajes que se muestran al tramitador
- también en documentos

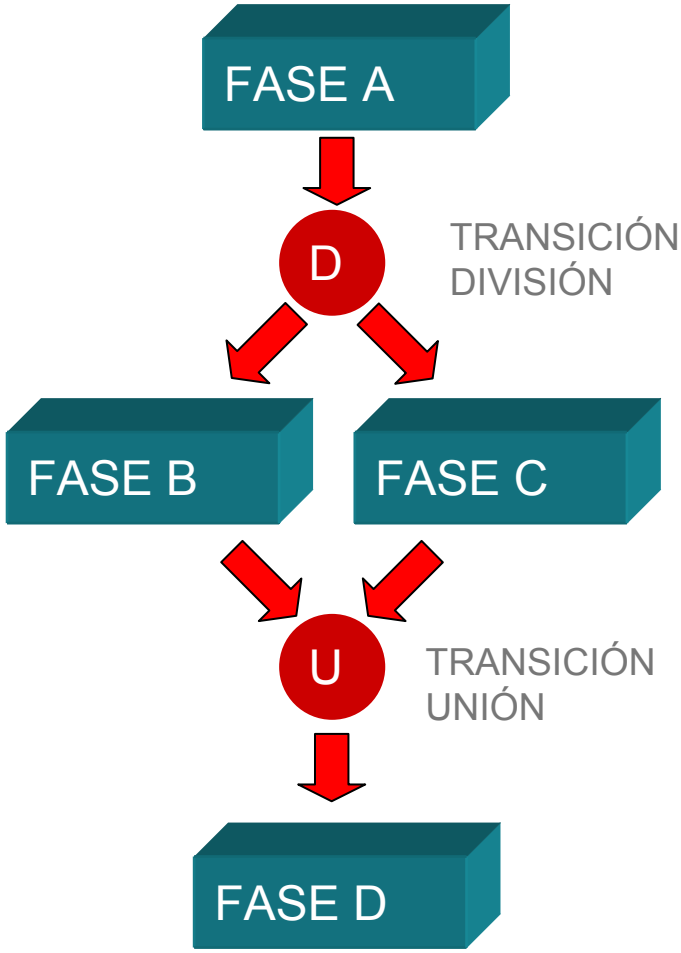
- acciones “disparadas” cuando se produce una transición (una vez verificadas las condiciones)
- también en documentos

En resumen …. Modelo de tramitación w@ndA



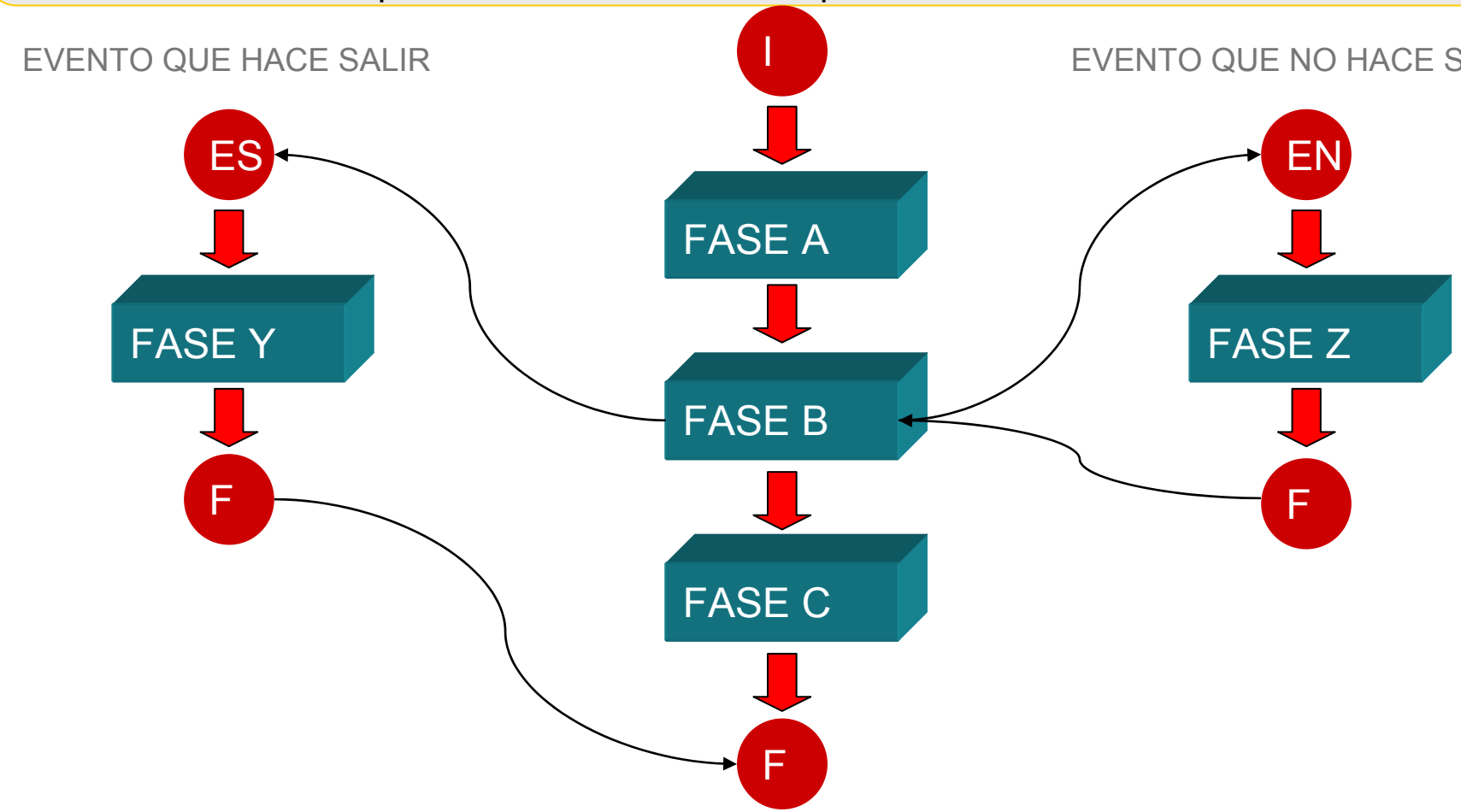
Tipos de transiciones: desde el punto de vista del modelado en w@ndA podemos encontrarnos con los siguientes tipos de transición en un procedimiento:



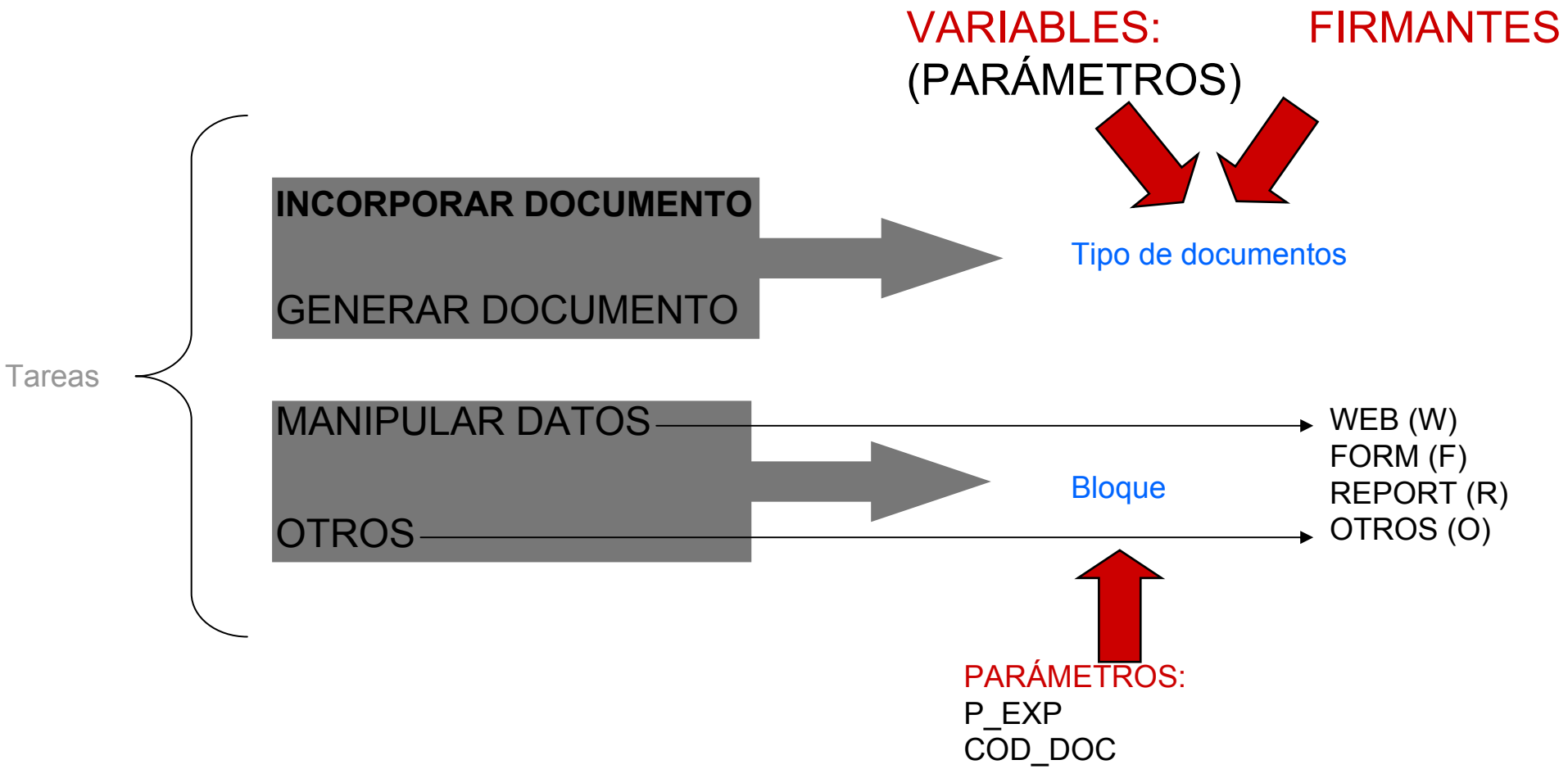


2. Introducción al Modelado de Procedimientos (XI)

Eventos: se dice que en la definición de un procedimiento hay eventos cuando existen circunstancias especiales que provocan que un expediente materializado en el mismo se sitúe en una fase actual preestablecida para esa circunstancia, con independencia de la fase o fases actuales en las que se encontrara antes de producirse dichas circunstancias.



Tipos de tareas



2. Introducción al Modelado de Procedimientos (XIII)

JUNTA DE ANDALUCIA
CONSEJERÍA DE JUSTICIA Y ADMINISTRACIÓN PÚBLICA

Técnicas de modelado: Identificación de metafases, fases, transiciones, tareas por fases, perfiles por transiciones y tareas, acciones, condiciones, avisos, etc.

- Identificar metafases, fases y transiciones.
  - *Metafases: las situaciones generales en las que puede estar un expediente. Es el "Estado del expediente"*.
  - *Fases* detalle de cada metafase definida anteriormente. "El expediente está en..."
  - *Transiciones* entre las distintas fases. Hitos que conducen al expediente de una fase a otra. "Fecha de..."
- Determinar perfiles de usuario para transiciones.
- Identificar tareas en cada fase y perfiles.
  - *Bloques de datos para introducir información.*
  - *Documentos a generar / incorporar.*
- Determinar condiciones de cada transición y de cada tarea (ejecutables como clases JAVA o PL/SQL instaladas en el mismo servidor de aplicaciones).
- Identificar acciones en cada transición y cada tarea.
- Identificar plazos máximos.
- Revisión de elementos.

JUNTA DE ANDALUCIA
CONSEJERÍA DE JUSTICIA Y ADMINISTRACIÓN PÚBLICA

Herramientas de modelado: MODEL@

Principales características de la herramienta gráfica:

- Interface gráfico, 100% Java, que funciona fuera de línea
- Incluye la importación y exportación en XML de definición de procedimientos, subfamilias y familias (XPDL de procedimientos).
- Posee una utilidad para “precargar” definición de fases, documentos, etc (elementos solo dependientes de SISTEMA) existentes en otro fichero XML generado desde Trew@ (XPDL de componentes)
- Permite dar de alta y mantener la información asociada a una definición de procedimientos, así como generar (incluso de forma automática) la representación gráfica asociada.
- Dispone de la funcionalidad necesaria para gestionar las tareas y tareas en fase relacionadas con un determinado procedimiento.
- Deriva un bean para monitorización que puede obtener información acerca de la evolución y estado actual de un expediente que siga un determinado procedimiento.
- Contiene un informe de salida gráfica del procedimiento tanto para impresión directa como a fichero JPG.
- Incorpora la generación de un informe del procedimiento a varios niveles de detalle.

JUNTA DE ANDALUCIA
CONSEJERÍA DE JUSTICIA Y ADMINISTRACIÓN PÚBLICA

Herramientas de modelado: MODEL@

❑ Permite dar de alta y mantener la información asociada a una definición de procedimientos así como generar (incluso de forma automática) la representación gráfica asociada:

- Identificación y definición de metafases.
- Identificación, definición y representación gráfica de fases.
- Identificación, definición y representación gráfica de transiciones tipo N (Normal), D (División), U (Unión), I (Inicio de procedimiento), ES (Evento que hace salir) y EN (Evento que no hace salir).
- Identificación de perfiles de usuarios.
- Establecimiento de perfiles de usuarios en transiciones.
- Definición de tareas en cada fase y asignación de perfiles.
- Definición de acciones y condiciones.

❑ Muchas de estas tareas pueden ser suplidas mediante la importación desde un fichero XML de intercambio previamente construido.

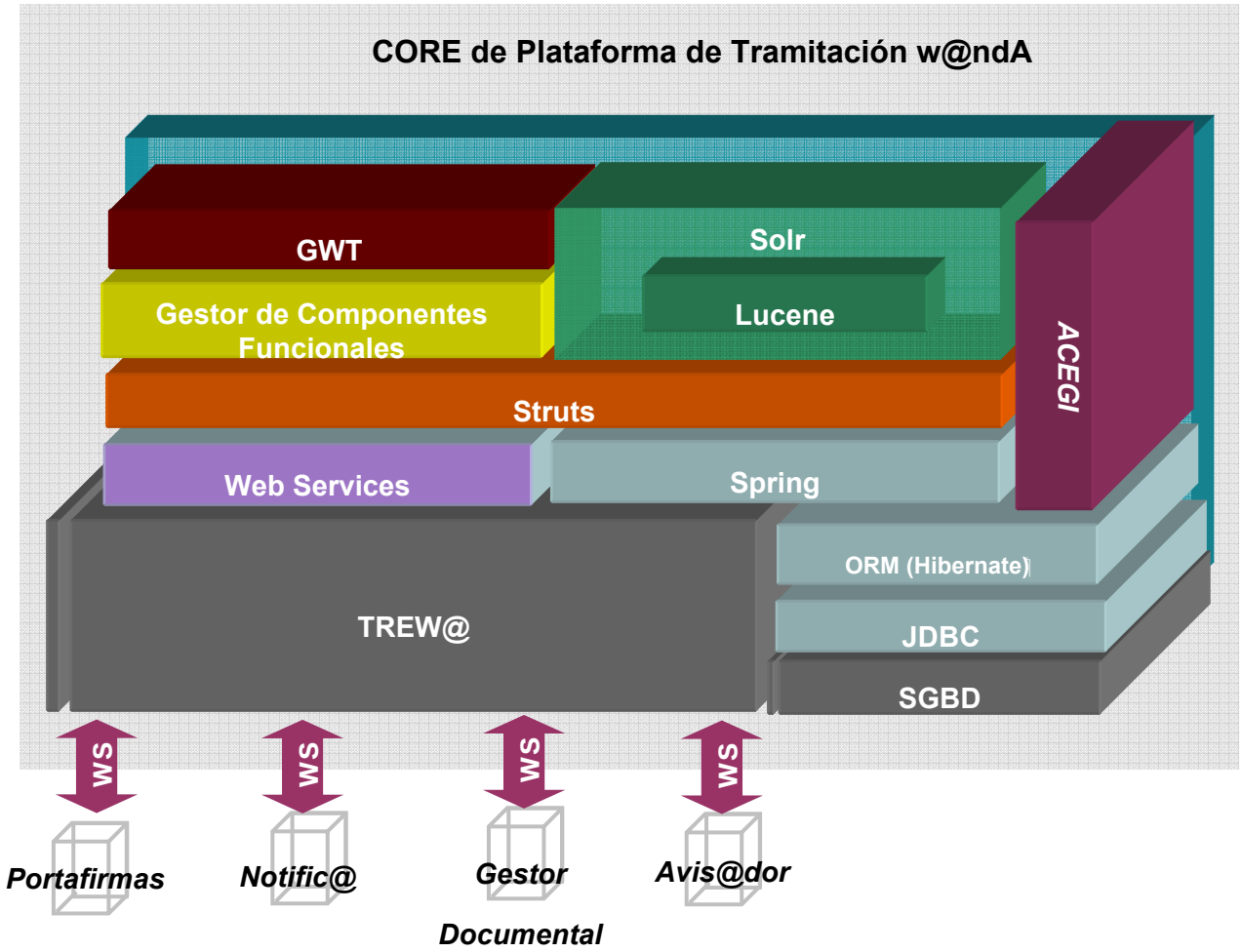
JUNTA DE ANDALUCIA
CONSEJERÍA DE JUSTICIA Y ADMINISTRACIÓN PÚBLICA

Fases de construcción de los procedimientos

- Toma de requisitos.
- Análisis de las necesidades del cliente.
- Primera aproximación a los flujos de los procedimientos (Microsoft Visio o ppt).
- Revisión de los procedimientos con el cliente (proceso iterativo).
- Trascripción de los flujogramas: Identificación de metafases, fases, transiciones, tareas por fase, permisos, condiciones, avisos, etc.
- Construcción de XML de intercambio para el alta de entidades: generación de etiquetas desde .xls de transcripción: metafases, fases, tareas en fase, perfiles.
- Carga de XML de intercambio en Model@.
- Generación del flujo (gráfico) con Model@:
  - Generar el grafico con transiciones.
  - Determinar si todas las tareas tienen todos los perfiles necesarios.
  - Asignar perfiles a transiciones.
  - Hacer nuevas asociaciones de bloques a fases, que no se pudieron hacer en el Excel.
- Ajustes del procedimiento retocando XML de intercambio o mediante la herramienta gráfica (según preferencia del usuario).
- Carga del XML del procedimiento en TREW@ desde la herramienta de Administración.
- Plantillas: desde model@, desde la herramienta de Administración o nuevo procedimiento.
- Estructura orgánica y firmantes desde la herramienta de Administración.

3. Tecnología Utilizada en PT_w@ndA

Como respuesta a toda la problemática asociada a la implantación de una solución de tramitación bajo el marco *w@ndA*, e incorporando las últimas innovaciones en *frameworks*, everis ha construido la siguiente arquitectura para la Plataforma de Tramitación *w@ndA*:

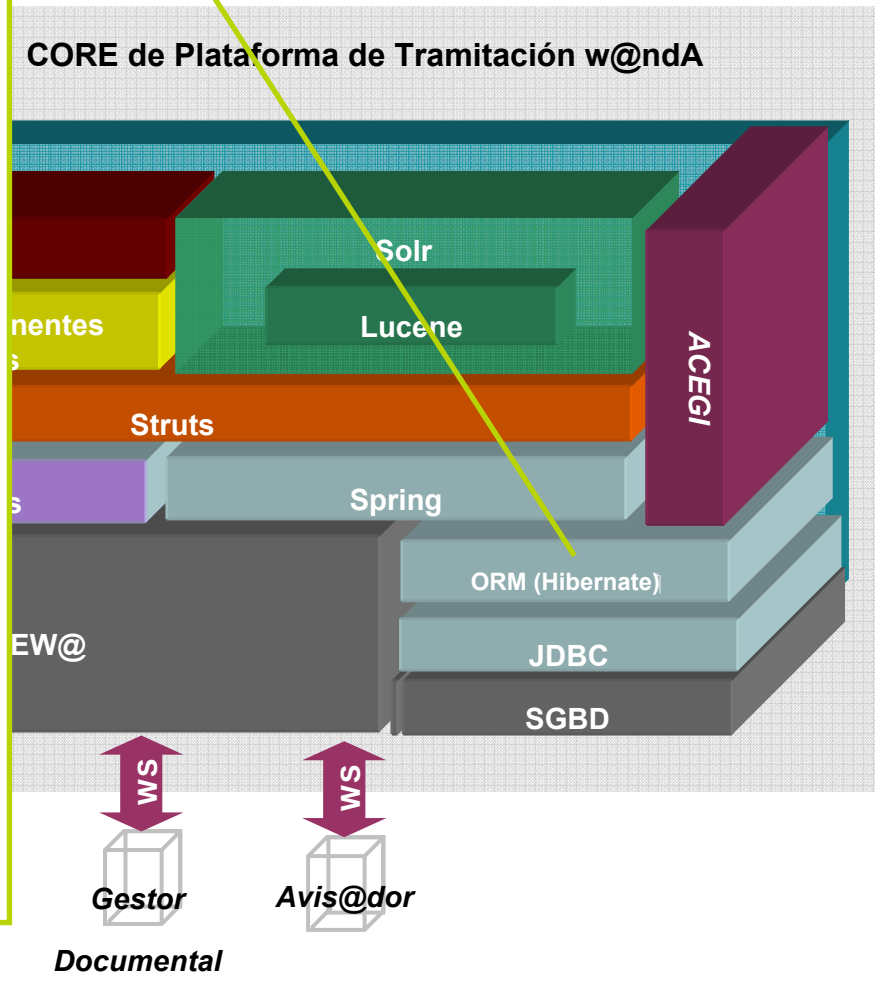


Acceso a Bases de Datos. Persistencia

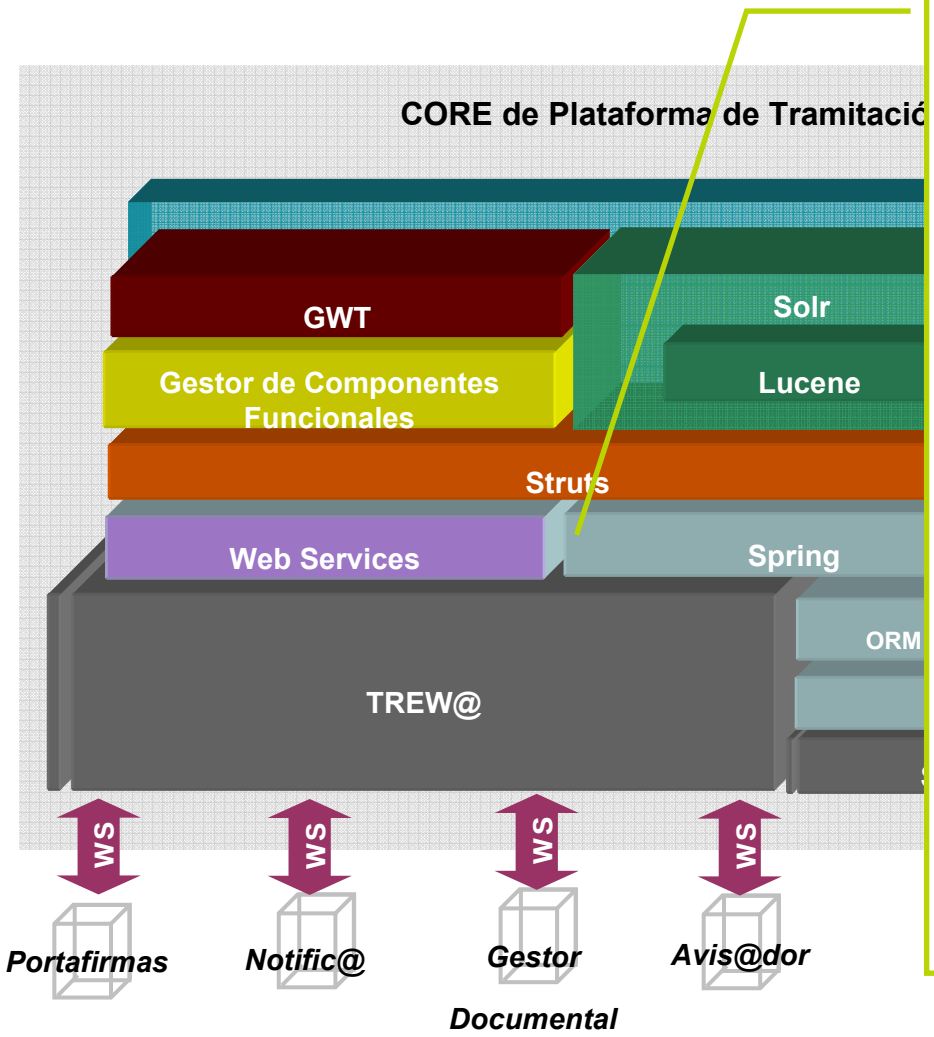
Hibernate, con las siguientes funciones:

- Creación automática de la base de datos al inicializar la aplicación
- Soporte para cualquier base de datos: Oracle, PostgreSql, Mysql, etc.
- Persistencia transparente, permitiendo que cualquier objeto sea persistente.
- Mapeo Objeto-Relacional basado en ficheros XML de configuración, admitiendo cualquier tipo de relación entre entidades.
- Lenguaje de consultas orientado a objetos, HQL, como dialecto de SQL con polimorfismo, de uso facultativo.
- Integración con JMX (operación) y JTA (transaccionalidad).
- Arquitectura de cachés de dos capas.

HIBERNATE

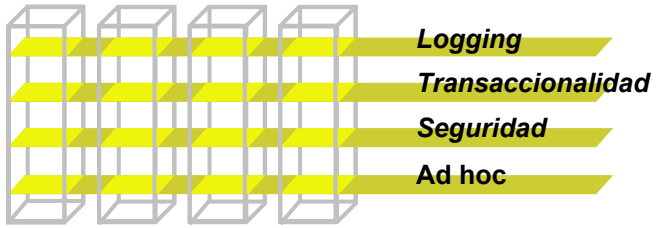


Framework de Servicios y Seguridad 1/3



Framework de Inicialización de Servicios y Seguridad (Spring):
Spring es un contenedor "ligero" con las siguientes características:

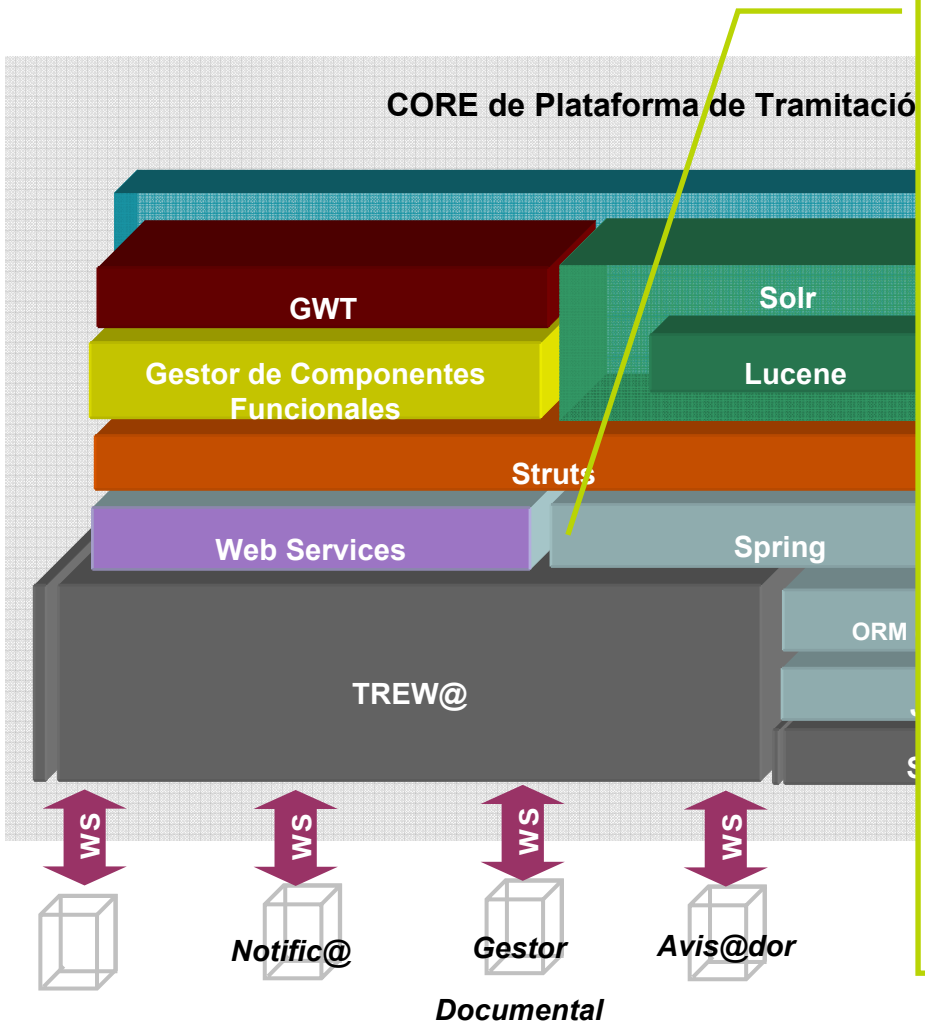
- Hace uso únicamente de objetos "POJO" (*Plain Old java Objects*), sin necesitar EJB's para la lógica de negocio.
- Emplea la inyección de dependencia para establecer las dependencias existentes entre objetos de manera declarativa.
- Incluye *AOP* (*Aspect Oriented Programming*) para poder definir declarativamente "asuntos cruzados" (*cross-cutting concerns*), sobre los que la propia lógica de negocio debería permanecer independiente (seguridad, transaccionalidad, distribución).



Componentes

Spring

Framework de Servicios y Seguridad 2/3



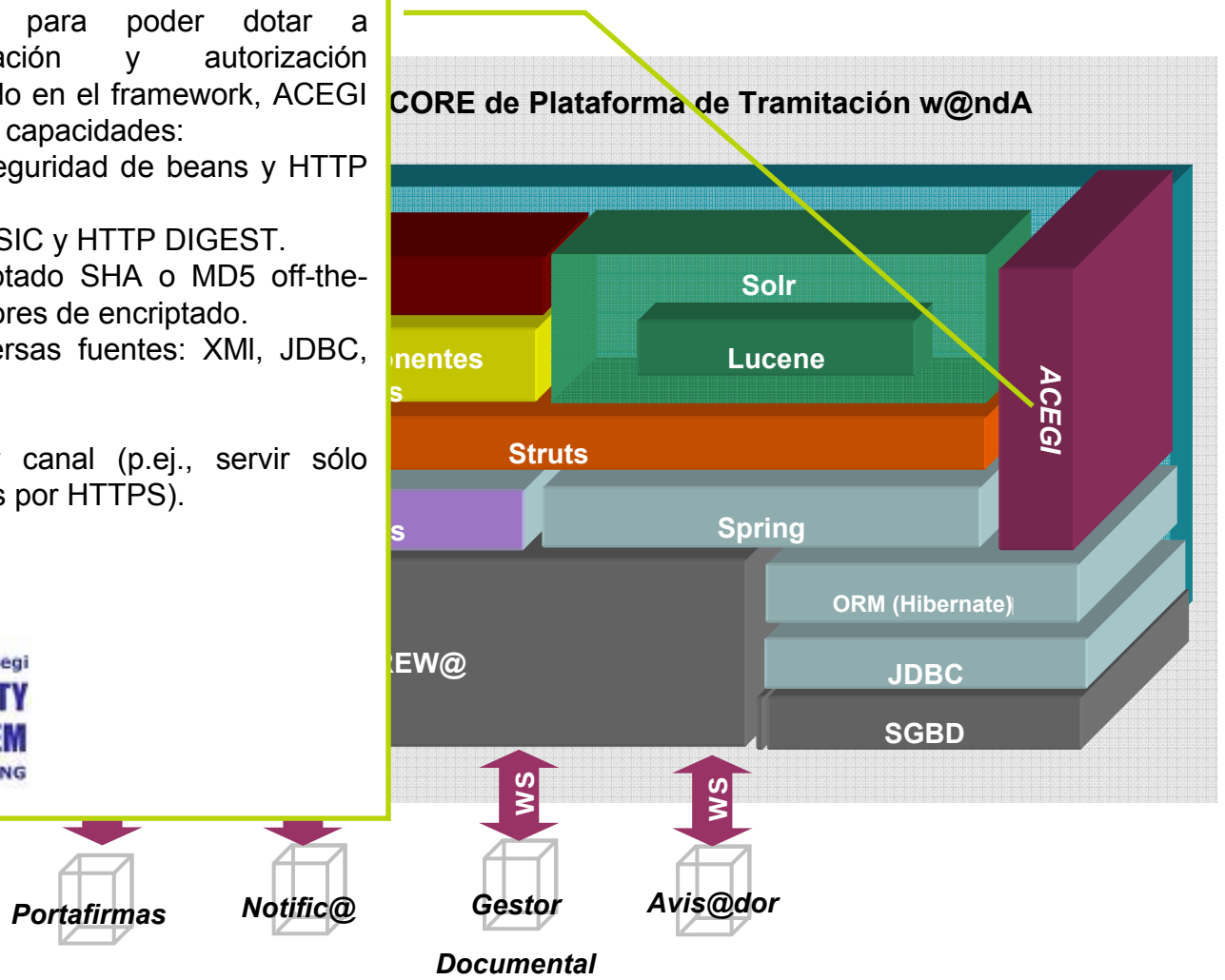
- *Plataforma* utiliza Spring como contenedor en cada una de las tres capas de la aplicación: *presentación, negocio* y *modelo*.
- En el caso de la presentación, recurre a un plug-in que integra directamente a este contenedor con *Struts-2*
- En el caso de negocio, se ha desarrollado una capa de servicios inyectada totalmente a través de *Spring*
- En el caso del modelo, se ha recurrido a un plug-in que inyecta los DAO's del ORM (Hibernate) a través de Spring, definiéndose en sus ficheros de configuración las transacciones y demás requerimientos relacionales.
- *Spring* ha posibilitado también la rápida y flexible integración de un *scheduler* de programación de tareas: *Quartz*, usado sobre todo para el mantenimiento de sincronismo entre el núcleo de la aplicación y el sistema gestor de la tramitación.
- En cuanto al *AOP* que proporciona el framework, se ha empleado para la realización de auditorías. Se definen *beans de servicios registrables* de tal manera que se audite las excepciones y se traceen las entradas y salidas a cada uno de los métodos.

Spring

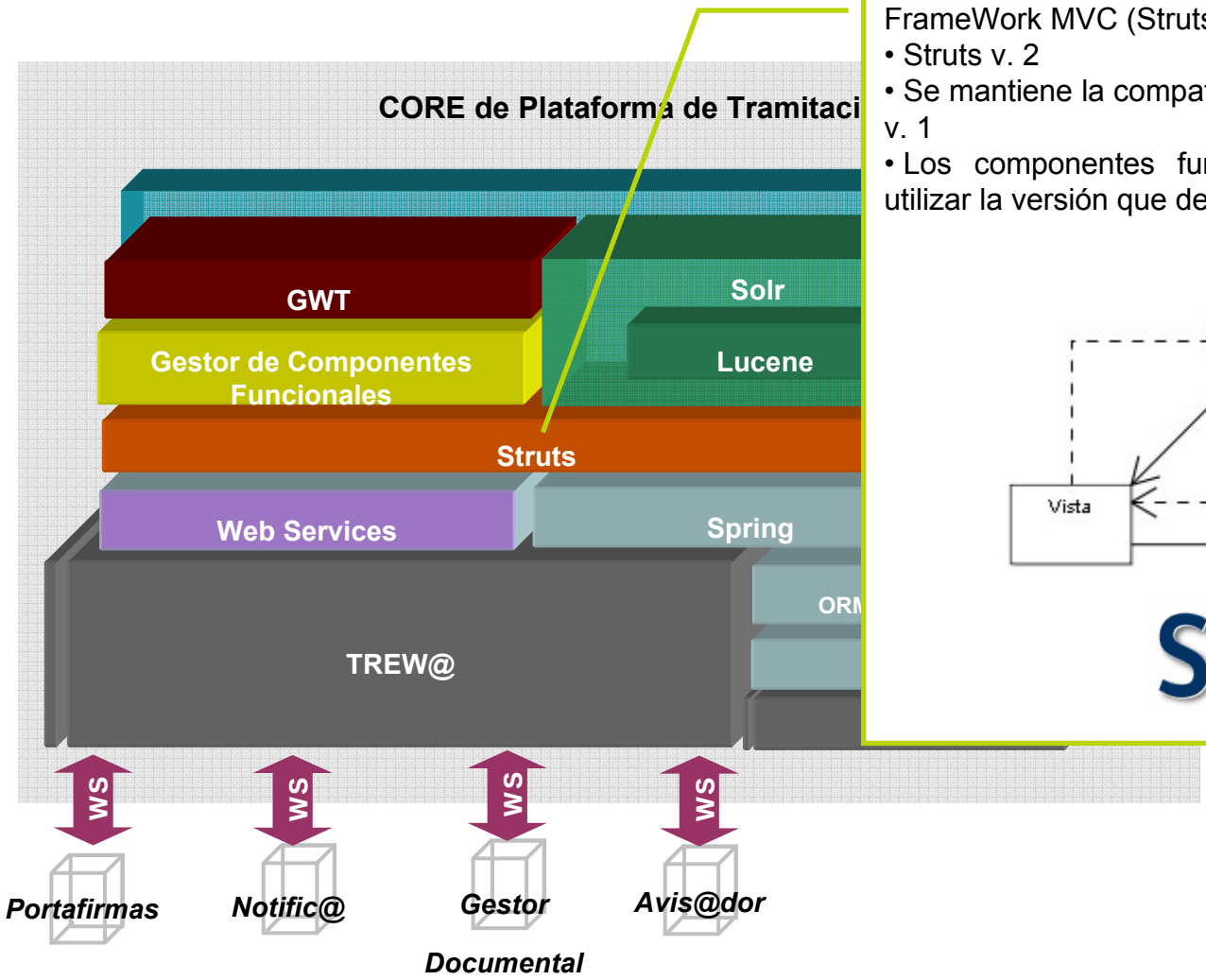
Framework de Servicios y Seguridad 3/3

ACEGI se integra con Spring para poder dotar a la arquitectura de autenticación y autorización de manera declarativa (AOP). Integrado en el framework, ACEGI proporciona, entre otras, las siguientes capacidades:

- Definición por parametrización de seguridad de beans y HTTP requests.
- Soporte para autenticación HTTP BASIC y HTTP DIGEST.
- Gestión avanzada de claves: encriptado SHA o MD5 off-the-shelf, o integración directa de proveedores de encriptado.
- Información de autorización en diversas fuentes: XMl, JDBC, fichero Properties, LDAP.
- Soporte para certificados X.509.
- Posibilidad de definir políticas por canal (p.ej., servir sólo recursos públicos por HTTP, y privados por HTTPS).

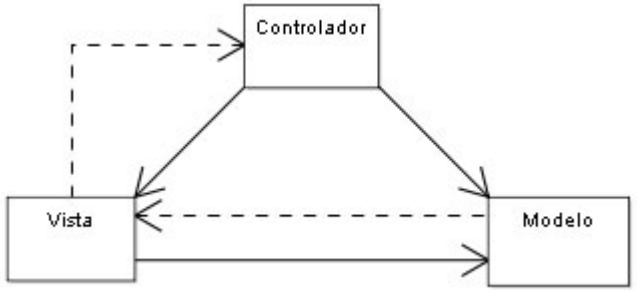


Framework MVC



FrameWork MVC (Struts):

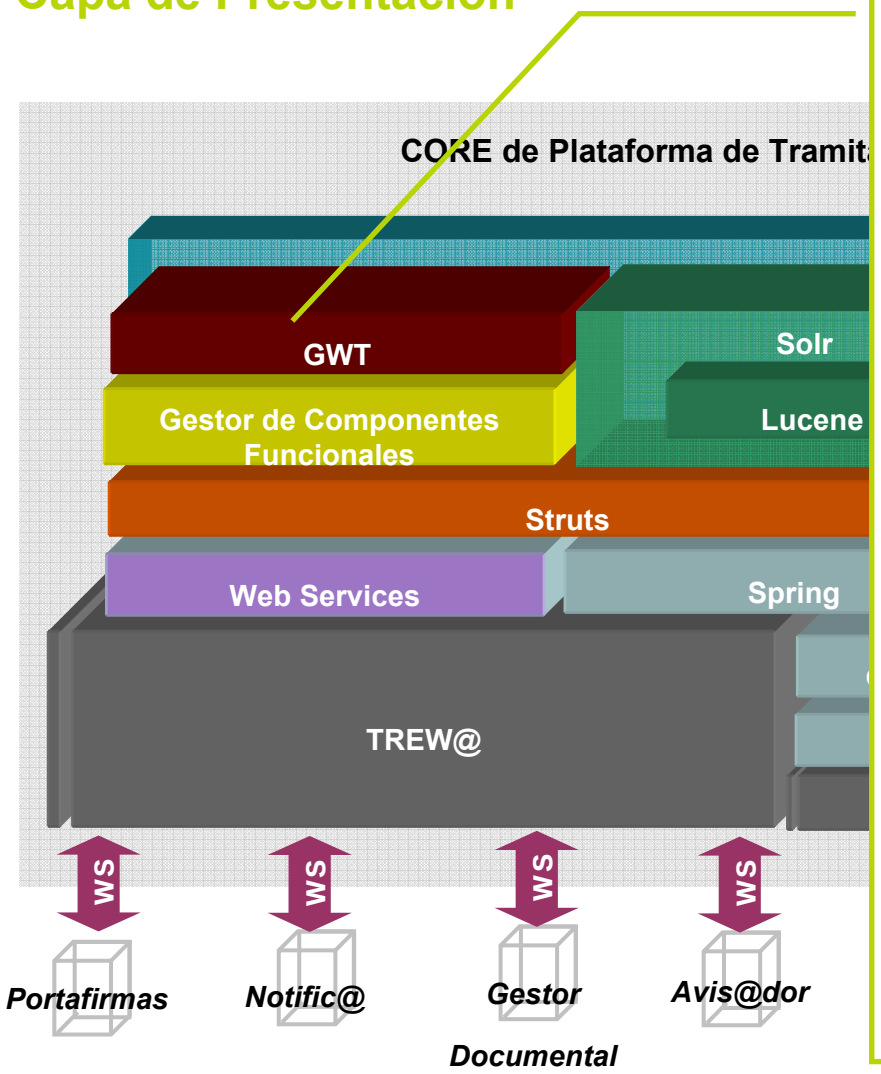
- Struts v. 2
- Se mantiene la compatibilidad con la versión anterior de Struts v. 1
- Los componentes funcionales que se desarrollen podrán utilizar la versión que deseen de Struts



Struts

3. Tecnología Utilizada en PT_w@ndA (VII)

Capa de Presentación



FrameWork de Presentación desarrollado haciendo uso de GWT (Google Web Toolkit):

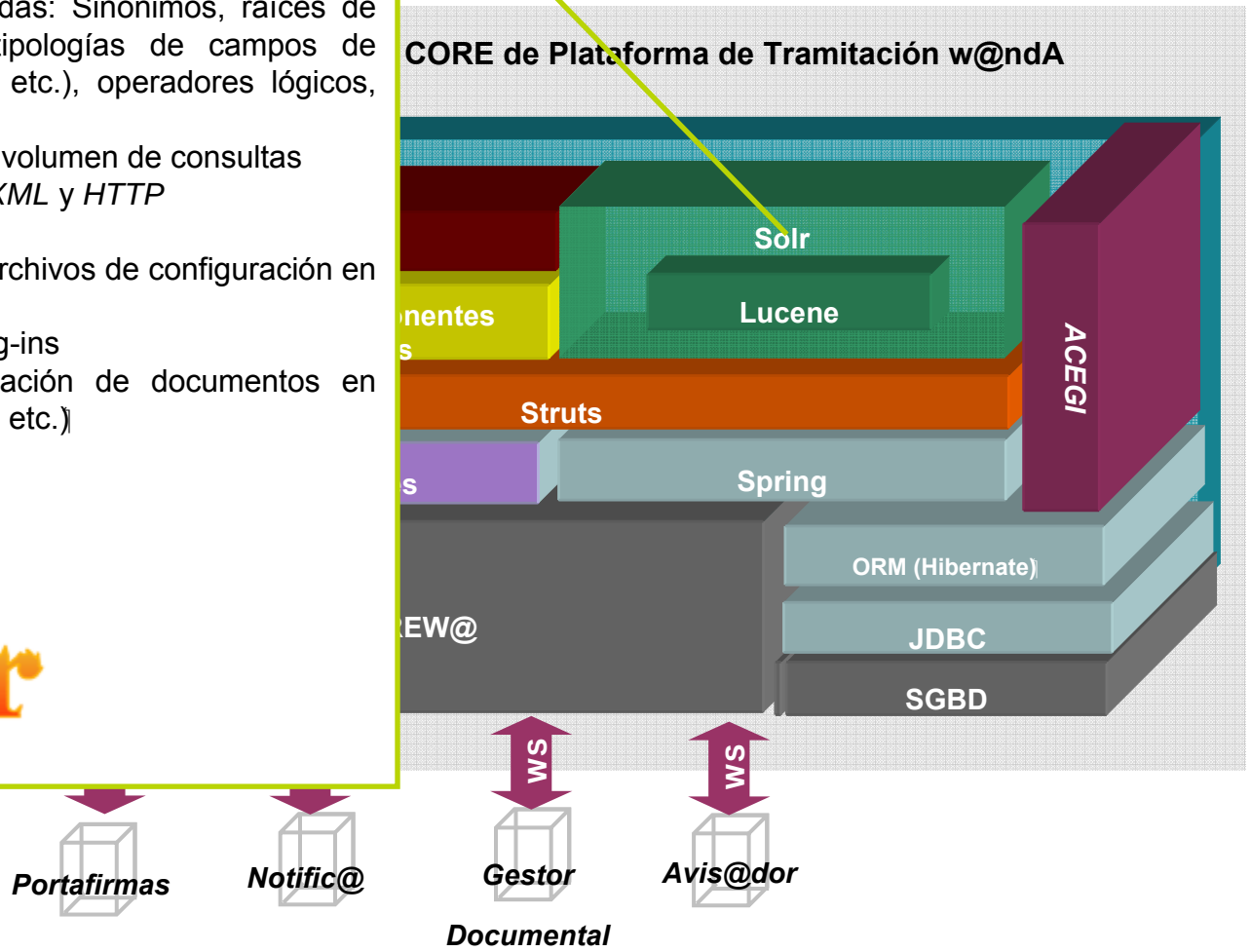
- La capa de presentación se confecciona bajo la configuración de una malla que delimita las zonas y estructuración de la página.
- Los desarrolladores de nuevos componentes funcionales se despreocupan totalmente de la presentación de sus módulos o portlets (posicionamiento, dependencias, control de visibilidad, etc.)
- La personalización de la interfaz del Escritorio en cada implantación es inmediata

Google Code

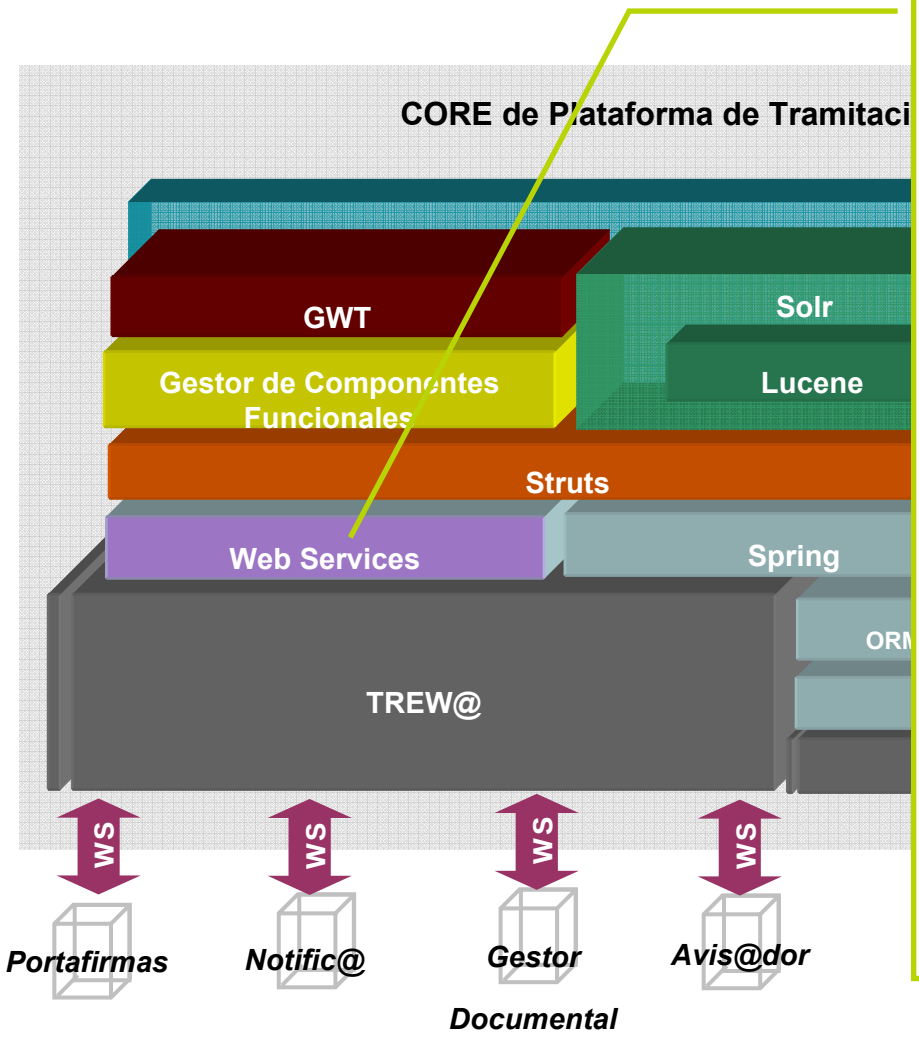
Motor de Indexación y Búsqueda

Solr del proyecto *Apache Lucene*:

- Capacidades de búsquedas avanzadas: Sinónimos, raíces de palabras, aproximación, diferentes tipologías de campos de búsqueda (fecha, real, entero, texto, etc.), operadores lógicos, etc.
- Optimizado para soportar un elevado volumen de consultas
- 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
- Integrado con *LIUS* para la indexación de documentos en formato binario (MS Word, Excel, PDF, etc.)



Capa de Servicios Web



Plataforma de Tramitación ofrece una capa de servicios web, cuyos métodos serán descritos más adelante, y cuya pretensión es llevar las funciones más básicas de la aplicación fuera de su propio ámbito: accesibles desde cualquier otro aplicativo que lo requiera.

Esta capa de servicios se implementa haciendo uso de Spring (*SpringWS*) y apoyándose en el motor de tramitación (*Trew@)* como contenedor de información.

- *SpringWS* proporciona una forma fácil de comunicar la información, a través del protocolo libre XML. Permite definir cualquier tipo de petición, con independencia de la naturaleza del mensaje.
- *SpringWS* soporta además numerosas APIs de tratamiento XML, permitiendo encapsular los mensajes XML en cómodos Java Beans.
- *SpringWS* hace uso, además, de WS-Security, permitiendo encriptar y desencriptar los mensajes SOAP, además de proceder a una verdadera autenticación a través de ellos.
- *SpringWS* se integra con el framework Acegi, de tal forma que la misma configuración de seguridad de la aplicación puede llevarse a la capa de servicios.
- *SpringWS* posee licencia Apache

Componentes Funcionales

Los Componentes Funcionales son el instrumento natural para incluir nuevas funcionalidades en la Plataforma de Tramitación w@ndA.

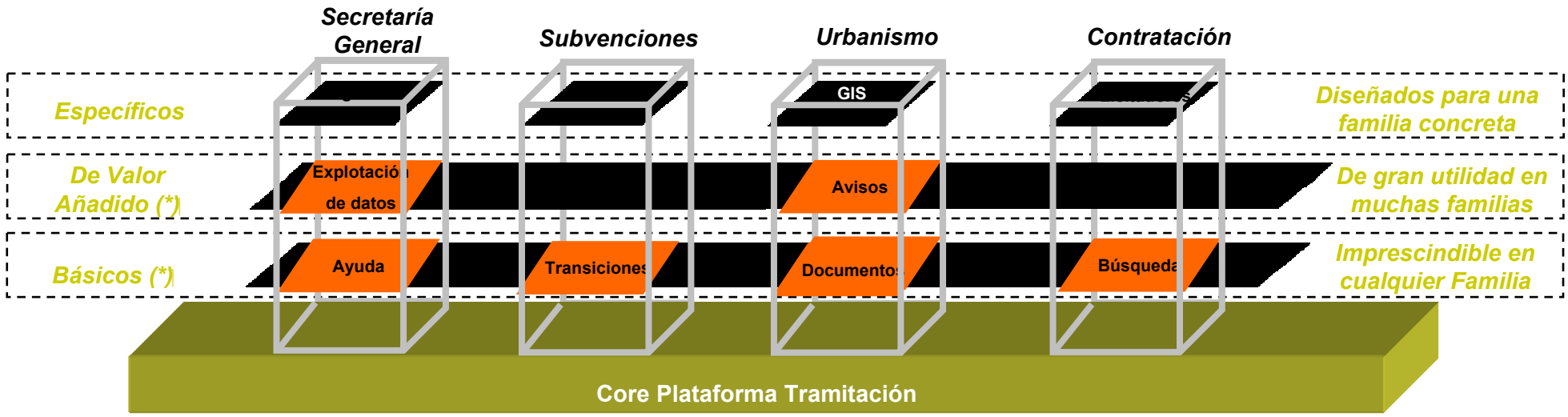
Los Componentes Funcionales de la Plataforma de Tramitación w@ndA ofrecen funcionalidades adicionales que no están disponibles en el core del sistema. El Gestor de Componentes Funcionales proporciona mecanismos sencillos y perfectamente definidos para la construcción de estos plug-ins funcionales para Plataforma.

Un componente de este tipo podrá contener cualquier recurso propio de una aplicación Web: Clases Java, librerías, imágenes, hojas de estilo Css, librerías Javascript, archivos de configuración, etc.

Estos componentes software podrán ser importados e instalados en la Plataforma de Tramitación a través de la herramienta de administración empaquetados en forma de archivos ZIP.

Componentes Funcionales

A continuación se describen los diferentes tipos de Componentes Funcionales que se pueden encontrar en la Plataforma:



(*) (LOS COMPONENTES BÁSICOS Y DE VALOR AÑADIDO ESTÁN INCLUIDOS EN LA DISTRIBUCIÓN DE PLATAFORMA)

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:

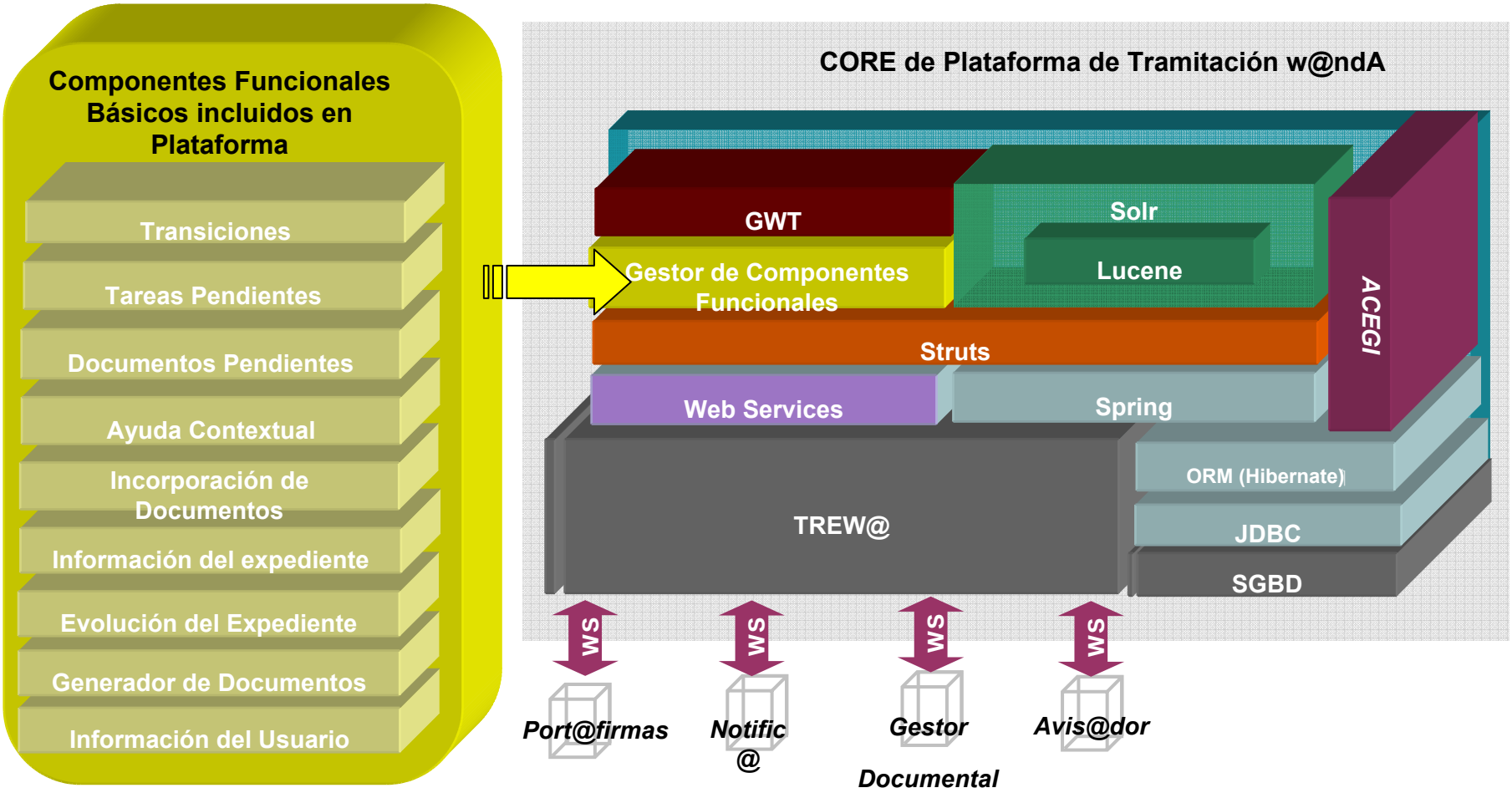
- Un módulo podrá incorporar los siguientes recursos bajo un archivo ZIP: librerías (jars, páginas JSP, imágenes, CSS, etc.)
- Los módulos podrán implementar las reglas de navegación bajo cualquier versión de Struts
- El módulo irá acompañado de un descriptor (archivo XML)
- La plataforma validará de 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..

Componentes Funcionales

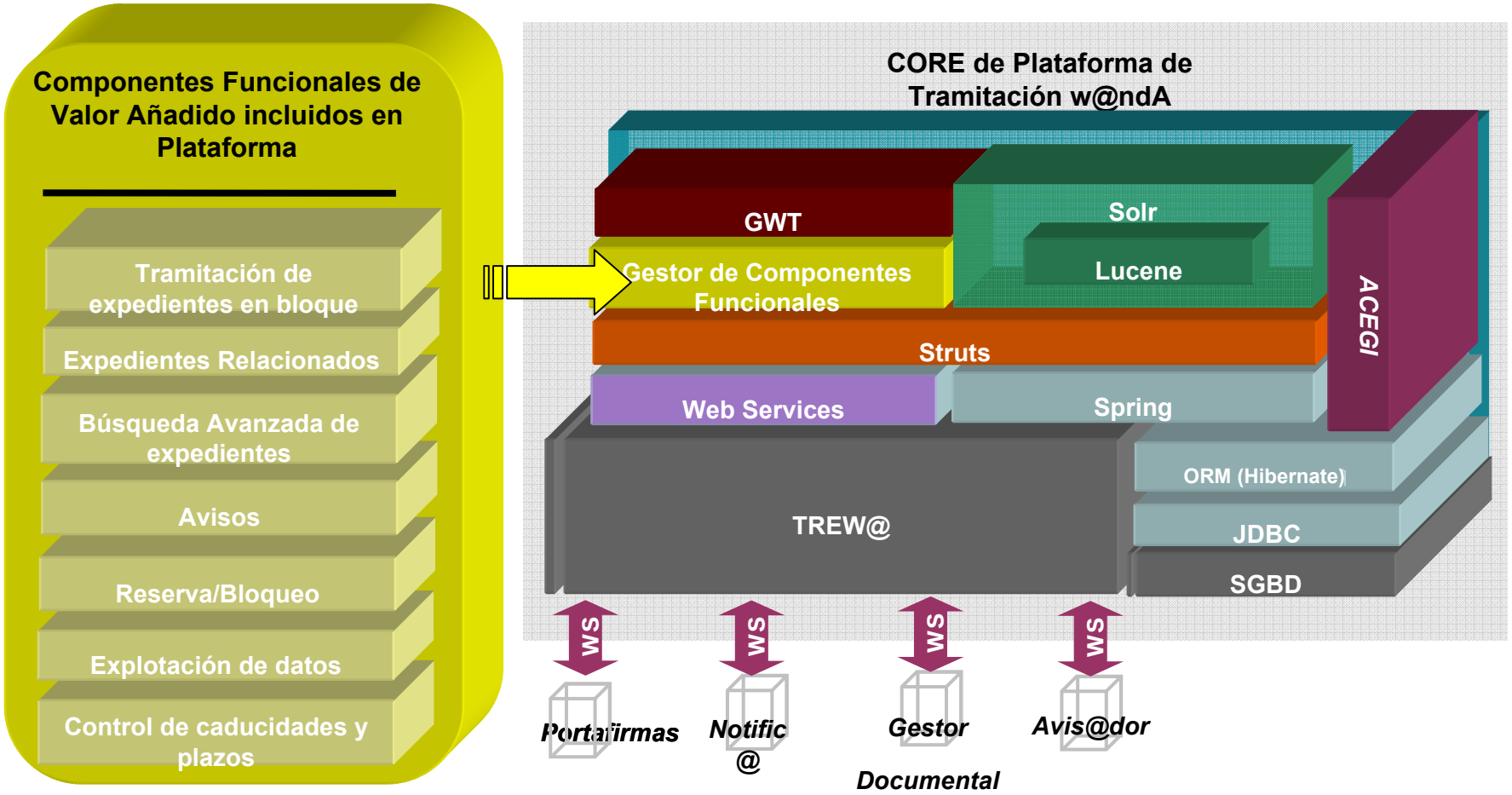
Un componente funcional o módulo de la Plataforma de Tramitación estará tipificada dentro de las siguientes categorías:

- Módulos de tipo ***PORTLET***
  - Son aquellos que se visualizan en el escritorio de tramitación en forma de ventanas minimizables y maximizables adosadas.
- Módulos de tipo ***UTILIDAD***
  - Son aquellos que son representados, dentro del escritorio de tramitación, como iconos '*pinchables*' que liberan una ventana emergente contenedora de la funcionalidad asociada.
- Módulos de tipo ***EXTERNO***
  - Son aquellos que pueden ser instanciados desde cualquier punto de la aplicación (o incluso desde aplicaciones externas), haciendo uso de la URL asociada al componente.
- Módulos de tipo ***MENU***
  - Se trata de una tipología especial de componente, ya que pertenecen a los módulos *externos*. Tienen la pecularidad que pueden formar parte de un *menú* de opciones.

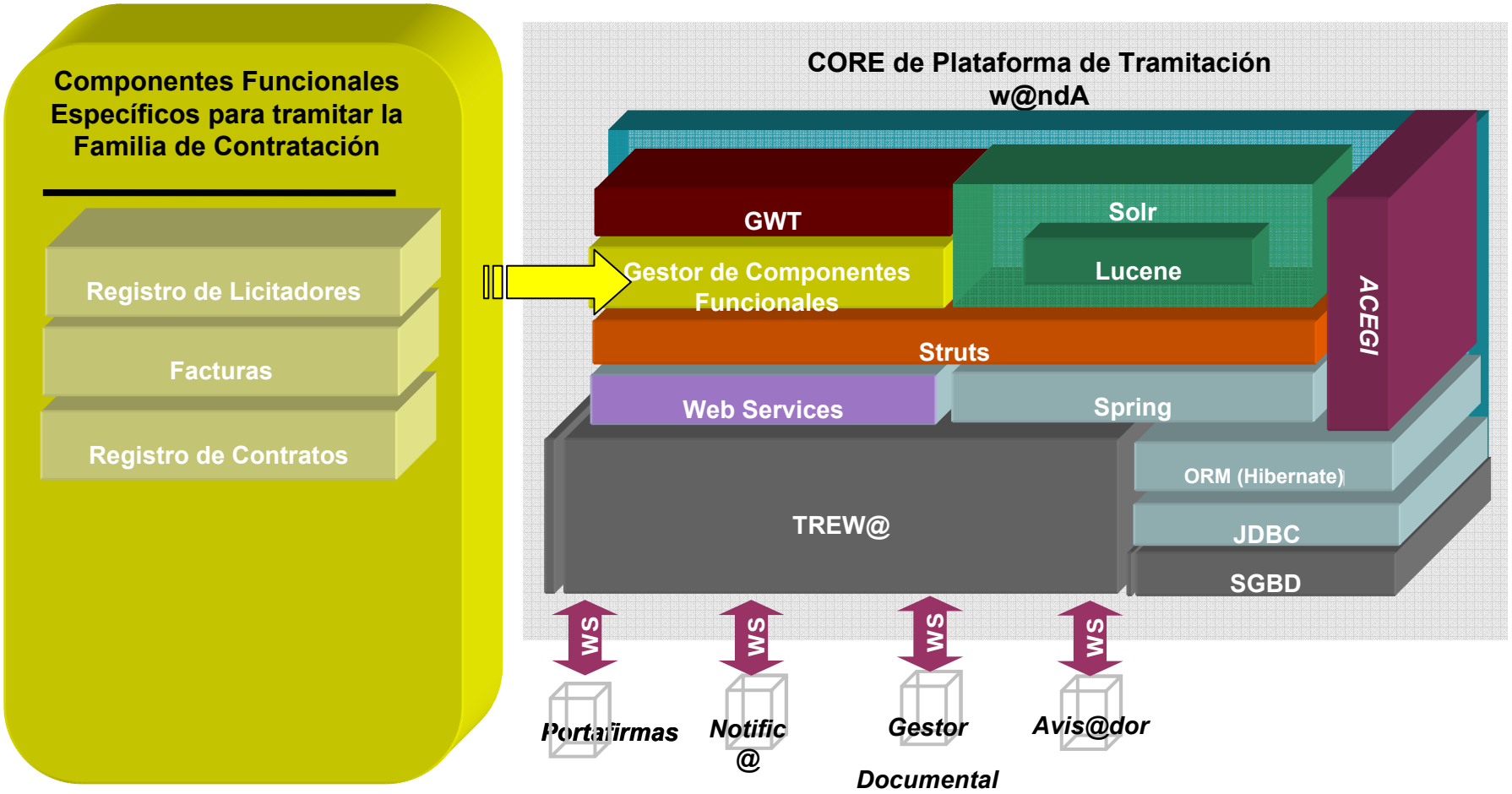
Componentes Funcionales Básicos



Componentes Funcionales de Valor Añadido



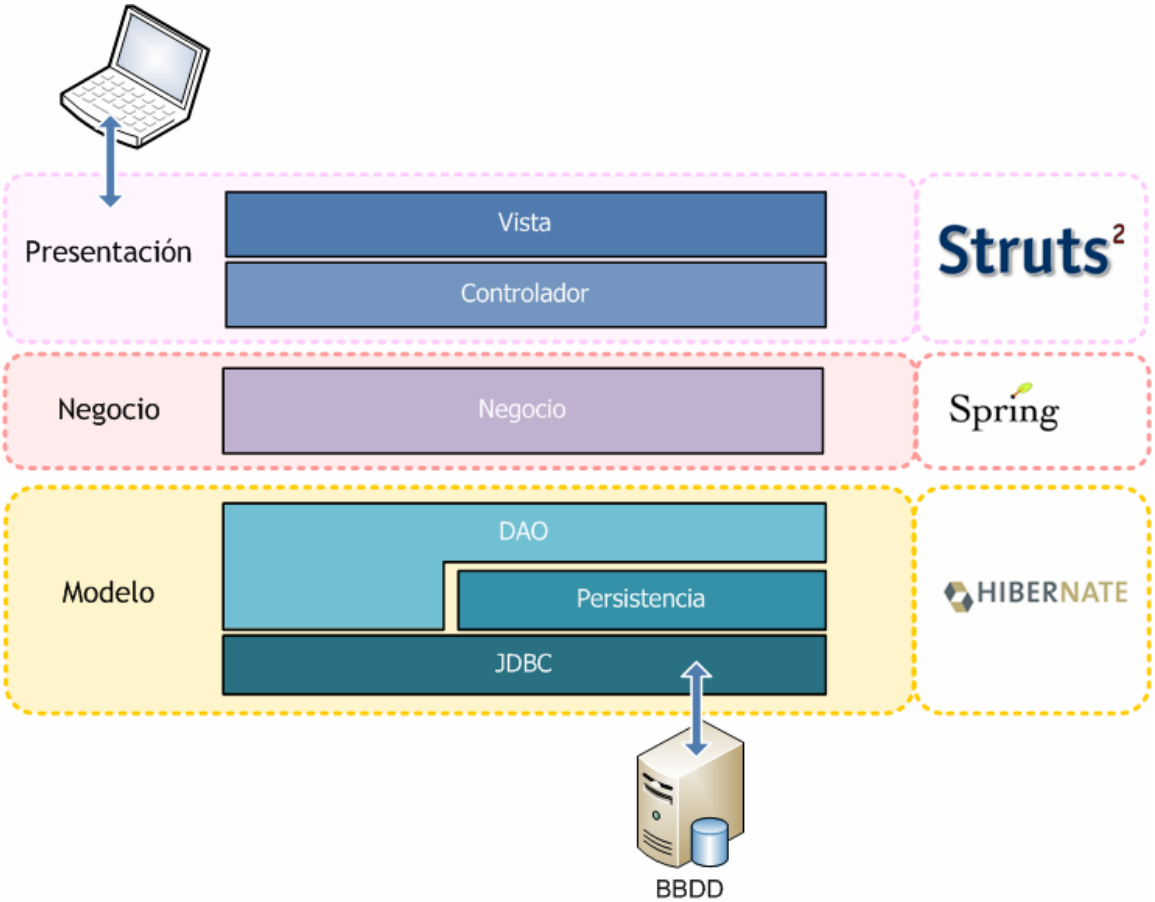
Componentes Funcionales Específicos



La **capa de Negocio** de Plataforma implementa la lógica de Negocio del Sistema, separando la lógica de datos de la lógica de negocio, cumpliendo así con el patrón **MVC (Modelo – Vista – Controlador)**

Modelo Vista Controlador (MVC) es un patrón de diseño de **arquitectura de software** que tiene como **objetivo** separar los **datos** de una aplicación, la **interfaz de usuario**, y la **lógica de control** en tres **componentes** o **capas** independientes.

Los **servicios** son clases Java ubicadas en la capa de Negocio, y que implementan funcionalidad para gestionar la información e interactuar con el motor de tramitación Trew@



4. Servicios (II)

Para hacer que los servicios sean flexibles y puedan ser modificados de forma sencilla, inicialmente se creará una interfaz Java donde se definirán los métodos necesarios para interactuar con la lógica de datos de la forma que convenga y si fuese necesario, realizar tantas clases que implementen dicha interfaz permitiendo la comunicación con la lógica de datos de múltiples formas.

1.El primer paso es crear una interfaz Java en la que se definan los métodos que va a ofrecer el servicio como se muestra en el siguiente ejemplo:

IAltaExpedientesService.java

```
package es.juntadeandalucia.plataforma.interfaces.alta;

public interface IAltaExpedientesService extends IConfigurableService {

   String generacionNumeroAlta(String tipoProcedimiento, UsuarioWeb
user);
   void setOtrosDatos(String otrosDatos);
   String getOtrosDatos();
   boolean tienePermisos(String idUser, String Pataformapermiso);
   boolean borrarExpediente(TpoPK idExpediente) throws TrException;
}
```

4. Servicios (III)

2. Una vez definida la interfaz, es necesario crear una clase Java que implemente dicha interfaz. Por ejemplo, esta interfaz se implementará en la clase *AltaExpedientesServiceImpl.java*.

3. Una vez creado el servicio, es necesario definir un fichero llamado applicationContext-xxxx.xml que contenga los servicios creados.
   Por ejemplo, el fichero creado se denominará *applicationContext-servicios.xml* en el cual se definirán todos los servicios creados específicamente que no sean de la Plataforma de Tramitación.

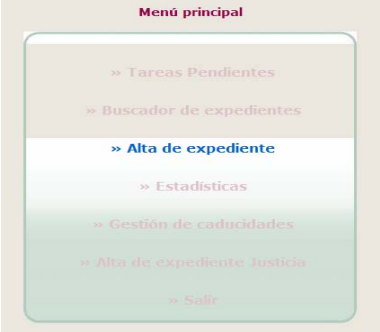
applicationContext-servicios.xml

```
<beans>
  <!-- Servicios disponibles en la plataforma de tramitación -->
   <bean id="altaExpedienteService" class="es.juntadeandalucia.plataforma.services.alta.AltaExpedientesServiceImpl"
    singleton="false">
      <property name="tramitacionService">     <ref bean="tramitacionService"/>     </property>
      <property name="cacheApiService">   <ref bean="cacheApiTramitadorService"/>   </property>
   </bean>
   <bean id="tareasService" class="es.juntadeandalucia.plataforma.services.tareas.TareasServiceImpl" singleton="false">
     <property name="cacheApiService">   <ref bean="cacheApiTramitadorService"/>    </property>
    </bean>
</beans>
```

En este fichero se define el identificador del servicio, mediante la etiqueta "**bean id**", así como la ruta en la que se encuentra la clase que implementa dicho servicio, mediante la etiqueta "**class**". Por último, cuando se precise utilizar servicios ya creados, se incluirá en la definición de las clases que vayan a usarlos el identificador del servicio, es decir, debe aparecer enla etiqueta "**ref bean**" dicho identificador. En el caso mostrado, el servicio altaExpedienteService utiliza los servicios *tramitacionService* y *cacheApiTramitadorService.*

5. Alta Expediente

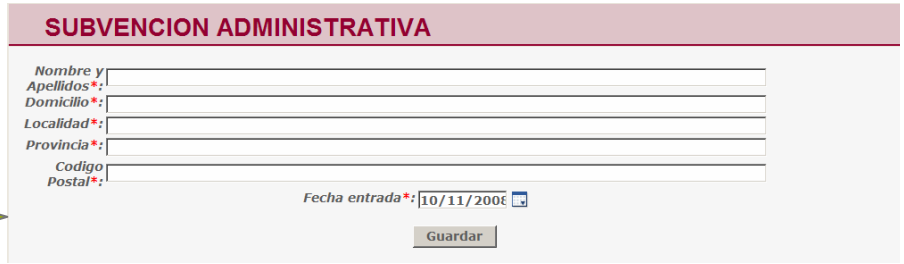
En el caso que la solicitud de expediente se realice de forma presencial o de oficio, será necesario desde Plataforma de Tramitación dar de alta el expediente.



El proceso de Alta de expediente se divide en dos partes fundamentales:

- ***Ventana de alta genérica:*** *Una ventana de alta genérica donde se solicitan los datos correspondientes a la figura*

- ***Ventana de alta específica:*** *En función del tipo de procedimiento seleccionado en la ventana de Alta Genérica, se redirigirá la aplicación a una ventana de Alta específica que contendrá los datos necesarios del tipo de procedimiento seleccionado*



5. Alta Expediente (II)

A continuación se especifican los ficheros a generar y/o modificar para realizar el alta de un expediente:

Fichero de propiedades "xxx.properties": *es necesario mantener un fichero de propiedades, que incluya el identificador del procedimiento en Trew@. Con este identificador, se podrá diferenciar el tipo de procedimiento para redirigir la aplicación a la ventana de alta específica.*

just.properties

reu_subadm=217

JSP de alta "altaXXX.jsp"*: formulario de adquisición de datos específico para cada procedimiento. En este formulario se debe definir mediante las etiquetas de Struts, el nombre del action que se ejecutará cuando se realice el "submit" del formulario.*

altaSUBADM.jsp

```
<s:form enctype="multipart/form-data"  theme="simple" name="moduloAltaExpEspecifica">

……
……
<s:submit value="Guardar" action="guardaAltaSUBADM"/>
```

5. Alta Expediente (III)

***Xml de struts "strutsXXX.xml"**: este fichero se incluyen las reglas de navegación entre la ventana del alta genérica de expediente y la ventana del alta específica de cada procedimiento indicado en el formulario de adquisición de datos específico ".jsp". Se define la relación entre el nombre del action, el nombre de la clase java y el método en el cual se van a tratar los datos.*

struts-alta-expediente.xml

```
<action name="guardaAltaSUBADM" method="guardaAltaSUBADM"
                                                class="AltaSUBADM">
        <result name="localizacion" type="redirect">
            /busqueda/irABuscadorGenerico.action?refExpedienteCreado=${numExpCreado}
            &amp;refNumExpediente=${numero}
        </result>
        <result name="input">altaSUBADM.jsp</result>
        <result name="error">/inicio/error.jsp</result>
        <interceptor-ref name="defaultStack" />
</action>
```

En el código anterior se indica, que cuando se invoque la acción "**action name**", se debe ejecutar el método que se le ha indicado en la etiqueta "**method**" que pertenece a la clase definida en la etiqueta "**class**".

***"struts.xml"**: en este fichero se deben incluir los struts previos, es decir, aquellos "strutsXXX.xml" que se han ido creando. Así, cuando se inicie la aplicación, mediante este fichero, se irán mapeando todas las acciones incluidas en los diversos strutsXXX.xml.*

struts.xml

```
<include file="struts-alta-expediente.xml"/>
```

5. Alta Expediente (IV)

Clase action "XXX.java": *En esta clase es necesario definir un método con el mismo nombre asignado a la etiqueta "**method**" en la regla del action del fichero "**strutsXXX.xml**". Ese método tendrá que realizar todo lo necesario para crear el expediente tanto en Trew@ como en el cliente Solr. El cliente Solr es un servicio de indexación y búsqueda, siendo necesario enviarle la información tal y como se encuentre definida en el fichero "schema.xml". Para crear el expediente, esta clase usará los servicios necesarios para interactuar con Trew@.*

AltaSUBADM.java

```
public class AltaSUBADM extends ConfigurableAction implements SessionAware{

public String guardaAltaSUBADM() throws BusinessException, ArchitectureException {
   ……..
   if (!resultado.equals("success"))
      return "error";
   else{
      configurarServicio(user, consultaExpedienteService);
       nuevoExpediente = consultaExpedienteService.obtenerExpedientes(altaExpedientesService.getIdExpedienteCreado().toString(),
null, null);
       if (nuevoExpediente == null)
          return "error";
      else{
          numero= nuevoExpediente.get(0).getNumeroExpediente();
          setRefExpedienteCreado(nuevoExpediente.get(0).getRefExpediente());
          setNumExpCreado(nuevoExpediente.get(0).getRefExpediente());
          asociarInteresado();
          return resultado;
       }
  }
……..
```

5. Alta Expediente (V)

***"applicationContext.xml"**: en este fichero se definen los identificadores de las clases Java (acciones), así como todos los servicios empleados por dichas clases. En este fichero se definen las clases Java que gestionarán las altas de los diferentes tipos de procedimientos y las acciones necesarias en las tareas, así como la definición de las clases Java que implementan los servicios desarrollados por y para el proyecto vertical.*

applicationContext.java

```
<bean id="AltaSUBADM" class="com.everis.altaSubadm.altaExpediente.AltaSUBADM"
        singleton="false">
                <property name="altaExpedientesService"><ref bean="altaExpedientesService"/></property>
                <property name="consultaExpedienteService"><ref bean="consultaExpedienteService"/></property>
                <property name="logService"><ref bean="logService"/>            </property>
</bean>
……
```

***Fichero de validacines "XXX-validation.xml"**: fichero donde se realizan las validaciones de los campos requeridos en el formulario del alta de expediente. El fichero debe tener el mismo nombre que la clase java donde se manejan los campos de la página JSP del alta.*

AltaSUBADM-validation.xml

```
<validators>
                <field name="selTipoExpediente">
                        <field-validator type="requiredstring">
                                <param name="trim">true</param>
                                  <message>Debe seleccionar el tipo de expediente</message>
                        </field-validator>
                </field>
…..
```

6. Tareas

Las ***tareas son las unidades elementales de trabajo*** *que pueden/deben realizarse en un determinada fase de un procedimiento. La especificación de qué tareas deben realizarse y quién (usuario) debe ejecutarlas forma parte del modelado de procedimientos, y deben quedar reflejadas en la Plataforma de Tramitación.*

En Trew@, existen cuatro tipos de tareas que se enumeran a continuación:

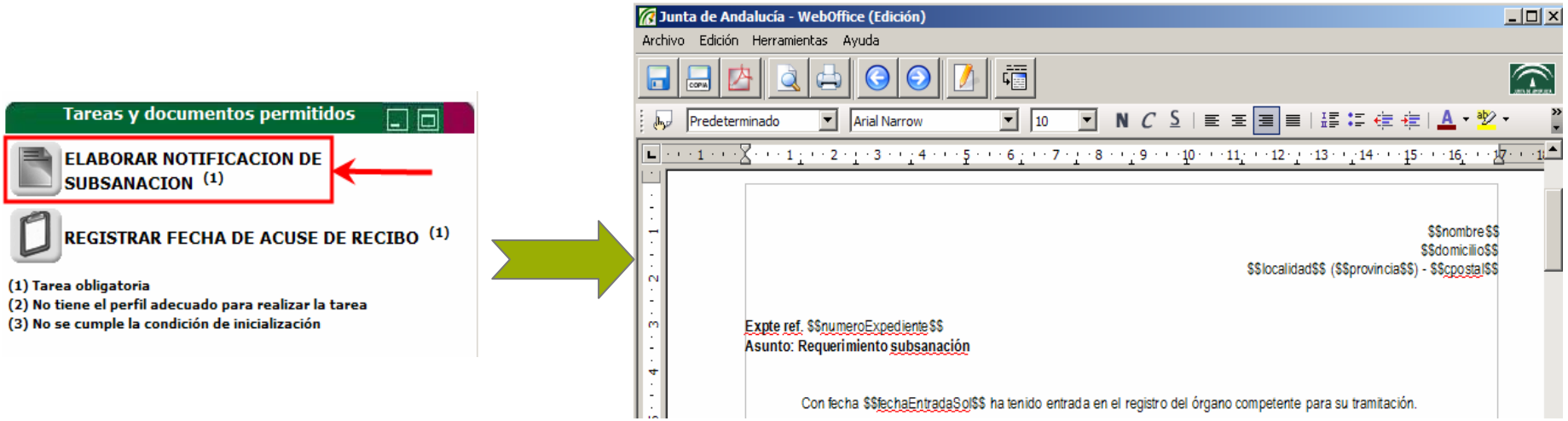
- ***Tareas de incorporar documentos***.
- ***Tareas de generación de documentos***.
- ***Tareas Web o de adquisición de datos.***
- ***Tareas Tipo Otros***

6.1. Tareas tipo Incorporar documento

Tareas de tipo incorporar documentos*: para las tareas tipo incorporar no es necesario ningún desarrollo en Plataforma de Tramitación. Una vez modelada la tarea en Trew@, automáticamente en el módulo funcional de "Documentos y Tareas a realizar" se obtendrá la tarea para incorporar documentos.*

6.2. Tareas tipo Generar documento

***Tareas de tipo generar documentos**: una vez modelada este tipo de tarea en Trew@, en el módulo funcional de "Documentos y Tareas a realizar" se obtendrá la tarea para generar documentos. El documento generado es una plantilla en WebOffice con un conjunto de variables.*



6. Tareas tipo Generar documento (II)

***Tareas de tipo generar documentos**: el único desarrollo necesario en las tareas de generación de documentos es el cálculo de variables presentes en dicho documento. Estas variables tienen el formato $$Nombre_de_la_variable$$. En Trew@, cada variable tiene asociada una clase Java y un método. Será necesario definir la clase y el método de forma que dicho método devuelva la cadena de caracteres que sustituirá a esa variable.*

Variables

Nuevo | Editar | Borrar | Buscar Registros 1 a 10 de 129 [1 2 3 4 5 6 7 8 9 10 11..13]

Nombre	Descripción	Implementación	Paquete	Función	Tipo Acto
ActuacionMGJP	Proyecto por el que se solicita la subvención	Java	es.juntadeandalucia.plataforma.variables.VariablesMGJP	obtenerActuacionMGJP	
ActuacionRMH	Proyecto por el que se solicita la subvención	Java	es.juntadeandalucia.plataforma.variables.VariablesRMH	obtenerActuacionRMH	
antecedente3Desis	antecedente3Desis	Java	es.juntadeandalucia.reas.variables.VariablesREAS	obtantecedente3Desis	
AnyoActualMGJP	Año actual del sistema.	Java	es.juntadeandalucia.plataforma.variables.VariablesMGJP	obtenerAnyoActualMGJP	
AnyoActualRMH	Año actual del sistema.	Java	es.juntadeandalucia.plataforma.variables.VariablesRMH	obtenerAnyoActualRMH	
CausaNoConcesionMGJP	Indica la causa por la cual no se ha concedido la subvención solicitada	Java	es.juntadeandalucia.plataforma.variables.VariablesMGJP	obtenerCausaNoConcesionMGJP	
CausaNoConcesionRMH	Indica la causa por la cual no se ha concedido la subvención solicitada	Java	es.juntadeandalucia.plataforma.variables.VariablesRMH	obtenerCausaNoConcesionRMH	
cifEmpresa	cifEmpresa	Java	es.juntadeandalucia.reas.variables.VariablesREAS	obtcifEmpresa	
cnaeEmpresa	cnaeEmpresa	Java	es.juntadeandalucia.reas.variables.VariablesREAS	obtcnaeEmpresa	
CodPostalMGJP	Código Postal introducido en la solicitud de subvención	Java	es.juntadeandalucia.plataforma.variables.VariablesMGJP	obtenerCodPostalMGJP	

VariablesCJAP.java

```
public class VariablesCJAP extends trewa.ext.TrAccesoUI {

public String obtcifEmpresa(Integer id)throws TrException
{
   String res = "";
   try{
        TrAPIUI apiui = getApiUI();
        String xml_cad = apiui.obtenerOtrosDatos(new TpoPK(id), "E");
         OtrosDatosExpedientes datosExp = null;
         datosExp = new OtrosDatosExpedientes(xml_cad);
         res = datosExp.getCifEmpresa();
         if(res == null)
              res = "";

         return res;
  }  catch (Exception e) {
       return "";
   }
 }
.....
```

6.3. Tareas Web o de adquisición datos y tipo Otros

***Tareas de tipo WEB y tipo OTROS**: Para estos tipos de tareas será necesario programación. Se utilizará un servicio creado para en PT-w@ndA llamado tareasService. Dicho servicio se encargará de obtener y guardar toda la información necesaria para realizar las tareas tipo web y otros..*

***"struts-xxxxx.xml"**: Inicialmente ha de crearse este fichero en el que se encuentran definidas todas las tareas. debiendo incluirse posteriormente dicho fichero en el fichero struts.xml. (*<include file="struts-xxxxxx.xml"/>*).*

struts-tareaWEB.xml

```
<!-- Fase: Validacion de documentacion  -->
<action name="tareaEntradaINS_VALIDAR_DOCUMENTACION" method="mostrarValidacionDoc"
class="tareaINS_Action">
<result name="success">tareaEntradaINS_VALIDAR_DOCUMENTACION.jsp</result>
<interceptor-ref name="basicStack" />
</action>

<action name="guardarValidacionDocINS" method="guardarValidacionDoc"
class="tareaINS_Action">
<result name="success">Correcto.jsp</result>
<result name="error">Error.jsp</result>
<result name="recarga">tareaEntradaINS_VALIDAR_DOCUMENTACION.jsp</result>
<interceptor-ref name="basicStack" />
</action>
…
```

6.3. Tareas Web o de adquisición datos y tipo Otros (II)

Para tareas Web o de Adquisición de datos, además de definir la regla de Struts para mostrar la página JSP deseada, es necesario crear otro "action" en el fichero struts-xxxxx.xml donde se definirá una clase y un método donde se tratarán los datos introducidos en el formulario, de forma similar al proceso de Alta de Expediente.

Para realizar una tarea de tipo Otros, primero es necesario definir un "action" con el mismo nombre que tenga la tarea en Trew@, en el fichero struts-xxxxx.xml. Mediante esa regla, se indicará la página Web que debe mostrarse. En este tipo de tareas se muestra un mensaje con la descripción de la tarea indicada en Trew@. La siguiente página JSP también es necesaria desarrollarla. .

struts-tareaWEB.xml

```
<!-- Fase: Requerimiento de Subsanacion -->
<action name="tareaEntradaINS_REG_FA_REQUERIMIENTO_SUBSANACION" method="inicio"
class="tareaINS_Action">
<result name="success">tareaEntradaINS_REG_FA_REQUERIMIENTO_SUBSANACION.jsp</result>
<interceptor-ref name="basicStack" />
</action>
```

6.3. Tareas Web o de adquisición datos y tipo Otros (III)

JUNTA DE ANDALUCIA
CONSEJERÍA DE JUSTICIA Y ADMINISTRACIÓN PÚBLICA

"tareaEntradaINS_REG_FA_REQUERIMIENTO_SUBSANACION.jsp": *Ejemplo de JSP para tarea WEB.*

 # tareaEntradaINS_REG_FA_REQUERIMIENTO_SUBSANACION.jsp

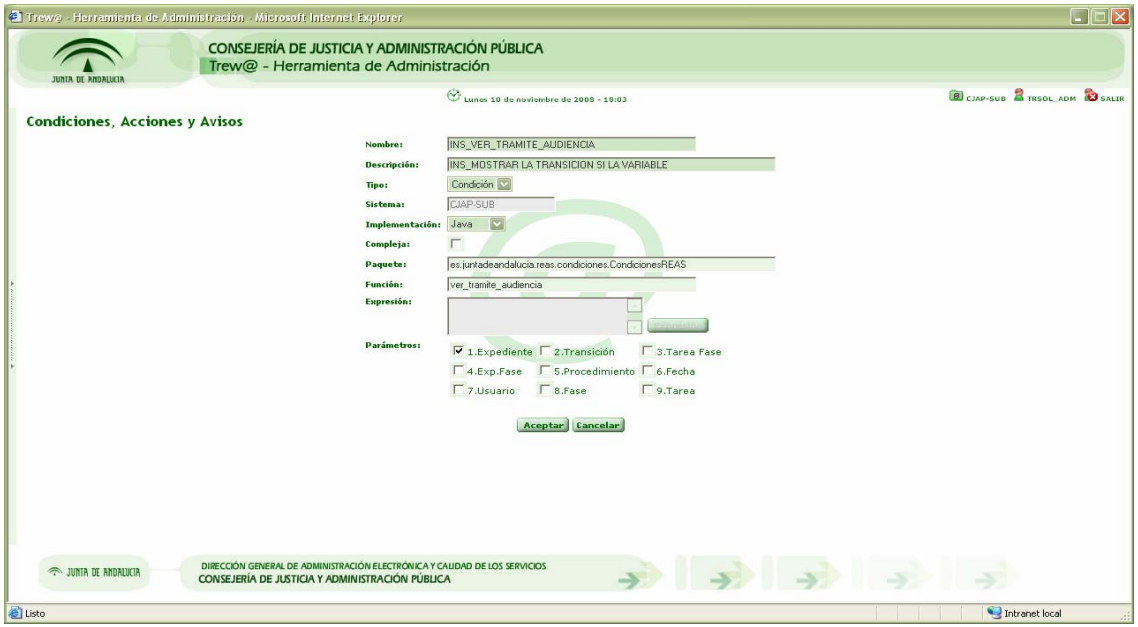
```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<%@ taglib prefix="s" uri="/struts-tags"%>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title><s:text name="Registrar Fecha - Acuse Recibo de la notificacion del documento de requerimiento de subsanación" /></title>

<!-- HOJAS DE ESTILOS -->
<link rel="stylesheet" type="text/css" href="../../instalaciones/css/hojaEstiloAplicacion.css" />
<s:head theme="ajax"/>
</head>
<body>
<s:div id="capa_ejecutarTarea">
<center>
<div id="contenedorCabecera">
<div id="cabeceraPaginaAplicacion"></div>
<div id="rellenoCabeceraPagina"></div>
</div>
<s:actionerror />
<s:fielderror  />
</center>

<s:div id="espacioAlto"> 
</s:div>
<br>
<s:div id="bordePpal" cssStyle="center">
</s:div>

...
```

7. Condiciones

La programación de una condición consiste en habilitar una de varias transiciones en función del valor de una variable obtenida en una tarea web o en el alta del expediente.

Los pasos a seguir para la programación de Condiciones son:

1. Definir la Condición en TREW@: indicando el nombre de la condición a implementar, la ruta del paquete en la que se va a encontrar y el nombre de la función que ejecutará la lógica para decidir si se cumple o no la condición, habilitando o deshabilitando la condición.



7. Condiciones (II)

2. Implementar un método: se debe programar la clase que se ha indicado en el paso uno, e implementar el método para que, en función de esa variable, devuelva un ‘1’ o un ‘0’. Dependiendo del valor del método se habilitará o deshabilitará la condición.

CondicionesCJAP.java

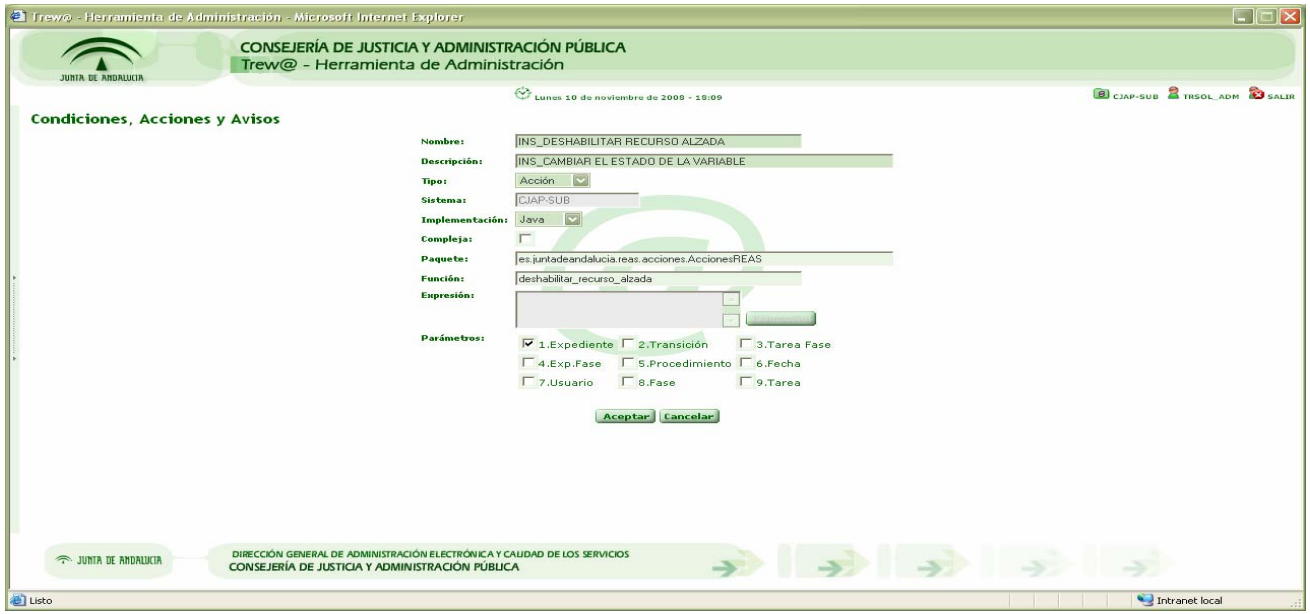
```
public class CondicionesCJAP  {
public Integer  ver_tramite_audiencia(BigDecimal id){
        Integer resultado=0;
        String ver_tram_aud;
         try {
          xml=apiUI.obtenerOtrosDatos(new TpoPK(id), "E");
          otrosDatos = new OtrosDatosExpedientes(xml);
          ver_tram_aud=otrosDatos.getHabilitarTramiteAudiencia();
          if (ver_tram_aud.equals(null))
                        return 0;
          else {
                        if (ver_tram_aud.equals("1"))
                                resultado = 1;
                        else
                                resultado = 0;
              }
              }
      catch (ArchitectureException e) {
         return 0; }
      return resultado;
}
```

8. Acciones

Las acciones son eventos lanzados cuando ocurre un hecho indicado en TREWA y a partir del cual se puede realizar lo que sea necesario para el correcto funcionamiento del procedimiento. Se pueden asociar las acciones a tareas o a transiciones.

Los pasos a seguir para la programación de Acciones son:

1. Definir la Acción en TREW@: indicando el nombre de la acción a implementar, la ruta del paquete en la que se va a encontrar y el nombre de la función que ejecutará la lógica para decidir si se cumple o no la acción.



2. Implementar un método: se debe programar la clase que se ha indicado en el paso uno, e implementar el método para que, en función de esa variable, devuelva un '1' indicando que se ha ejecutado correctamente toda la lógica implementada en la acción.

AccionesCJAP.java

```
public class AccionesCJAP  {

 public String deshabilitar_recurso_alzada(BigDecimal id){
      try{
            datosXml = apiui.obtenerOtrosDatos(new TpoPK(id), "E");
            OtrosDatosExpedientes nuevos = new OtrosDatosExpedientes (datosXml);
            nuevos.setHabilitarRecursoAlzada("0");
            apiui.actualizarOtrosDatos(new TpoPK(id), "E", nuevos.toString());
            apiui.commit();
      }catch(Exception e){
            System.out.println("Error en cDEVFIA_AC_REG_RECURSOS()");
      }
            return "1";
}
```

9. Desarrollo de Utilidades

Las utilidades son herramientas, disponibles desde el escritorio de tramitación de Plataforma de Tramitación, que añaden funcionalidad adicional para el usuario, como por ejemplo el acceso al applet de Model@ para seguir el flujo del procedimiento, la posibilidad de adjuntar un documento al expediente en cualquier fase del mismo o mostrar la evolución del expediente.

9.1. Especificaciones para la construcción de una nueva utilidad

Para el desarrollo de una nueva utilidad visible desde el escritorio de tramitación de Plataforma de Tramitación w@ndA, será necesario crear y/o modificar los siguientes ficheros:

Ficheros a modificar / crear:

1. Crear un fichero en Struts-2, para la Utilidad correspondiente.

2. Modificar el fichero struts.xml de PT-W@nda para añadir la ruta del fichero anterior.

3. Crear una clase Java que contendrá toda la lógica de negocio de la utilidad.

4. Crear un fichero JSP que será la página que se lanzará al pulsar sobre la utilidad.

5. Añadir al fichero applicationContext-XXXX.xml que utilice PT-W@nda la nueva clase creada.

9.1. Especificaciones para la construcción de una nueva utilidad (II)

JUNTA DE ANDALUCIA
CONSEJERÍA DE JUSTICIA Y ADMINISTRACIÓN PÚBLICA

Crear un fichero de Struts-2: este permite separar la capa de negocio de la capa de presentación, es decir, en este fichero se relaciona la ventana JSP de la utilidad con la clase que se encargará de toda la lógica de negocio, siendo este XML el punto de unión.

struts-utilidadCJAP.xml

```
<struts>
   <constant name="struts.objectFactory" value="spring" />
   <constant name="struts.devMode" value="true" />
<package name="datosExpedienteReas" namespace="/modulos/datosExpedienteReas“
   extends="struts-default">
<global-results>
<result name="error">/administracion/error.jsp</result>
</global-results>
<!-- UTILIDAD : MODIFICAR DATOS EXPEDIENTE -->
<action name="datosExpedienteReas" method="datosExpedienteReas"
class="datosExpedienteReasAction">
<result name="ins">/modulos/datosExpedienteReas/datosExpedienteReasINS.jsp</result>
<result name="cer">/modulos/datosExpedienteReas/datosExpedienteReasCER.jsp</result>
<result name="ext">/modulos/datosExpedienteReas/datosExpedienteReasEXT.jsp</result>
<result name="can">/modulos/datosExpedienteReas/datosExpedienteReasCAN.jsp</result>
<result name="mod">/modulos/datosExpedienteReas/datosExpedienteReasMOD.jsp</result>
<result name="cao">/modulos/datosExpedienteReas/datosExpedienteReasCAO.jsp</result>
<result name="ren">/modulos/datosExpedienteReas/datosExpedienteReasREN.jsp</result>

<result name="error">/modulos/datosExpedienteReas/Error.jsp</result>
<interceptor-ref name="basicStack" />
</action>

<!-- INS -->
<action name="guardarDatosReasIns" method="guardarDatosReasINS"
class="datosExpedienteReasAction">
<result name="success">/modulos/datosExpedienteReas/Correcto.jsp</result>
<result name="error">/modulos/datosExpedienteReas/Error.jsp</result>
<interceptor-ref name="basicStack" />
</action>
```

Modificar el fichero struts.xml de PT-W@nda para añadir la ruta del fichero anterior.

struts.xml

```
<include file="/struts/modulos/datosExpedienteReas/struts-datosExpedienteReas.xml"/>
```

Programación de una clase Java: que contendrá toda la lógica de negocio de la utilidad. En ella se incluirán todos los métodos referenciados en el fichero de struts de la utilidad, además del resto de métodos auxiliares que sean necesarios.

DatosExpedienteCJAPAction.java

```
public DatosExpedienteCJAPAction(){
}
public String datosExpedienteCJAP(){
      int version;
      ResourceBundle propiedades = ResourceBundle.getBundle("cjap");
      UsuarioWeb user = (UsuarioWeb) session.get("usuario_en_sesion");
      IExpediente exp = user.getExpediente();

      session.put("expediente",exp.getRefExpediente());
      try  {
            //Obtenemos el id del procedimiento del expediente desde el que se
            //carga la utilidad
            version = Integer.parseInt(exp.getProcedimiento().getRefProcedimiento());

     .......
```

Programar un fichero JSP: que será la página que se lanzará al pulsar sobre la utilidad. Para ello, en el fichero de struts de la utilidad es necesario definir un action que relacione un método definido en la clase Java anterior y la página JSP, de forma que el método devuelva un valor que lance la JSP.

Modificar el fichero applicationContext-XXXX.xml: que utilice PT-W@nda un nuevo "Bean" definiendo el identificador de la clase Java, así como todos los servicios que ésta use.

applicationContext.xml

```
<bean id="datosExpedienteCJAPAction"
class="es.juntadeandalucia.plataforma.modulos.datosExpedienteReas.DatosExpedi
enteCJAPAction" singleton="false">
        <property name="logService">
                <ref bean="logService"/>
        </property>
        <property name="consultaService">
                <ref bean="consultaExpedienteService"/>
        </property>
        <property name="tareasService">
                <ref bean="tareasService"/>
        </property>
</bean>
```

10. Procedimiento de Creación de Nuevos Módulos Funcionales

En el presente apartado, se detallan las especificaciones y pautas a seguir para la construcción de nuevos componentes funcionales que podrán ser incorporados a Plataforma de Tramitación w@ndA.

Dentro de Plataforma de Tramitación se pueden distinguir tres tipos distintos de módulos funcionales:

- ***Portlet**: Son módulos cuya funcionalidad es accesible a través de un portlet en el escritorio de tramitación.*

- ***Externo**: Son módulos cuya funcionalidad es accesible desde cualquier punto de la aplicación,es decir, son aplicaciones externas independientes que podrían ser accesibles vía URL.Dentro de este tipo se incluyen los de tipo Menú, cuya funcionalidad puede ser accesible desde cualquier menú definido en la aplicación.*

- ***Utilidades**. (Apartado anterior).*

10.1. Construcción de un módulo funcional mediante un fichero “.zip”

Todos los componentes funcionales deben seguir un formato y estructura predeterminada y empaquetarla en un fichero comprimido en formato ZIP.

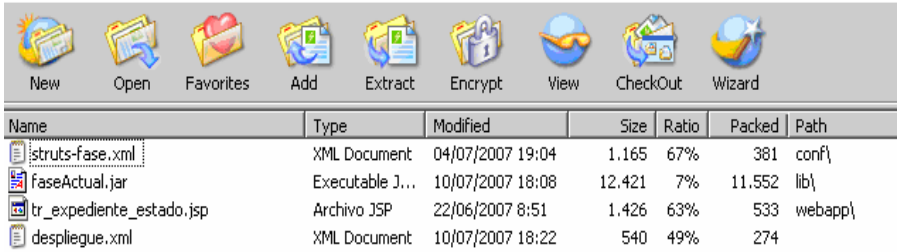
Dicha estructura se detalla a continuación:

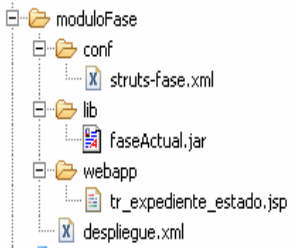
- *El fichero de despliegue denominado ‘despliegue.xml’. Este fichero es obligatorio.*
- *La carpeta ‘conf’ que contendrá los ficheros de configuración del módulo (ficheros de struts). Esta carpeta debe existir en el fichero ZIP, aunque puede encontrarse vacía.*
- *La carpeta ‘lib’ que contendrá las librerías y dependencias especificas del componente funcional y que aún no están disponibles en plataforma. Esta carpeta debe existir en el fichero ZIP, aunque puede encontrarse vacía.*
- *La carpeta ‘webapp’ que contendrá los ficheros JSP, JavaScript, imágenes, css, etc… que complementan la construcción del módulo. Esta carpeta debe existir en el fichero ZIP, aunque puede encontrarse vacía.*

En las siguientes ilustraciones se muestra un ejemplo del contenido de un fichero “.zip” referente a un módulo funcional.

La carpeta ‘conf’ que contendrá los ficheros de configuración del módulo (ficheros de struts). Esta carpeta debe existir en el fichero ZIP, aunque puede encontrarse vacía.

Carpeta Lib. En esta carpeta deben estar contenidas todas las librerías que se deben usar y que son específicas del componente funcional. Se pueden adjuntar tanto aquellas generadas por su módulo como algunas librerías necesarias para su correcto funcionamiento.Todas las clases que componen un módulo deben ir en un fichero .jar, y no pueden existir clases por separado.





El fichero de despliegue denominado ‘**despliegue.xml**’. Es el fichero descriptor donde se detallan aspectos descriptivos e informativos del módulo.

Los campos que estructuran este documento en formato xml son:

- ***Nombre**: Nombre del módulo.*
- ***Versión**. Versión del módulo.*
- ***Título**: Título del módulo.*
- ***Descripción**: Breve descripción de la funcionalidad del módulo.*
- ***Autor**: Datos del autor del módulo.*
- ***Type**: Existen 2 tipos diferentes struts-1 y struts-2.*
- ***Postfuncion**: Permite ejecutar una función javascript tras la recarga del módulo.*
- ***Dependencias**: Describe las dependencias del módulo con otros módulos.*
- ***Observados**: Indica los módulos que tiene que “observar” para actualizarse cuando alguno de esos módulos sufre una actualización. (Solo tiene sentido para los módulos que tienen dependencias).*

La carpeta ‘webapp’ que contendrá los ficheros JSP, JavaScript, imágenes, css, etc… que complementan la construcción del módulo. Esta carpeta debe existir en el fichero ZIP, aunque puede encontrarse vacía.

En esta carpeta, deben estar contenidos todos los ficheros jsp, JavaScript, imágenes, etc… para el correcto funcionamiento del módulo. Se recomienda:

- *Definir una cierta estructura de carpetas, como por ejemplo situar todos los ficheros javascript de una carpeta que cuelgue de webapp y que se denomine “js”, al igual que las imágenes en una carpeta llamada “imágenes” o “img” que cuelgue del directorio webapp, los ficheros jsp pueden colgar directamente del directorio webapp.*

- *Todas las rutas que se especifiquen en nuestro nuevo módulo deben ser rutas relativas.*

Requisitos previos a la construcción de procedimientos

- *Es necesario disponer de una instalación válida de Trew@ con la definición del procedimiento que va a im implementar cargada en ella.*
- *Para la plataforma de tramitación, se requiere de una instancia de cualquier motor de base de datos relacional compatible con Hibernate 3 (por ejemplo, MySQL, PostgreSQL, Oracle, etc.).*

Debido a que frecuentemente será necesario reiniciar la información relativa a módulos desplegados en la plataforma de tramitación, configuraciones, etc., se recomienda que cada desarrollador tenga una instancia local del SGDB utilizado para almacenar el esquema de datos de la plataforma.

Preparación del entorno de desarrollo

Para el desarrollo de procedimientos se partirá de la última versión estable del kit de desarrollo de la plataforma de tramitación w@ndA. Dicho kit viene empaquetado en un fichero comprimido (.zip) que contiene:

- *Las librerías utilizadas por la plataforma de tramitación. Entre ellas, destaca el fichero escritorio-core.jar, que contiene la lógica de la aplicación.*
- *Páginas web del escritorio de tramitación. Normalmente se tratará de ficheros JSP (Java Server Pages).*
- *Recursos web de la aplicación: imágenes, hojas de estilo, archivos con funciones JavaScript.*
- *Archivos de configuración, tanto de la propia aplicación como de los diferentes frameworks utilizados en su construcción (Struts, Spring, Hibernate, log4java, etc.).*

Estructura lógica de la información

Para la construcción de un nuevo procedimiento en PT-W@nda se distinguen 3 tareas:

Tarea	Descripción de la Entidad	Ficheros
Módulo - Alta Expediente	Módulo para realizar el alta de expediente.	Librerias utilizadas,páginas web y recursos de configuración necesarios(ficheros de struts, spring y archivo de despliegue de módulo)
Módulo - Tareas	Módulo para realizar las tareas del expediente.	Librerias utilizadas,páginas web y recursos de configuración necesarios(ficheros de struts, spring y archivo de despliegue de módulo)
Módulo – Utilidades	Módulo con las utilidades del procedimiento.	Librerias utilizadas,páginas web y recursos de configuración necesarios(ficheros de struts, spring y archivo de despliegue de módulo)

Para el desarrollo de un módulo hay que tener en cuenta la ubicación de los fichero dentro del sdk. Se distinguirán 4 grandes apartados a la hora de estructurar los ficheros de un módulo:

- *configuración: ficheros de configuración tanto de struts2 como de spring en el caso de que se usara.*
- *java: clases java para el desarrollo del módulo junto con sus ficheros properties, xml, etc. que fueran necesarios.*
- *recursos jsp: ficheros jsp del módulo*
- *css e imágenes: estilos e imágenes usadas en el módulo*

JUNTA DE ANDALUCIA
CONSEJERÍA DE JUSTICIA Y ADMINISTRACIÓN PÚBLICA

Estructura lógica – Configuración

STRUTS 2	Su ubicación debe ser <classpath>:/modulos/<NOMBRE_MODULO>
	La nomenclatura debe ser struts-*.xml
	Habrá un fichero de configuración por cada tipo de acción, esto es: -configuración del alta -configuración de las tareas -Configuracion de las utilidades
	El fichero de configuración para el alta debe tener el namespace: <ABREVIATURA_PROCEDIMIENTO>/altaExpediente
	El fichero de configuración para las tareas debe tener el namespace: agenda/tareas
	El fichero de configuración para las utilidades debe tener el namespace: <ABREVIATURA_PROCEDIMIENTO>/utilidades
SPRING	Su ubicación debe ser <classpath>:/modulos/<NOMBRE_MODULO>
	La nomenclatura debe ser applicationContext.xml
	Sólo habrá un fichero de configuración

10.2. Construcción de un Procedimiento (IV)

Estructura lógica – JAVA: El paquete base de cualquier fichero java/properties/config será paquete.**<ABREVIATURA_PROCEDIMIENTO>,** a partir de esta ruta tendremos los siguientes paquetes:

PAQUETE	DESCRIPCIÓN
altaExpediente	ficheros relacionados con el proceso de alta de un expediente
Tareas	ficheros relacionados con las tareas de un expediente
utilidades	ficheros relacionados con las utilidades
condiciones	ficheros relacionados con las condiciones
Acciones	ficheros relacionados con las acciones
Variables	ficheros relacionados con las variables
xmlExpediente	ficheros relacionados con la estructura XML de otros datos de expedientes
Dao	ficheros relacionados con el acceso a BBDD
Dto	tipos propios de datos
Config	ficheros de configuración (xml,betwin,etc)
properties	ficheros properties
Actions	ficheros de acciones de struts

Estructura lógica – Recursos jsp: La ruta base de cualquier recurso jsp será: /modulos/<**ABREVIATURA_PROCEDIMIENTO**>. A partir de aquí tendremos los siguientes bloques :

BLOQUE	DESCRIPCIÓN
altaExpediente	jsp relacionadas con el alta de un expediente
tareas	jsp relacionadas con las tareas de un expediente
utilidades	jsp relacionadas con las utilidades
decorators	jsp para la decoración

Estructura lógica – css e imágenes

- *La ubicación de las hojas de estilo será:* ***/instalaciones/<NOMBRE_INSTALACION>/css.***
- *La ubicación de las imágenes será:* ***/instalaciones/<NOMBRE_INSTALACION>/imágenes.***

Tanto para las imágenes como las hojas de estilo será importante usar un prefijo de nombre unívoco.

Empaquetamiento ZIP- Estructura

ruta	descripción
/	Raiz del zip. En esta ubicación se debe colocar el archivo de despliegue del módulo cuyo nombre debe ser **despliegue.xml**
/webapp	En esta se colocan los páginas jsp del módulo, Según la misma estructura definida en el desarrollo (a partir de la ruta /src/main/webapp/modulos/<nombreModulo>).
/webapp/CSS	En esta carpeta se deben colocar las hojas de estilos css usadas.
/webapp/IMÁGENES	En esta carpeta se deben colocar las imágenes usadas en el módulo.
/conf	En esta carpeta se deben colocar los archivos de configuración de struts, spring y los archivos properties usados. Según la misma estructura definida en el desarrollo (a partir de la ruta /src/main/config/<nombreModulo>)
/lib	En esta carpeta se deben colocar la librería jar generada con las clases del módulos y otras librerías usadas para su desarrollo.

10.2. Construcción de un Procedimiento (VI)

JUNTA DE ANDALUCIA
CONSEJERÍA DE JUSTICIA Y ADMINISTRACIÓN PÚBLICA

Empaquetamiento ZIP- Archivo de despliegue :

```
<def-modulo>
<nombre>nombreModulo</nombre>
<version>versión (ejemplo: 1.0)</version>
<titulo>título del módulo</titulo>
<descripcion>descripción del módulo</descripcion>
<url>url de acceso al action. Necesario para el módulo de alta</url>
<menu>
    <nombre>ESCRITORIO o ADMINISTRACION</nombre>
  </menu>
<autor>
    <nombre>José Manuel Garrido Barrera</nombre>
    <email>Jose.Garrido.Barrera@everis.com</email>
    <grupo>general</grupo>
</autor>
<type>STRUTS-2</type> //indica el framework usado para el desarrollo del módulo. A fecha de redacción de este documento se aseguro el correcto funcionamiento con struts-2.
<tipoInstalacion>Tipo de instalación (NONE,EXTERNO,PORTLET,UTILIDADES)</tipoInstalacion>
</def-modulo>
```

Modificación de Estilos en las pantallas de Inicio, login y menú.

Para la modificación de estilos en las pantallas de inicio, login y menú de PT-w@ndA, debemos redefinir los atributos :

- *contenedorCabecera*
- *cabeceraPaginaAplicacion*
- *rellenoCabeceraPagina*
- *ContenedorIconosCabecera*
- *TituloPagina*
- *ContenedorFondosLogin*
- *FondoTrazaLogin*
- *TextoTituloPagina*

*De la hoja de estilos **"hojaEstiloAplicacion.css“** de PT-w@ndA**.***

11. Creación y configuración de Estilos (II)

JUNTA DE ANDALUCIA
CONSEJERÍA DE JUSTICIA Y ADMINISTRACIÓN PÚBLICA

Modificación de Estilos en los portlets de PT-w@ndA

Para la modificación de estilos en las pantallas de inicio, login y menú de PT-w@ndA, debemos redefinir los atributos :

- *barra_titulo*
- *titulo_portlet*
- *capaFondo*
- *portlet_fondo_tarea*
- *contenido_tabla_portlet*
- *texto_titulo_portlet*
- *...*

*De la hoja de estilos **"portlets.css"** de PT-w@ndA.*

11. Creación y configuración de Estilos (III)

Modificación de Estilos en las pantallas de Inicio, login y menú de la Administración de PT-w@ndA

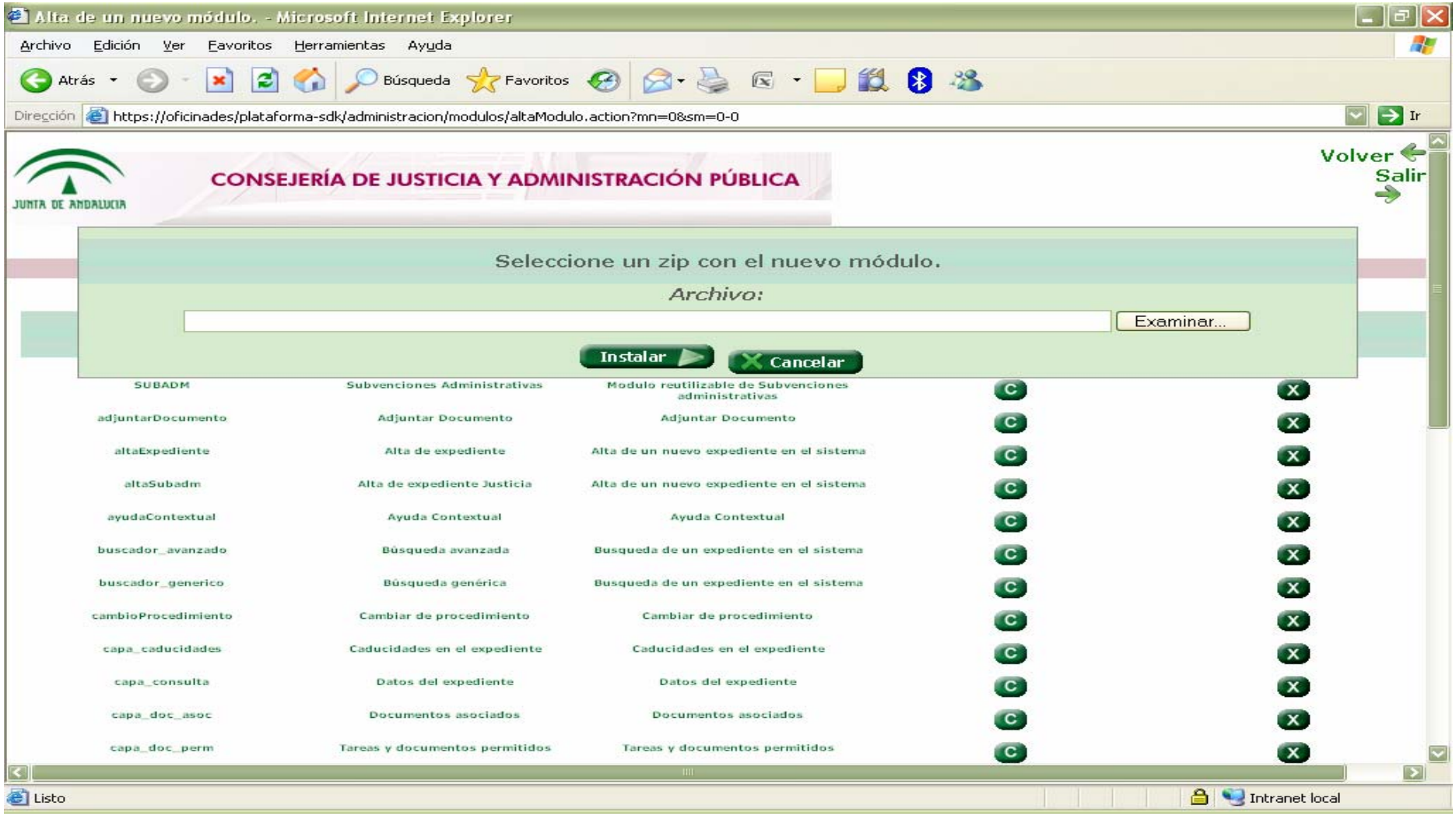
Para la modificación de estilos en las pantallas de inicio, login y menú en la administración de PT-w@ndA, debemos redefinir los atributos :

- *contenedorCabecera*
- *cabeceraPaginaAplicacion*
- *rellenoCabeceraPagina*
- *ContenedorIconosCabecera*
- *TituloPagina*
- *ContenedorFondosLogin*
- *FondoTrazaLogin*
- *TextoTituloPagina*

*De la hoja de estilos **"hojaEstiloAdmon.css“** de PT-w@ndA.*

12. Herramienta de Administración en PT_w@ndA

Ejemplo de interfaz gráfica, para el alta de un módulo vertical desde una de las opciones del módulo de Adminitración de PT_w@ndA



JUNTA DE ANDALUCIA
CONSEJERÍA DE JUSTICIA Y ADMINISTRACIÓN PÚBLICA

Ejercicio Práctico

GRACIAS POR SU ATENCIÓN

JUNTA DE ANDALUCIA

CONSEJERÍA DE JUSTICIA Y ADMINISTRACIÓN PÚBLICA

Plataforma de Tramitación w@ndA